计算理论
Theory of Computation

https://courses.zju.edu.cn/course/join/8P7F9K7S0KL

第2章 正则语言和有限自动机
Regular Language and Finite Automata

计算理论(周四第7… 内部

该群属于“浙江大学”内部群，仅组织内部成员可以加入，如果组织外部人员收到此分享，需要先申请加入该组织。

杨莹春

yyc@zju.edu.cn

浙江大学曹光彪西楼-201

2021年9月30日 钉群 34658556

内容安排

– 3 classes（9/16,9/23,9/26）
  Sets, Relations and Language（CH1）
– **3 classes(9/30,10/14,10/21)**
  Regular Language and Finite Automata (CH2)
– 3 classes(10/28,11/4, 11/11)
  Context-free Languages (CH3)
– 3 classes (11/18,11/25,12/2)
  Turing machine (CH4)
– 2 classes(12/9,12/16)
  Undecidability（CH5）
– 2 classes(12/23,12/ 30)
  Review

Exam:2022/1

计算理论

第2章 正则语言和有限自动机

Ch2. Regular Language and Finite Automata

Keywords II

- Ch2. Regular Language and Finite Automata

Finite automata and Nondeterministic finite automata,
Equivalence of finite automata and regular expressions,
Languages that are and are not regular

Homework 2:	
P60	2.1.1 2.1.2 (c)(d) 2.1.3 (c)(e)
P74	2.2.2 (a)(b) 2.2.6 (a)(b) 2.2.10
P83	2.3.4 (b) 2.3.7 (a)(b)(c)(d)
P90	2.3.5 (a) 2.4.8

Goal:

— to define increasingly powerful models of computation, more and more sophisticated devices for

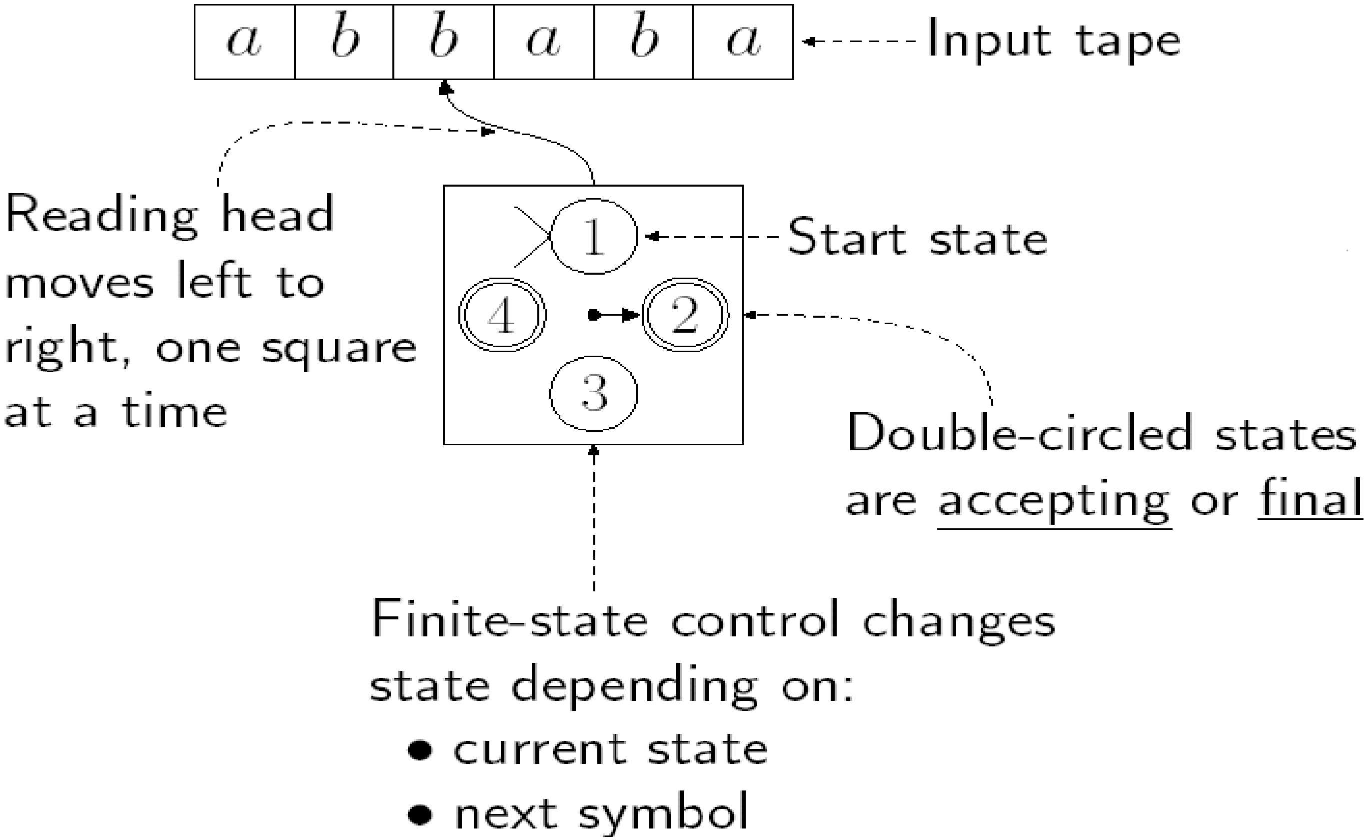
- accepting languages
- generating languages

The Chomsky hierarchy

Language type	Automata type
regular	finite
context-free	pushdown
context-sensitive	linear bounded
unrestricted	Turing Machine

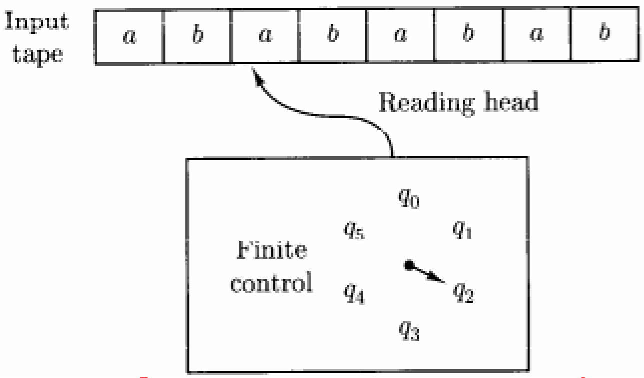
2.1 Deterministic Finite Automata

□ Introduction to DFA



Definition: A **deterministic finite automata(DFA)** is a quintuple $(K, \sum, \delta, s, F)$, where

- K is a finite set of states
- $\sum$ is an alphabet
- $s \in K$ is the initial state
- $F \subseteq K$ is the set of final states
- δ: transition function, $K \times \sum \rightarrow K$.



Remarks:

Transition function will determine **unique** next state based on current input and state.

Remark:

1) A **configuration** of a DFA $(K, \sum, \delta, s, F)$ is any element of $K \times \sum^*$.

2) The binary relation $\vdash_M$ between two configurations of M:

$$(q, w) \vdash_M (q', w') \Leftrightarrow \exists a \in \sum, w = aw' \text{ , and } \delta(q, a) = q'.$$

— say (q, w) yields (q', w') in one step.

3) The reflexive, transitive closure of $\vdash_M$:$\vdash_M^*$
$(q, w) \vdash_M^* (q', w') \Leftrightarrow (q, w)$ yields (q', w') after some number, possibly zero, of steps.

4) A **string** $w \in \sum^*$ **is said to be accepted by** M iff 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.

Example: $\sum = \{a, b\}$

$K = \{q_0, q_1\}$

$s = q_0$

$F = \{q_0\}$

Transition function

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

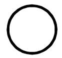
Consider the string $aabba$:

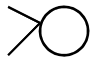
$$q_0 \xrightarrow[a]{} q_0 \xrightarrow[a]{} q_0 \xrightarrow[b]{} q_1 \xrightarrow[b]{} q_0 \xrightarrow[a]{} q_0$$

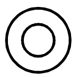
— q_0 is final, $aabba$ is accepted.

□ Graphical representation

— State Diagram

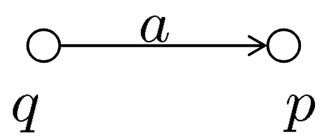
State 

Initial state 

Final state 

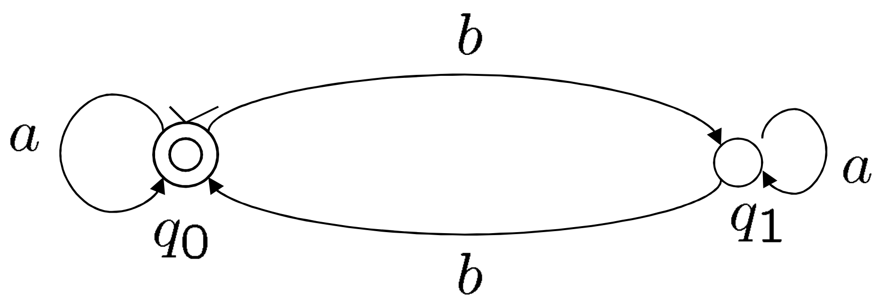
Transition function:

$\delta(q, a) = p$



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

The DFA in example can be represented:



Example:

Let us design a DFA M that accepts the language

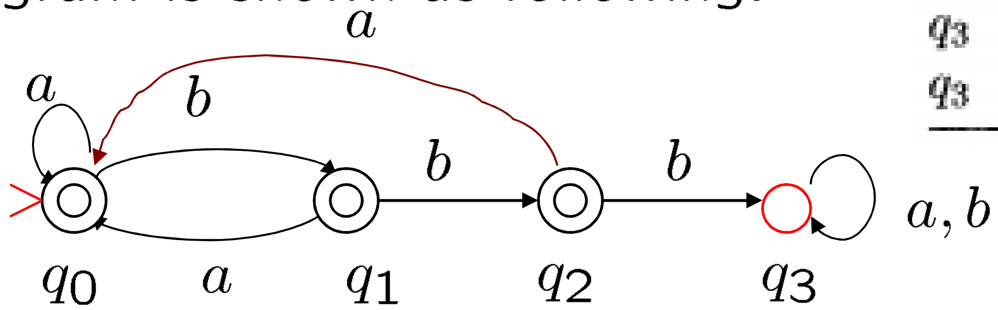
$L_1 = \{w \in \{a,b\}^* : w$ does not contain three consecutive b's$\}$.

Solution: Let DFA $M = (K, \sum, \delta, s, F)$ where

$$K = \{q_0, q_1, q_2, q_3\}$$
$$\sum = \{a, b\}$$
$$s = q_0$$
$$F = \{q_0, q_1, q_2\}$$

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_2	a	q_0
q_2	b	q_3
q_3	a	q_3
q_3	b	q_3

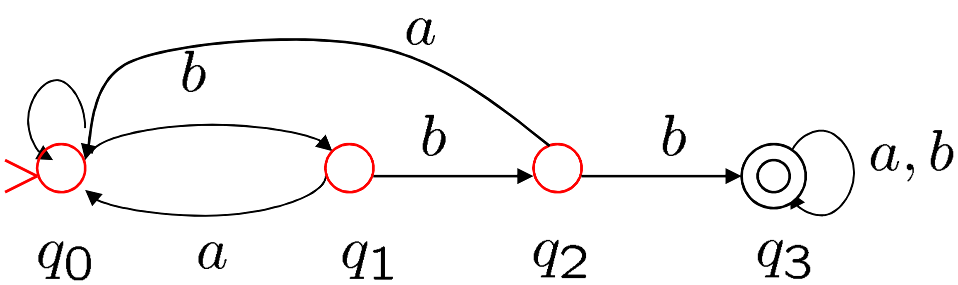
The state diagram is shown as following:



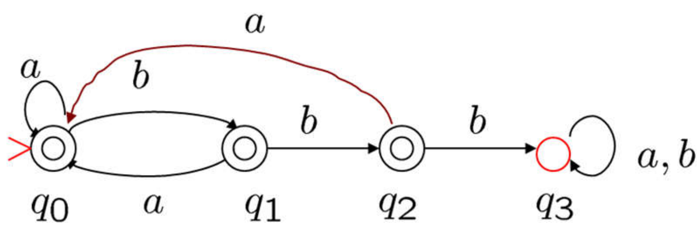
Question:

$L_2 = \{w \in \{a,b\}^* : w$ contains three consecutive b's$\}$.

The DFA that accept language L_2?



Note: $L_2 = \Sigma^* - L_1$.

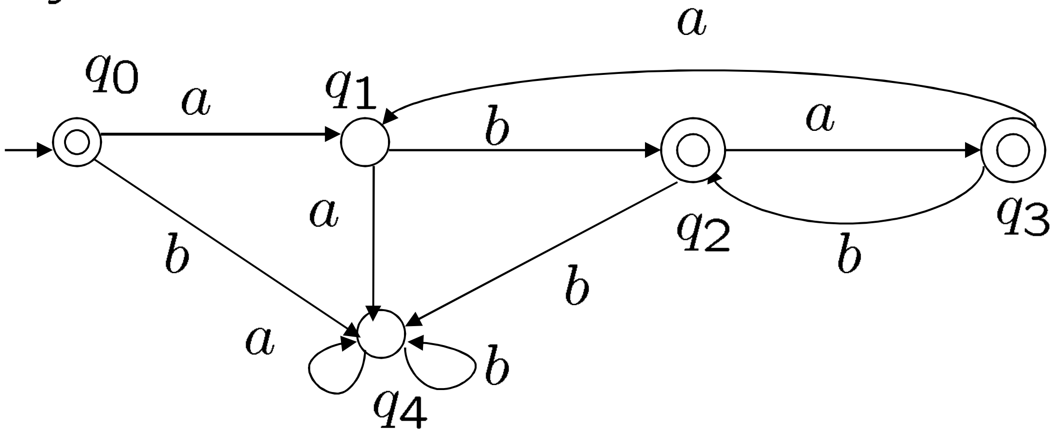


2.2 Nondeterministic Finite Automata

Generalization of determinism:

- Many "next-states"
- Computation is a " tree "
- Acceptance: ∃ a path to accepting leaf

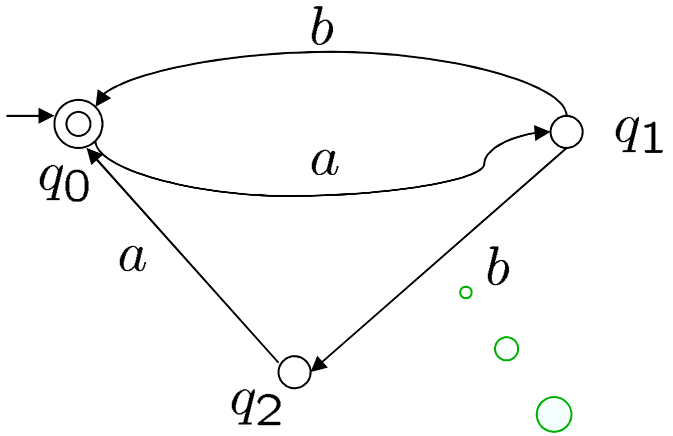
Example: Consider the language $L = (ab \cup aba)^*$ which is accepted by the DFA :



Note:

- No DFA with fewer than **five** states can accept this language.
- It is not easy to ascertain that a DFA is shown.
- L is accepted by the simple Nondeterministic Finite Automata:

$$M = (K, \sum, \delta, s, F), \text{ where}$$
$$K = \{q_0, q_1, q_2\},$$
$$S = \{a, b\},$$
$$s = q_0,$$
$$F = \{q_0\}$$



$\delta(q_1, b) = \{q_0, q_2\}$

$\delta(q_0, b) = \emptyset$

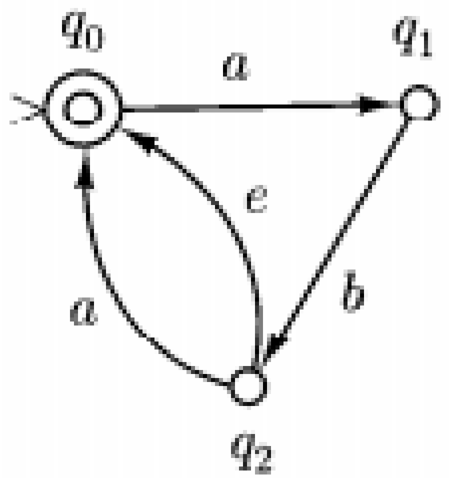


Figure 2-6

- Consider input aba
- In state diagram of a NFA arrows that are labeled by the empty string e.

Definition: A **nondeterministic finite automata(NFA)** is a quintuple $(K, \sum, \Delta, s, F)$, where

- K is a finite set of states
- $\sum$ is an alphabet
- $s \in K$ is the initial state
- $F \subseteq K$ is the set of final states
- Δ, transition relation, is a subset of $K \times (\sum \cup \{e\}) \times K$.

Definition 2.1.1: A **deterministic finite automaton** is a quintuple $M = (K, \Sigma, \delta, s, F)$ where

K is a finite set of **states**,
Σ is an alphabet,
$s \in K$ is the **initial state**,
$F \subseteq K$ is the set of **final states**, and
δ, the **transition function**, is a function from $K \times \Sigma$ to K.

Remark:

For DFA, the transition function δ is a function, but for NFA, Δ only a relation.

- triple $(q, u, p) \in \Delta$ — transition of M:

  $u \neq e$

  $u = e$: no input symbol

- configuration — an element of $K \times \sum^*$.

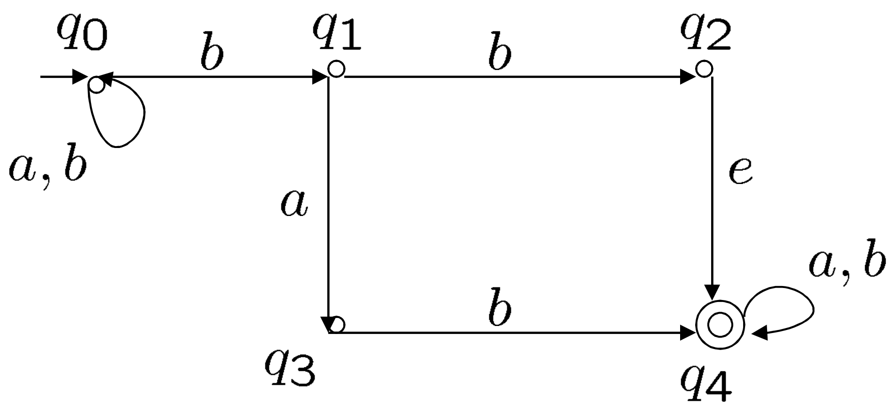
- binary relation $\vdash_M$ and its reflexive transitive closure $\vdash_M^*$.

- A **string** $w \in \sum^*$ **is accepted by** M iff 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.

Example:

$L = \{w \in \{a,b\}^* : w$ contains the pattern bb or $bab\}$.

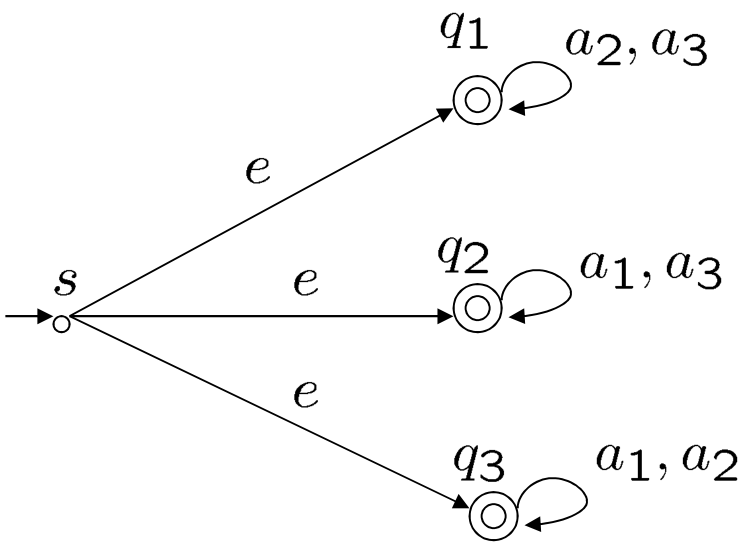


- Consider the string $bababab \in L$?

Example:

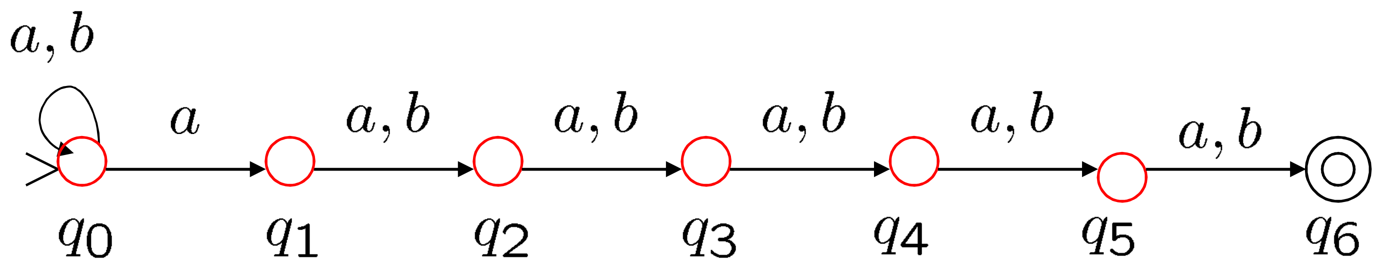
$\sum = \{a_1, \cdots, a_n\}$, where $n \geq 2$.

$L = \{w \in \sum^* : \exists a_i$ not appearing in $w\}$.



Example:

$L = \{w \in \{a, b\}^* : w$ 6^{th} symbol from end is an "a"$\}$.

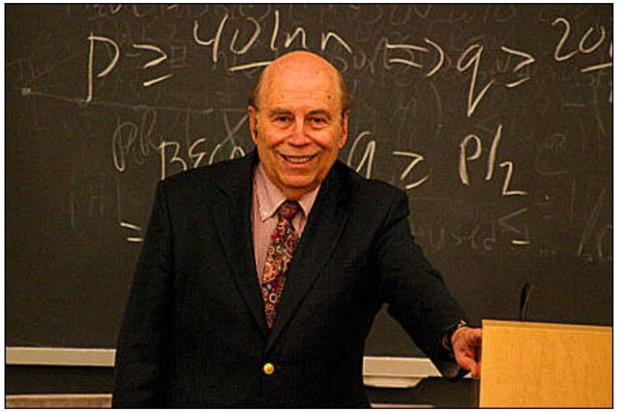


□ Expressive power of NFAs vs. DFAs

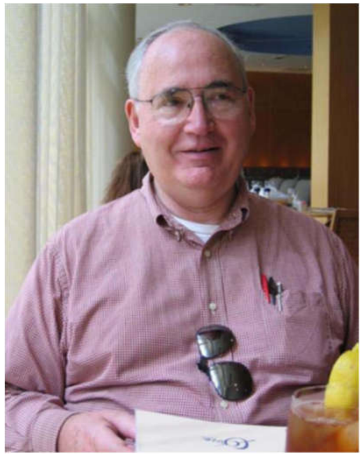
- Every DFA is an NFA
- NFAs allow more transitions

Is the language more expressive?

– *Is there a language* $L = L(M)$ *for some NFA* M*, where* $L = L(M')$ *for some DFA* M'*?*



Michael Oser Rabin（1931-）



Dana Stewart Scott (1932–)

Turing Award Citation（1976）: For their joint paper "Finite Automata and Their Decision Problem," which introduced the idea of nondeterministic machines, which has proved to be an enormously valuable concept. Their (Scott & Rabin) classic paper has been a continuous source of inspiration for subsequent work in this field.

□ NFA/DFA Equivalence

Definition: Two FA M_1 and M_2 (deterministic or non deterministic) are **equivalent** iff $L(M_1) = L(M_2)$.

- To prove equivalence of DFAs and NFAs we must do two things:

I: For each DFA, produce an NFA that accepts the same language

II: For each NFA, produce a DFA that accepts the same language

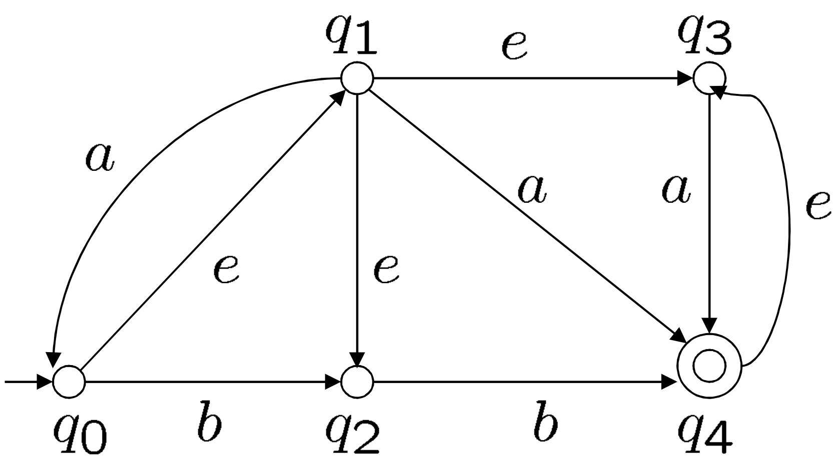
We'll do part II; part I is immediate.

Theorem: For each NFA, there is an equivalent DFA.

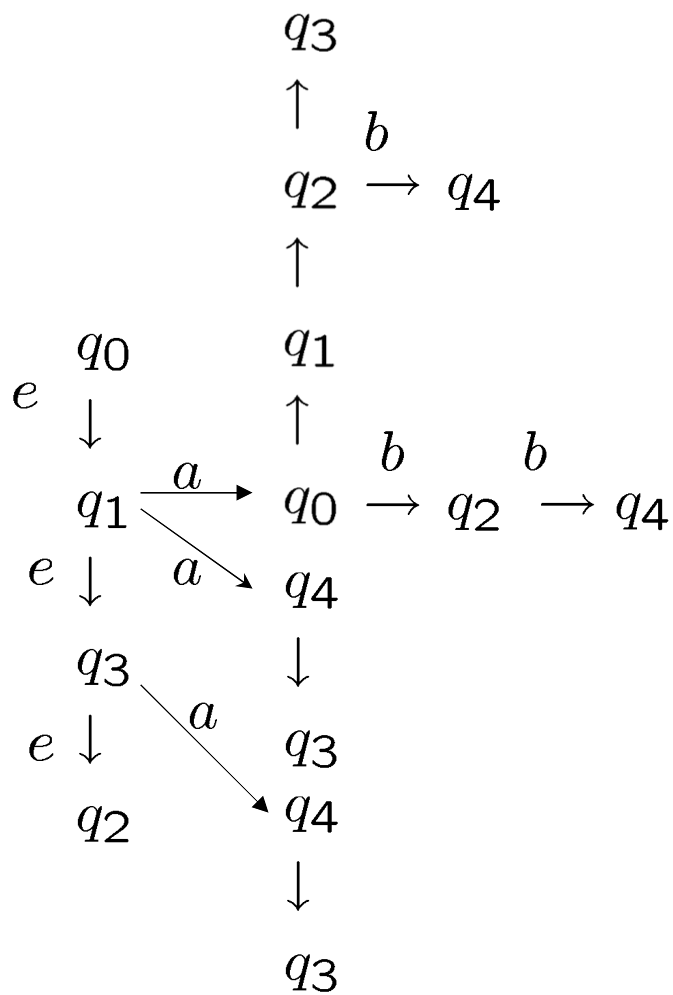
Proof:

Key idea: Every subset of K becomes a single state in our new machine!

For Example:
NFA M, $K = \{q_0, q_1, q_2, q_3, q_4\}$, after reading a certain input string, M could be in $\{q_0, q_2, q_3\}$, not in $\{q_1, q_4\}$, then DFA M' is in state $\{q_0, q_2, q_3\}$.


q_1
e
q_3
a
a
a
e
e
e
q_0
b
q_2
b
q_4

$$w = abb$$


q_3
b
q_2
q_4
q_1
q_0
e
b
b
q_1
a
q_0
q_2
q_4
e
a
q_4
q_3
a
q_3
e
q_4
q_2
q_3

□ NFA $\Rightarrow$ DFA construction

Consider an arbitrary NFA $M = (K, \sum, \Delta, s, F)$.
To construct an equivalent DFA $M' = (K', \sum, \delta, s', F')$.

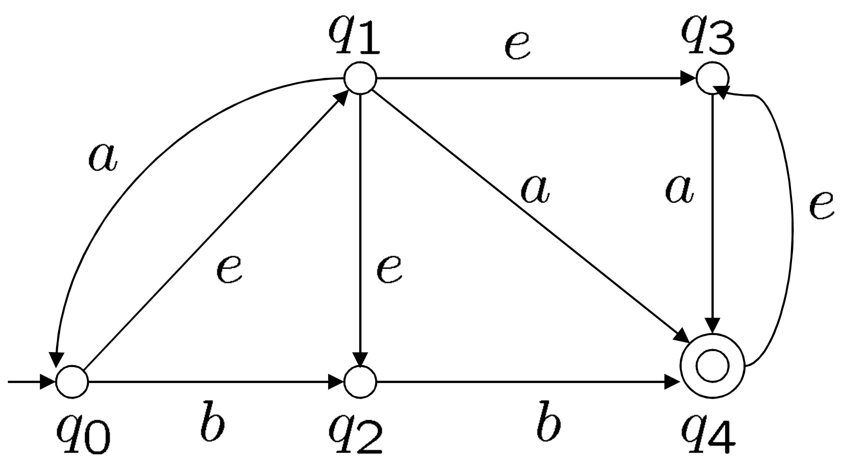
Definition:

$$E(q) = \{p \in K : (q, e) \vdash_M^* (p, e)\}$$

Remark:

$E(q)$— the set of all states of M that are reachable from state q without reading any input.

Example:



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

$$E(q_2) = \{q_2\} \qquad E(q_3) = \{q_3\}$$

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

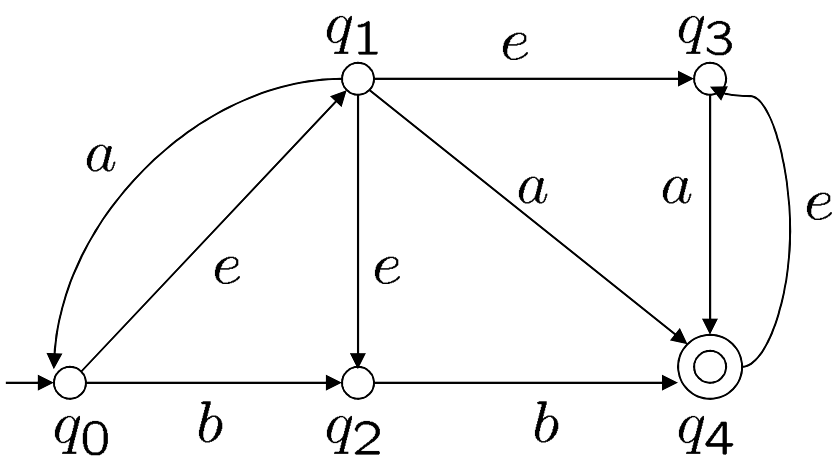
I. Formal Definition of DFA

NFA $M = (K, \sum, \Delta, s, F)$.
To construct an equivalent DFA $M' = (K', \sum, \delta, s', F')$.

- Let $K' = 2^K$
- Let $s' = E(s)$
- Let $F' = \{Q \mid Q \subseteq K, Q \cap F \neq \emptyset\}$
- For each $Q \subseteq K$ and $\forall a \in \sum$,

  Let $\delta(Q, a) = \cup\{E(p) \mid p \in K$ and $(q, a, p) \in \Delta$ for some $q \in Q\}$

Example: (continued)



$$E(q_0) = \{q_0, q_1, q_2, q_3\}$$
$$E(q_1) = \{q_1, q_2, q_3\}$$
$$E(q_2) = \{q_2\}$$
$$E(q_3) = \{q_3\}$$
$$E(q_4) = \{q_3, q_4\},$$

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

since $(q_1, a, q_4), (q_1, a, q_0) \in \Delta$

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

II. NFA $\Rightarrow$ DFA construction: verification

We next show:

- M' is deterministic
- M' is equivalent to M.

- **M' is deterministic:**

□ δ is a single-valued and well defined on all $Q \in K'$ and $a \in \sum$ by the way it is constructed.

– For each $Q \subseteq K$ and $\forall a \in \sum$,
Let $\delta(Q, a) = \cup\{E(p) \mid p \in K$ and $(q, a, p) \in \Delta$ for some $q \in Q\}$

□ For some $Q \in K'$ and $a \in \sum$, $\delta(Q, a) = \emptyset$ does not mean δ is not well defined; $\emptyset$ is a member of K'.

- **M' is equivalent to M.**

Claim: For any string $w \in \sum^*$ and any states $p, q \in K$,

$$(q, w) \vdash_M^* (p, e) \Leftrightarrow (E(q), w) \vdash_{M'}^* (P, e)$$

for some set P containing p.

Consider any string $w \in \sum^*$.

definition

claim

$$\begin{aligned} w \in L(M) &\Leftrightarrow (s, w) \vdash_M^* (f, e) \text{ for some } f \in F \\ &\Leftrightarrow (E(s), w) \vdash_{M'}^* (Q, e) \text{ for some } f \in Q \\ &\Leftrightarrow (s', w) \vdash_{M'}^* (Q, e) \text{ with } Q \in F' \\ &\Leftrightarrow w \in L(M') \end{aligned}$$

From the claim, M' is equivalent to M.

- **Proof of Claim:**

Basis step:

For $|w| = 0$, i.e. $w = e$, we must show that

$$(q, e) \vdash_M^* (p, e) \Leftrightarrow (E(q), e) \vdash_{M'}^* (P, e)$$

for some set P containing p.

– For each $Q \subseteq K$ and $\forall a \in \sum$,

Let $\delta(Q, a) = \cup\{E(p) \mid p \in K$ and $(q, a, p) \in \Delta$ for some $q \in Q\}$

$$\boxed{(q, e) \vdash_M^* (p, e) \Leftrightarrow p \in E(q)}$$

M' is deterministic

$(E(q), e) \vdash_{M'}^* (P, e) \quad \Leftrightarrow P = E(q)$

$$p \in P \Rightarrow p \in E(q)$$

Induction hypothesis:

Suppose the claim is true for all string w of length k or less for some $k \geq 0$.

Induction step:

Let $|w| = k+1$. $w = va$, where $a \in \sum, v \in \sum^*$.

$$\Rightarrow \qquad \boxed{(q, w) \vdash_M^* (p, e) \Leftrightarrow (E(q), w) \vdash_{M'}^* (P, e)}$$

$(q, w) \vdash_M^* (p, e) \Leftrightarrow \exists$ states r_1 and r_2 such that

$$(q, va) \vdash_M^* (r_1, a) \vdash_M (r_2, e) \vdash_M^* (p, e)$$

I $(q, va) \vdash_M^* (r_1, a) \Leftrightarrow (q, v) \vdash_M^* (r_1, e)$

$\Leftrightarrow (E(q), v) \vdash_{M'}^* (R_1, e)$ for some R_1,$r_1 \in R_1$

Induction hypothesis

II $(r_1, a) \vdash_M (r_2, e) \Leftrightarrow (r_1, a, r_2) \in \Delta$

$\Rightarrow E(r_2) \subseteq \delta(R_1, a)$

Definition of $\delta(R_1, a)$

– For each $Q \subseteq K$ and $\forall a \in \sum$,
Let $\delta(Q, a) = \cup\{E(p) \mid p \in K$ and $(q, a, p) \in \Delta$ for some $q \in Q\}$

$p \in \delta(R_1, a)$

III $(r_2, e) \vdash_M^* (p, e) \Rightarrow p \in E(r_2)$

$\Rightarrow (E(q), va) \vdash_{M'}^* (R_1, a) \vdash_{M'}^* (P, e).$

$P = \delta(R_1, a)$

$\Leftarrow$

- $(E(q), va) \vdash^*_{M'} (R_1, e) \vdash^*_{M'} (P, e)$ for some $p \in P$
- $(R_1, a) \vdash^*_{M'} (P, e) \Rightarrow P = \delta(R_1, a)$
- $\delta(R_1, a) = \cup\{E(r_2) : \exists r_2 \in K$ and $(r_1, a, r_2) \in \Delta$

  for some $r_1 \in R_1\}$

  $p \in P \Rightarrow \exists r_2$ such that $p \in E(r_2)$ and for smoe

  $r_1 \in R_1, (r_1, a, r_2) \in \Delta$

  $(r_2, e) \vdash^*_M (p, e)$ by the definition of $E(r_2)$

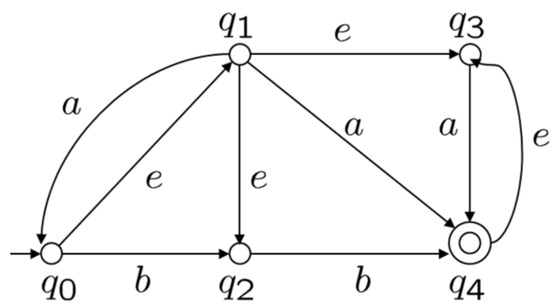
- By IH, $(q, va) \vdash^*_M (r_1, a)$

Therefore, $(q, va) \vdash^*_M (r_1, a) \vdash^*_M (r_2, e) \vdash^*_M (p, e)$.

Remark:

1) Size of new DFA can be exponential in size of old NFA!

2) The proof of theorem provides an actual algorithm for constructing an equivalent DFA from any NFA.



Example:

Convert the NFA in example(this section) to a DFA.

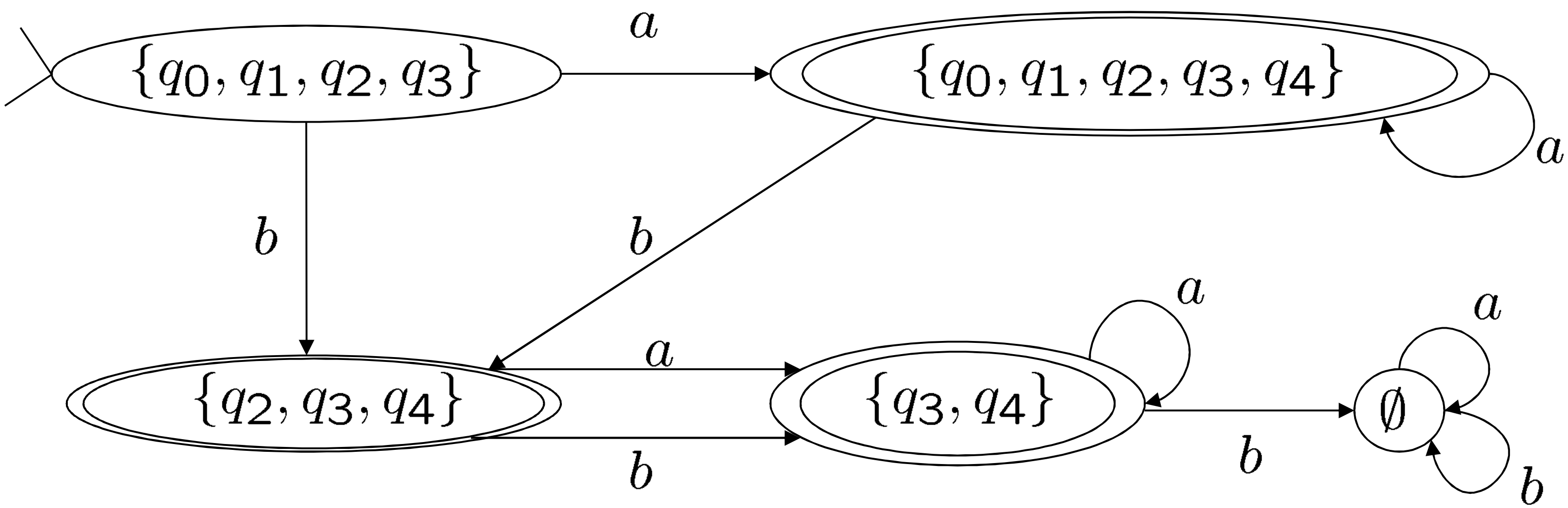
$|K| = 5 \Rightarrow |K'| = 2^5 = 32.$

— *Only a few of these states will be relevant to the operation of M.*

(q_1, a, q_0),(q_1, a, q_4),(q_3, a, q_4) are all the transitions for some $q \in s'$.

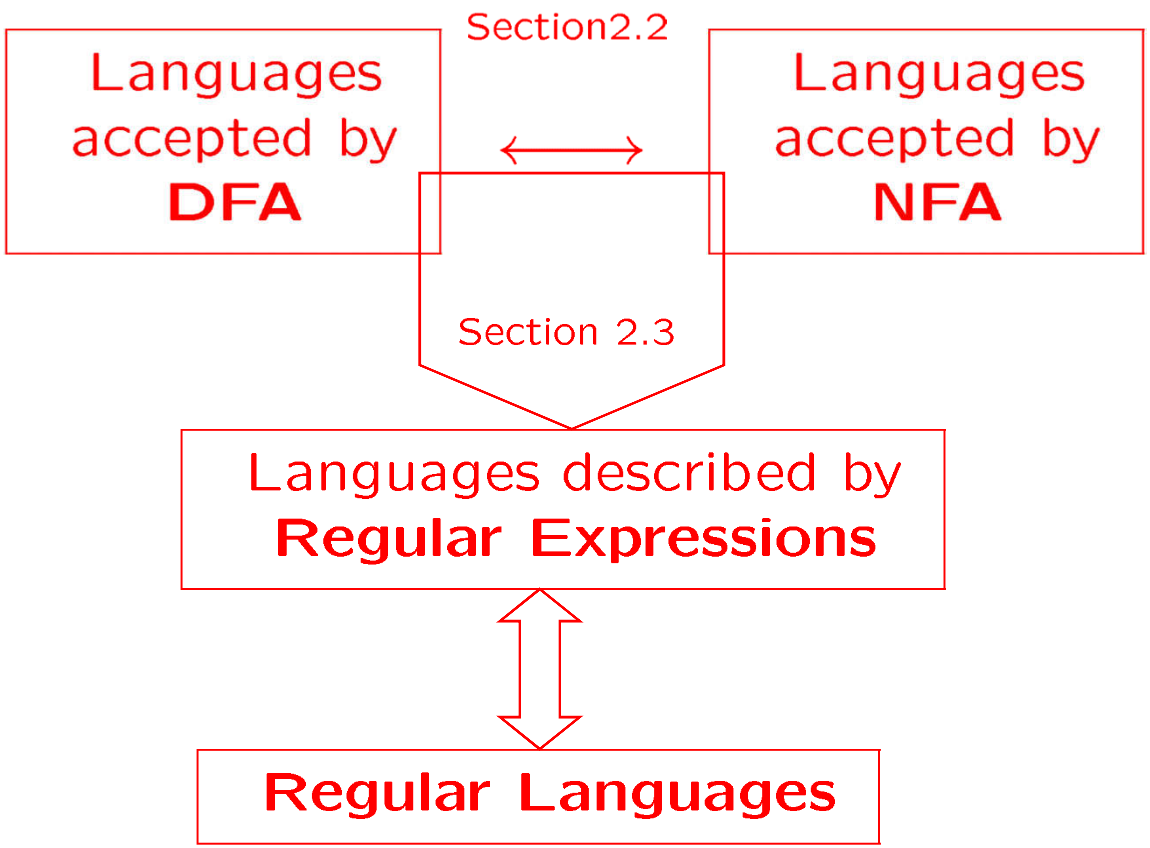
$$s' = E(q_0) = \{q_0, q_1, q_2, q_3\}$$
$$\delta(s', a) = E(q_0) \cup E(q_4) = \{q_0, q_1, q_2, q_3, q_4\}$$
$$\delta(s, b) = E(q_2) \cup E(q_4) = \{q_2, q_3, q_4\}$$

$$\delta(\{q_0, q_1, q_2, q_3, q_4\}, a) = \{q_0, q_1, q_2, q_3, q_4\}$$
$$\delta(\{q_0, q_1, q_2, q_3, q_4\}, b) = \{q_2, q_3, q_4\}$$
$$\delta(\{q_2, q_3, q_4\}, a) = \{q_3, q_4\}$$
$$\delta(\{q_2, q_3, q_4\}, b) = \{q_3, q_4\}$$
$$\delta(\{q_3, q_4\}, a) = \{q_3, q_4\},\ \delta(\{q_3, q_4\}, b) = \emptyset$$
$$\delta(\emptyset, a) = \delta(\emptyset, b) = \emptyset$$



Homework 2:	
P60	2.1.1 2.1.2 (c)(d) 2.1.3 (c)(e)
P74	2.2.2 (a)(b) 2.2.6 (a)(b) 2.2.10
P83	2.3.4 (b) 2.3.7 (a)(b)(c)(d)
P90	2.3.5 (a) 2.4.8

2.3 Finite Automata & Regular Expressions



□ Closure Properties of Regular Languages

Theorem: The class of languages accepted by FA is **closed** under

(a) Union

(b) Concatenation

(c) Kleene star

(d) Complementation

(e) Intersection

Proof:

Main idea To construct an FA that accept the appropriate language in each case.

(a) Union:

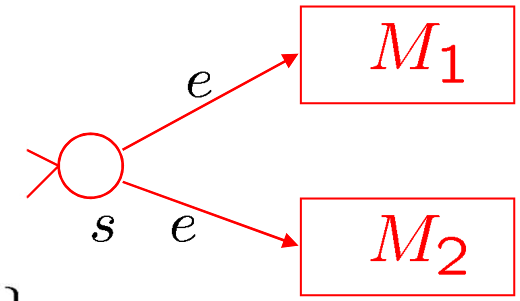
Let $M_i = (K_i, \sum, \Delta_i, s_i, F_i)$ $(i = 1, 2)$ be two NFA.

$\Rightarrow$ construct a NFA $M = (K, \sum, \Delta, s, F)$, such that

$$L(M) = L(M_1) \cup L(M_2).$$

Where

- $K = K_1 \cup K_2 \cup \{s\}$
- $F = F_1 \cup F_2$
- $\Delta = \Delta_1 \cup \Delta_2 \cup \{(s, e, s_1), (s, e, s_2)\}$



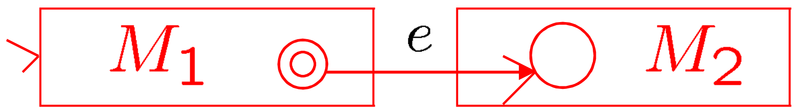
Note: Any finite union of regular sets is regular. Inifinite unions may not be regular.

(b) Concatenation:

Let $M_1 = (K_1, \sum, \Delta_1, s_1, F_1)$ and $M_2 = (K_2, \sum, \Delta_2, s_2, F_2)$ be two NFA.

$\Rightarrow$ construct a NFA $M = (K, \sum, \Delta, s, F)$, such that

$$L(M) = L(M_1) \circ L(M_2).$$



Where

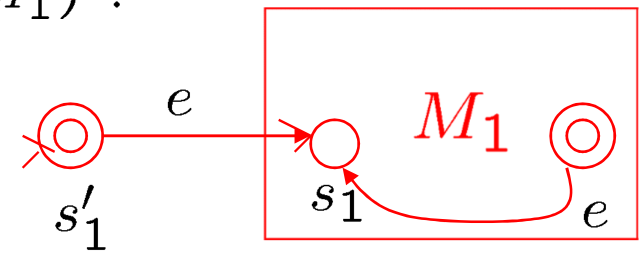
- $K = K_1 \cup K_2$
- $F = F_2$
- $\Delta = \Delta_1 \cup \Delta_2 \cup \{(q_i, e, s_2) : q_i \in F_1\}$

(c) Kleene star:

Let $M_1 = (K_1, \sum, \Delta_1, s_1, F_1)$ be a NFA.

$\Rightarrow$ construct a NFA $M = (K, \sum, \Delta, s, F)$, such that

$$L(M) = L(M_1)^*.$$



Where

- $K = K_1 \cup \{s_1'\}$
- $F = F_1 \cup \{s_1'\}$
- $\Delta = \Delta_1 \cup \{(q_i, e, s_1) : q_i \in F_1\} \cup \{(s_1', e, s_1)\}$

(d) Complementation:

Let $M_1 = (K_1, \sum, \delta_1, s_1, F_1)$ be a DFA.

$\Rightarrow$ construct a DFA $M = (K, \sum, \delta, s, F)$, such that

$$L(M) = \overline{L(M_1)} = \sum^* - L(M_1).$$

Where

- $K = K_1$
- $s = s_1$
- $F = K - F_1$
- $\delta = \delta_1$

(e) Intersection:

$$\boxed{L(M_1) \cap L(M_2) = \sum^* - (\sum^* - L(M_1)) \cup (\sum^* - L(M_2))}$$

Let $M_i = (K_i, \sum, \delta_i, s_i, F_i)$ $(i = 1, 2)$ be two DFA.

$\Rightarrow$ construct a DFA $M = (K, \sum, \delta, s, F)$, such that

$$L(M) = L(M_1) \cap L(M_2).$$

Where

- $K = K_1 \times K_2$
- $s = (s_1, s_2)$
- $F = F_1 \times F_2$
- $\delta : K \times \sum \rightarrow K$

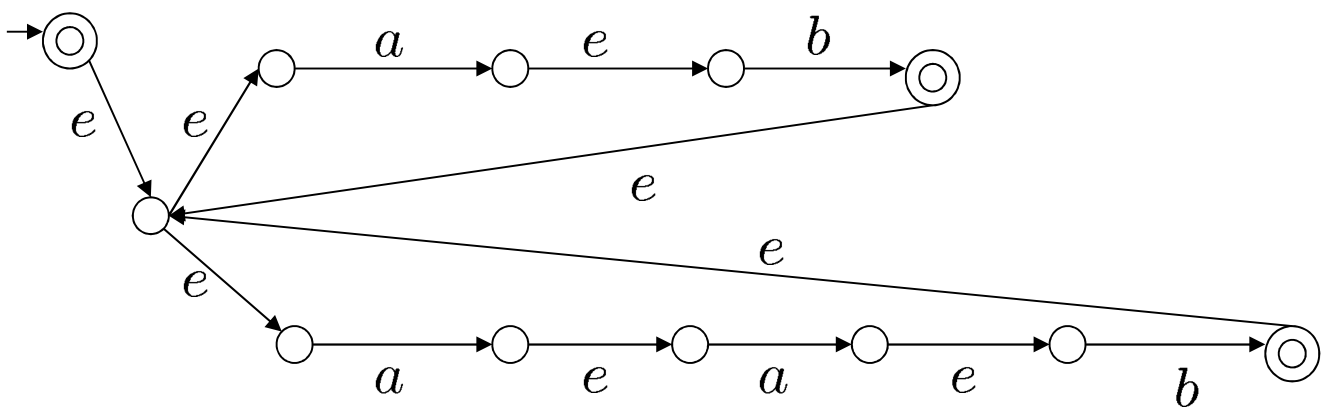
$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$

"parallel" simulation!

How about $L(M_1) \cup L(M_2)$?

Example:

Consider the regular expression $(ab \cup aab)^*$. A NFA accepting the language denoted by this regular expression can be shown as following:



□ FA and Regular Express are equivalent

Theorem: A language is regular iff it is accepted by a FA.

Proof: $\Rightarrow$

The class of regular languages is the smallest class of languages containing $\emptyset$ and the singletons $\{a|a \in \sum\}$, and closed under **union**, **concatenation**, and **Kleene star**.

$\emptyset$ and the singletons $\{a|a \in \sum\}$ are accepted by FA.

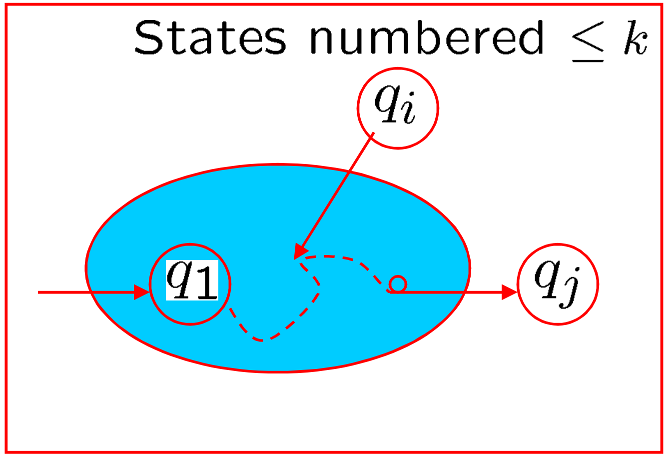
$\Rightarrow$ every regular language is accepted by some FA.

$\Leftarrow$

Let $M = (K, \sum, \Delta, s, F)$ be a FA.
To construct a regular expression R, s. t. $L(M) = L(R)$.

Let $K = \{q_1, \cdots, q_n\}$ and $s = q_1$.

We'll do induction over k for executions that use only states in $\{q_1, \cdots, q_k\}$



Definition: For $i, j = 1, \cdots, n$ and $k = 0, \cdots, n$, define $R(i,j,k) = \{w \mid w \in \sum^*, \delta(q_i, w) = q_j$ and for any prefix x of w, $x \neq e$, $\delta(q_i, x) = q_l \wedge (l \leq k)\}$

$$R(i,j,0) = \begin{cases} \{a \mid \delta(q_i, a) = q_j\} & \text{if } i \neq j \\ \{a \mid \delta(q_i, a) = q_j\} \cup \{e\} & \text{if } i = j \end{cases}$$

Note:

$R(i,j,n) = \{\ w | w \in \sum^*, (q_i, w) \vdash_M^* (q_j, e)\ \}$

$L(M) = \cup\{\ R(1,j,n) | q_j \in F\ \}$

Theorem: $R(i,j,k)$ are regular languages.

Proof: By induction on k.

Basis step: $k = 0$

$$R(i,j,0) = \begin{cases} \{a|\ \delta(q_i,a) = q_j\} & \text{if } i \neq j \\ \{a|\ \delta(q_i,a) = q_j\} \cup \{e\} & \text{if } i = j \end{cases}$$

Induction step:

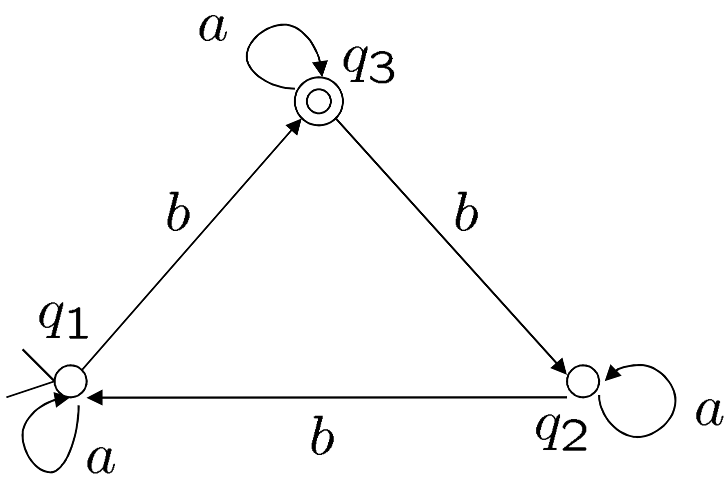
$$R(i,j,k) = R(i,j,k-1) \cup R(i,k,k-1)R(k,k,k-1)^* R(k,j,k-1)$$

$\Rightarrow R(i,j,k)$ is regular for all i,j,k.

Example: Construct a regular expression for the language

$$L = \{w \in \{a,b\}^* : w \text{ has } 3k+1\ b'\text{s for some } k \in \mathbb{N}\}$$

accepted by the following DFA.



Assume that the DFA M has two simple properties:

1) It has a single final state, $F = \{f\}$.

2) If $(q, u, p) \in \Delta$, then $q \neq f$ and $p \neq s$.

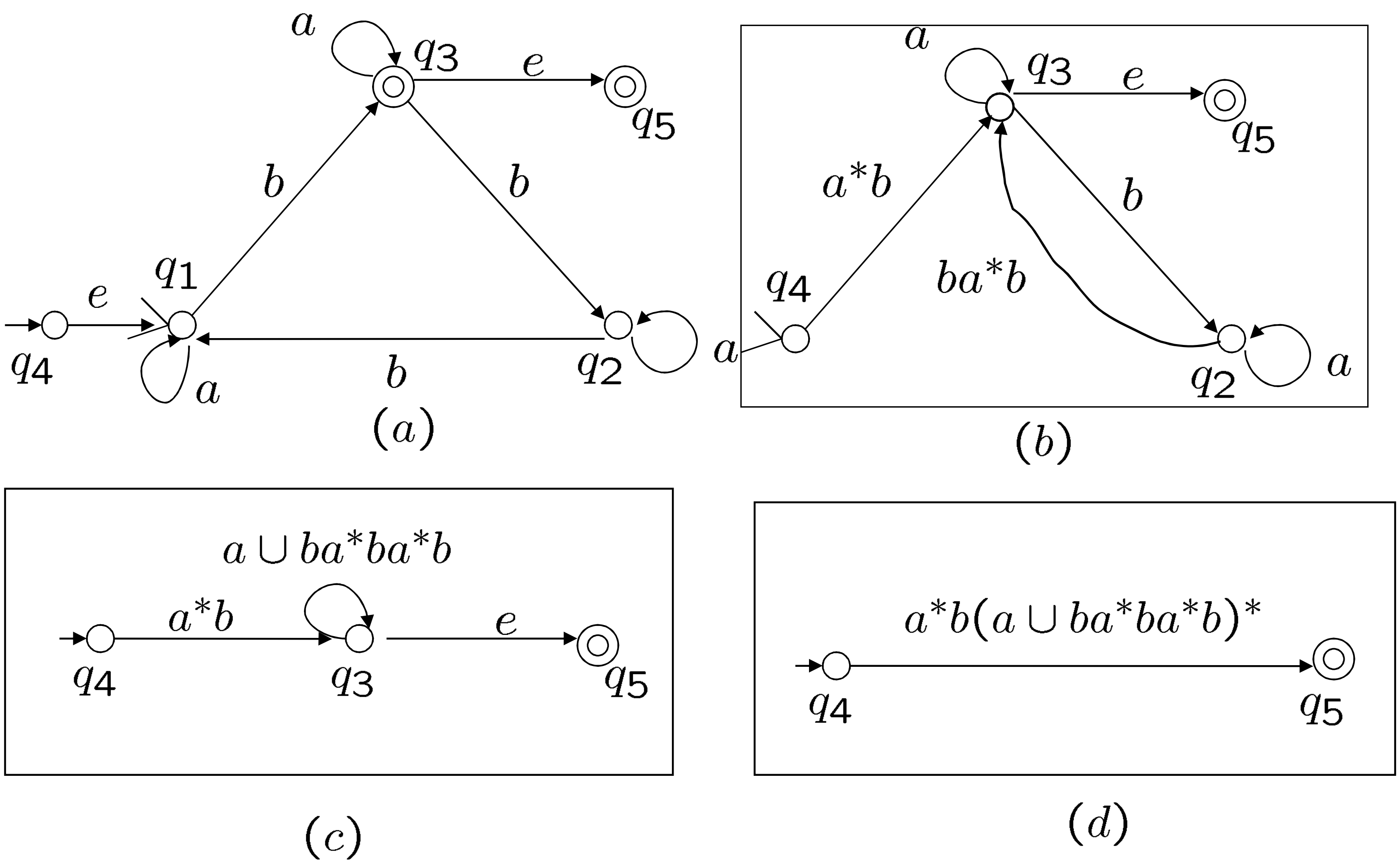
Any DFA $\Rightarrow$ An equivalent FA with properties 1)2).

Let $K = \{q_1, q_2, \cdots, q_n\}$, so that $s = q_{n-1}, f = q_n$.

$\Rightarrow$ The regular expression $R(n-1, n, n) = L(M)$.

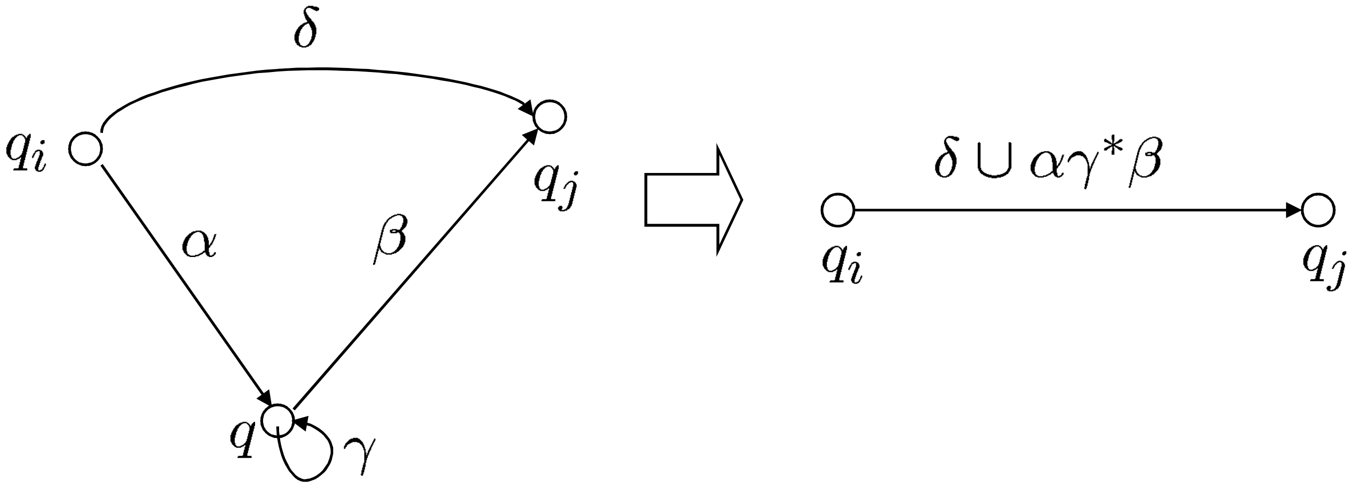
We compute

$$R(i, j, 0) \rightarrow R(i, j, 1) \rightarrow \cdots \rightarrow R(i, j, n)$$


a
q3
e
q5
b
b
q1
e
q4
a
b
q2
(a)
a
q3
e
q5
a*b
b
q4
ba*b
a
q2
a
(b)
a ∪ ba*ba*b
a*b
e
q4
q3
q5
(c)
a*b(a ∪ ba*ba*b)*
q4
q5
(d)

Remark:

In general, for each pair of states $q_i \neq q$ and $q_j \neq q$, eliminate state q as following:



2.4 Languages that are and are not Regular

To show a language be **Regular**

- Accepted by **FA**
- Specified by **Regular Expressions**
- Closure properties of **Regular Languages**

Example: Let $\sum = \{0, 1, \cdots, 9\}$.
$L = \{w : w \in \sum^*$ be the decimal representation for non-negative integers(without redundant leading $0'$s) divisible by 2 and 3$\}$. Then L is regular.

Solution:

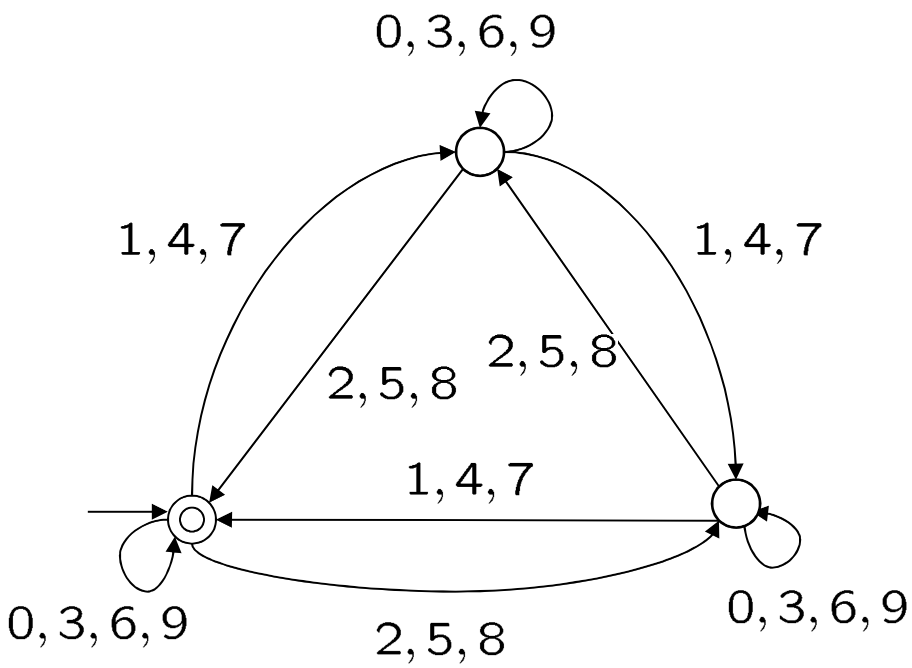
- Let L_1 be the set of decimal representation of nonnegative integers.

$$L_1 = 0 \cup \{1, 2, \cdots, 9\} \sum^* .$$

- Let L_2 be the set of decimal representation of nonnegative integers divisible by 2.

$$L_2 = L_1 \cap \sum^* \{0, 2, 4, 6, 8\}.$$

- Let L_3 be the set of decimal representation of nonnegative integers divisible by 3. L_3 can be accepted by the following FA.



$$L = L_2 \cap L_3$$

– L is regualar.

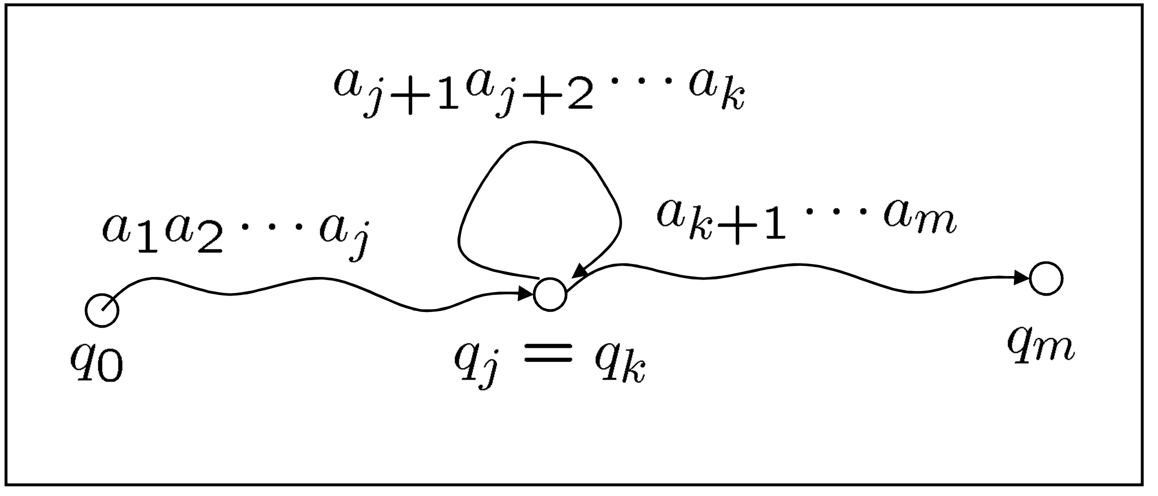
□ What languages are not regular?

Intuition:

Since a FA has only finite state, it can "remember" only a finite number of things

- Some things we would expect are not regular
  - *Balanced parentheses (have to remember an arbitrary nesting depth)*
  - $\{a^n b^n : n \geq 0\}$

If a FA accepts a string that is "long enough", it must repeat a state



– $\{a^n : n \geq 1$ *is a prime*$\}$ *is not regular.*

Theorem: (Pumping Theorem) Let L be a regular language. $\exists$ an integer $n \geq 1$ such that any string $w \in L$ with $|w| \geq n$ can be written as $w = xyz$ such that

- $y \neq e$
- $|xy| \leq n$
- for each $i \geq 0$ $xy^iz \in L$

Proof:

Let L be a regular language, accepted by a DFA M with n states, and string w, $|w| \geq n$.

Consider the first n steps of computation of M:

$$(q_0, a_1 \cdots a_n) \vdash_M (q_1, a_2 \cdots a_n) \vdash_M \cdots \vdash_M (q_n, e)$$

Pigeonhole Principle

$\Rightarrow \exists i, j$, $0 \leq i < j \leq n$, such that $q_i = q_j$

Let $y = a_i \cdots a_j, x = a_1 \cdots a_{i-1}, z = a_{j+1} \cdots a_m$.

Example: Show $L = \{a^i b^i | i \geq 0\}$ is not regular.

Proof:

Assume L regular. By Pumping Theorem, $\exists$ integer n.

Consider the string $w = a^n b^n \in L$.

It can be written as $w = xyz$ such that

- $|xy| \leq n$
- $y \neq e$

$\Rightarrow y = a^i$ for some $i > 0$

But $xz = a^{n-i} b^n \notin L$

contradiction!!

Remark: Proving that a language is not regular:

- Let L be the proposed regular language
- There is some n, by the pumping lemma
- Choose a string s, longer than n symbols, in the language L
- Using the pumping lemma, construct a new string s' that is not in the language

Example: Show $L = \{a^n | n \text{ is prime }\}$ is not regular.

Proof:

Assume L regular. $\exists w = xyz$, and $x = a^p$, $y = a^q$ and $z = a^r$, where $p, r \geq 0$ and $q > 0$.

By Pumping theorem, $xy^nz \in L$ for each $n \geq 0$; that is, $p + nq + r$ is prime for each $n \geq 0$.

But it is impossible; for let $n = p + 2q + r + 2$; then $p + nq + r = (q + 1) \cdot (p + 2q + r)$.

contradiction!!

Example: Show

$L = \{w \in \{a, b\}^* : w$ has an equal number of a's and b's$\}$

is not regular.

Proof:

If L is regular, then so would be $L \cap a^*b^*$.

Closure under intersection of regular language

But $L \cap a^*b^* = \{a^n b^n : n \geq 0\}$

— not regular language.

L is not regular.

□ Reprise on FA and Regular Languages

Which of the following are necessarily regular?

- A finite language
- A union of a finite number of regular languages
- A union of a countable number of regular languages
- An intersection of a countable number of regular languages
- $\{x : x \in L_1 \text{ and } x \notin L_2\}$, L_1 and L_2 are both regular
- A subset of a regular language

$$\boxed{\bigcup_{i=0}^{\infty} L_i = \overline{\bigcap_{i=0}^{\infty} \overline{L_i}}}$$

Homework 2:	
P60	2.1.1 2.1.2 (c)(d) 2.1.3 (c)(e)
P74	2.2.2 (a)(b) 2.2.6 (a)(b) 2.2.10
P83	2.3.4 (b) 2.3.7 (a)(b)(c)(d)
P90	2.3.5 (a) 2.4.8