

3. Regular Languages

The family of regular languages

all finite language

closed under union, concatenation and closure.

*Three equivalent **descriptions** on regular languages*

regular expression

finite automata

regular grammar

Section 3.1

regular expression

regular language

Section 3.2

finite automata

regular expression \leftrightarrow finite automata

Section 3.3

regular grammar

finite automata \leftrightarrow regular grammar

Section 3.4

deterministic finite automata \subseteq finite automata

finite automata \rightarrow deterministic finite automata

Section 3.5

some decision problems on regular languages

Section 3.6

lexical analysis

3.1 Regular Expression

Language description

any finite means of specifying languages

finite number of finitely structured elements

writing down all the sentences **finite language**

rewriting system

infinite language

$L \subseteq V^*$ *languages over V*

countably infinite $\{0, 1\}^*$

$\mathcal{L} \subseteq 2^{V^*}$ *family of languages over V*

uncountably infinite $\{0, 1\}^{\mathbb{N}}$

no description in general

regular expression over an alphabet V

well-formed expression

arguments $\in V \cup \{\underline{\epsilon}, \underline{\emptyset}\}$

operators $*$ *closure*

\cdot *concatenation*

\cup *union*

Let E be a string over $V \cup \{\underline{\epsilon}, \underline{\emptyset}, *, \cdot, \cup, \cdot, (\}\}$

(1) E is a **regular primary** over V , if

symbol in $V \cup \{\underline{\epsilon}, \underline{\emptyset}\}$; or

(E_1) E_1 is a regular expression over V .

(2) E is a **regular factor** over V , if

regular primary over V ; or

E_1^* E_1 is a regular factor over V .

(3) E is a **regular term** over V , if

regular factor over V ; or

$E_1 \cdot E_2$ E_1 is a regular term

and E_2 is a regular factors over V .

(4) E is a **regular expression** over V , if

regular term over V ; or

$E_1 \cup E_2$ E_1 is a regular expression

and E_2 is a regular term over V .

Context-free grammar for syntax of regular exp. $\langle E \rangle$

$\langle P \rangle \rightarrow \underline{\emptyset} \mid \underline{\epsilon} \mid \langle V \rangle \mid (\langle E \rangle)$

$\langle F \rangle \rightarrow \langle P \rangle \mid \langle F \rangle^*$

$\langle T \rangle \rightarrow \langle F \rangle \mid \langle T \rangle \cdot \langle F \rangle$

$\langle E \rangle \rightarrow \langle T \rangle \mid \langle E \rangle \cup \langle T \rangle$

Ambiguous context-free grammar for $\langle E \rangle$

$\langle E \rangle \rightarrow \underline{\emptyset} \mid \underline{\epsilon} \mid \langle V \rangle \mid (\langle E \rangle) \mid$

$\langle E \rangle \cup \langle E \rangle \mid \langle E \rangle \cdot \langle E \rangle \mid \langle E \rangle^*$

with $\text{prec}(\cdot) > \text{prec}(\cup) > \text{prec}(\cdot)$ and

$\text{rightassoc}(\cdot)$, $\text{leftassoc}(\cdot)$, $\text{leftassoc}(\cup)$.

$L(E)$ language denoted by a regular expression E .

$$(1a) L(\underline{\emptyset}) = \emptyset.$$

$$(1b) L(\underline{\varepsilon}) = \{\varepsilon\}.$$

$$(1c) L(a) = \{a\} \quad \forall a \in V.$$

$$(1d) L((E)) = L(E) \quad \forall \text{ regular expression } E \text{ over } V.$$

$$(2) L(E^*) = (L(E))^* \quad \forall \text{ regular factor } E \text{ over } V.$$

$$(3) L(E_1 \cdot E_2) = L(E_1) \cdot L(E_2) \quad \forall \text{ r.t. } E_1 \text{ and r.f. } E_2 \text{ over } V$$

$$(4) L(E_1 \cup E_2) = L(E_1) \cup L(E_2) \quad \forall \text{ r.e. } E_1 \text{ and r.t. } E_2 \dots$$

regular language L over V

$L = L(E)$ for some **regular expression** E over V

Fact 3.2 Any finite language is **regular**.

A language family \mathbb{F} is **effectively closed** under n -ary operation f , if any n -tuple D_1, \dots, D_n of language **descriptions**, can be **transformed** into a **description** of the language $f(L(D_1), \dots, L(D_n))$.

A language family \mathbb{F} is **closed** under f if

$$L_1, \dots, L_n \in \mathbb{F} \text{ implies } f(L_1, \dots, L_n) \in \mathbb{F}.$$

closed vs effectively closed

Lemma 3.3 Let E be a regular expression over V . E can be transformed in time $O(|E|)$ into a regular expression that denotes $L(E)^*$.

Any pair of regular expressions E_1 and E_2 can be transformed in time $O(|E_1| + |E_2|)$ into regular expressions that denotes $L(E_1)L(E_2)$ and $L(E_1) \cup L(E_2)$.

$(E)^*$, $(E_1)(E_2)$, $(E_1) \cup (E_2)$.

Theorem 3.4 (Kleene) For any alphabet V , the **family of regular languages over V** is the **smallest** family of languages over V that contains all finite language over V and is (**effectively**) **closed** under closure, concatenation, and finite union.

Two language descriptions D_1 and D_2 are **equivalent**, if $L(D_1) = L(D_2)$; **inequivalent**, otherwise.

Fact 3.5 For any regular expression there exists a **countably infinite** number of equivalent regular expressions.

Proof. $E, E \cup E, E \cup E \cup E, \dots$

Given any class of language description \mathbb{D} , any description in \mathbb{D} usually has a **countably infinite** number of equivalent descriptions.

(renaming of symbols)

A language description D is **ambiguous** if some sentence in $L(D)$ is **described in two ways**. A description D is **unambiguous** if it is not ambiguous.

- (1) \emptyset , ε , a and (E) are **unambiguous**, if
 $\forall a \in V$, E is unambiguous, respectively.
- (2) E^* is **unambiguous** if E is unambiguous and
 $\forall x \in L(E^*) \exists$ one $n \geq 0$, \exists one sequence (x_1, \dots, x_n)
 $\cdot \exists. x = x_1 \dots x_n, 1 \leq i \leq n, x_i \in L(E)$.
- (3) E_1E_2 is **unambiguous**, if
 (a) $L(E_1E_2) = \emptyset$; or
 (b) if E_1, E_2 are unambiguous, and
 $\forall x \in L(E_1E_2), \exists$ one $(y, z) \in L(E_1) \times L(E_2)$
 $\cdot \exists. yz = x$.
- (3) $E_1 \cup E_2$ is **unambiguous**, if
 E_1, E_2 are unambiguous, $L(E_1) \cap L(E_2) = \emptyset$.

ambiguity in language description
 it may have two different **meanings**
enhanced description power

$$(0 \cup 1)^*(000 \cup 111)(0 \cup 1)^*$$

unambiguity of language description
 may reduce the descriptive power

*description power of unambiguous regular language
same as unrestricted one*

No “inherently ambiguous” regular languages

Theorem 3.6 Any regular expression can be transformed into unambiguous one.

Proof

regular expression \Rightarrow deterministic automaton

unambiguous automaton \Rightarrow unambiguous r.e.

deterministic automaton \subseteq unambiguous automaton

The descriptions in \mathbb{D}_1 are at least as descriptive as those in \mathbb{D}_2 , if

$$\forall D_2 \in \mathbb{D}_2, \exists D_1 \in \mathbb{D}_1, L(D_1) = L(D_2). \\ (L(\mathbb{D}_1) \supseteq L(\mathbb{D}_2))$$

The description in \mathbb{D}_1 are more descriptive as those in \mathbb{D}_2 , if the descriptions in \mathbb{D}_1 are at least as descriptive as those in \mathbb{D}_2 but

$$\exists D_1 \in \mathbb{D}_1, \exists D_2 \in \mathbb{D}_2, L(D_1) = L(D_2). \\ (L(\mathbb{D}_1) \supset L(\mathbb{D}_2))$$

The descriptions in \mathbb{D}_1 are as descriptive as those in \mathbb{D}_2 , if the descriptions in \mathbb{D}_1 are at least as descriptive as those \mathbb{D}_2 and vice versa.

$$(L(\mathbb{D}_1) = L(\mathbb{D}_2)).$$

The descriptions in \mathbb{D}_1 are **at least as succinct as those** in \mathbb{D}_2 , if

$$\forall D_2 \in \mathbb{D}_2, \exists D_1 \in \mathbb{D}_1, L(D_1) = L(D_2),$$

the size of D_1 is at most linear in the size of D_2 .

The descriptions in \mathbb{D}_1 are **as succinct as (or equivalent in succinctness)** those in \mathbb{D}_2 , if ...

Let f be a function: natural numbers \rightarrow positive reals
 The descriptions in \mathbb{D}_1 can be **$f(n)$ more succinct than** those in \mathbb{D}_2 , if $\exists L_1, \dots, L_n, \dots$ each L_n has description in \mathbb{D}_1 , size of L_n is $O(n)$ but the equivalent description in \mathbb{D}_2 are of size at least $f(n)$.

Proposition 3.7 There exist a constant $c > 0$ and an infinite sequence of regular languages L_1, L_2, \dots over $\{0, 1\}$ such that each L_n is denoted by an **ambiguous regular expression** of length $O(n)$ but any **unambiguous regular expression** denoting L_n must have length at least $2^{c \cdot n}$.

(Ambiguous) regular expressions are
 2^n **more succinct than**
unambiguous regular expressions.

3.2 Finite Automata

Let $M = (V, P)$ be a rewriting system.

$$V = Q \cup \Sigma, Q \cap \Sigma = \emptyset, q_s \in Q, \text{ and } F \subseteq Q.$$

$M = (Q, \Sigma, P, q_s, F)$ is a **finite automaton** with

state alphabet(set) Q ,

input alphabet(set) Σ ,

initial state q_s ,

set of final states F ,

set of rules P ,

$qx \rightarrow p \in P$, if $q, p \in Q, x \in \Sigma^*$, *or* $p \in \delta(q, x)$.
x transition(transition on x) from state q to p.

configuration(instantaneous description) of M

$$qw \quad qw \in Q\Sigma^*. \quad \text{or} \quad (q, w) \in Q \times \Sigma^*.$$

$$qw \quad \text{initial for } w \in \Sigma^*, \text{ if } q = q_s. \quad (q_s, w)$$

$$qw \quad \text{accepting, if } w = \varepsilon, q \in F. \quad (q, \varepsilon)$$

$$qw \quad \text{error, if nonaccepting,}$$

$$0 \leq \forall k \leq |w|, qk:w \rightarrow p \notin P.$$

computation(process) of M on input string w

derivation in M from initial configuration $q_s w$

accepting computation,

if it end with accepting configuration

M accepts w, if it has an accepting computation.

language accepted (recognized or described) by M

$$L(M) = \{w \in \Sigma^* \mid q_s w \xRightarrow{*} q \text{ in } M, q \in F\}.$$

transition graph

<i>nodes</i>	Q
<i>edges</i> (q, p)	$qx \rightarrow p \in P$
<i>labeled by x</i>	

A finite automaton is **ambiguous**,
 if it accepts some sentence in two distinct ways
 \exists at least two accepting computation,
 it is **unambiguous**, otherwise.

A state p is **reachable** from a state q upon reading w
 $qw \xRightarrow{*} p$.

A state p is **accessible** upon reading w ,
 $q_s w \xRightarrow{*} p$.

Fact 3.8 A finite automaton accepts w , iff some final
 state is accessible upon reading w .

accessible state

A state that is accessible upon reading some string
inaccessible state, if it is not accessible.

A fa with no inaccessible state is called **reduced**.

Lemma 3.9 *The set of states **reachable** from a given state of a finite automaton M can be computed in time $O(|M|)$.*

Proof.

q_1 **reaches** q_2 , if M has a transition from q_1 to q_2 ,
in $O(|M|)$

q_1 **reaches**^{*} q_2 , in $O(|M|)$ by theorem 2.2

Theorem 3.10 *Any finite automaton $M(Q, \Sigma, P, q_s, F)$ can be transformed in time $O(|M|)$ into an **equivalent reduced** finite automaton $M(Q', \Sigma, P', q_s, F')$, where $Q' \subseteq Q$, $P' \subseteq P$, and $F' \subseteq F$.*

A state q is **live**, if some final state is reachable from it. A state that is not live is **dead**.

Lemma 3.11 *The set of **live** states of a finite automaton M can be computed in time $O(|M|)$.*

Proof. (set of live states) $(\text{reaches}^{-1})^* F$.

Fact 3.12 *Let M be an **unambiguous** finite automaton, Then for any accessible live state q_1 and q_2 , and any input string x , there is **at most one derivation** of q_2 from q_1x in M .*

A finite automaton is **normal-form**, if

$$q_1 t \rightarrow q_2 \in P, t \in \Sigma \cup \{\varepsilon\}.$$

A finite automaton is **ε -free**, if no ε -transition.

Normal-form finite automaton are equivalent in description power as well as **succinctness** to unrestricted finite automaton.

Theorem 3.13 Any fa $M = (Q, \Sigma, P, q_s, F)$ can be transformed in time $O(|M|)$ into an equivalent **normal-form** fa $M' = (Q', \Sigma, P', q_s', F')$. Moreover M' is **ambiguous** iff M is; and M' is **ε -free** iff M is.

Proof.

$$Q' = \{[q] \mid q \in Q\} \cup \{[qx] \mid qxy \rightarrow p \in P, y \in \Sigma^+\}$$

$$q_s' = [q_s],$$

$$F' = \{[q] \mid q \in F\}.$$

$$P' = \{[q] \rightarrow [p] \mid q \rightarrow p \in P\}$$

$$\cup \{[qx]a \rightarrow [qxa] \mid a \in \Sigma, q xay \rightarrow p \in P, y \in \Sigma^+\}$$

$$\cup \{[qx]a \rightarrow [p] \mid a \in \Sigma, qxa \rightarrow p \in P\}$$

i) $q \rightarrow p \in P$

$$[qw] \Rightarrow_{M'} [pw].$$

ii) $qa_1 \dots a_n \rightarrow p \in P, n \geq 1$

$$[q]a_1 \dots a_n w \Rightarrow_{M'} [qa_1]a_2 \dots a_n w \Rightarrow_{M'} \dots \Rightarrow_{M'}$$

$$[qa_1 \dots a_{n-1}]a_n w \Rightarrow_{M'} [p]w \text{ in } M'$$

$$\therefore qx \rightarrow p \in P, \text{ iff } [q]xw \Rightarrow^* [p]w \text{ in } M'.$$

If $qa_1 \dots a_n \rightarrow p \in P, n \geq 1$

$[q]a_1 \rightarrow [qa_1],$

$[qa_1]a_2 \rightarrow [qa_1a_2],$

...

$[qa_1 \dots a_{n-2}]a_{n-1} \rightarrow [qa_1 \dots a_{n-1}],$

$[qa_1 \dots a_{n-1}]a_n \rightarrow p \in P'$

$|P|: (n+2) \Rightarrow |P'|: 3n$

Lemma 3.14 Any normal-form fa $M = (Q, \Sigma, P, q_s, F)$ can be transformed in time $O(|Q| \cdot |M|)$ into an equivalent ε -free fa $M'' = (Q, \Sigma, P'', q_s, F'')$. Moreover if M is unambiguous, then so is M'' .

Proof

$P'' = \{qa \rightarrow p \mid \exists q'' . \exists. q \xrightarrow{*} q'', q''a \rightarrow p \in P\}.$

$F'' = \{q \mid q \xrightarrow{*} q'' \in F\}$

$\therefore qw \xrightarrow{*} p$ in M , iff $qw \xrightarrow{*} q''$ and $q'' \xrightarrow{*} p$ in M .

M'' may remove some ambiguities on sequences of ε -moves.

q empty-trans p , iff $q \rightarrow p \in P$ of size $O(|M|)$.

q empty-trans* p , iff $q \xrightarrow{*} p \in P$

in time $O(|Q| \cdot |M|)$ Theorem 2.3.

Theorem 3.15 Any fa M can be transformed in time $O(|M|^2)$ into an equivalent ε -free normal-form fa. Moreover if M is unambiguous, so is transformed automaton.

Nonlinear time bound in theorem 3.15

fa with ε -transition is $O(|M|^2)$ more **succinct** than ε -free counterpart

L_1, L_2, \dots each L_n non- ε -free n.f. fa $O(n)$
 ε -free normal-form fa at least $3n(n+1)/2$.

finite automaton

\Downarrow $O(n)$ in succinctness

reduced fa

\Downarrow $O(n)$ in succinctness

normal form fa

\Downarrow $O(n^2)$ in succinctness

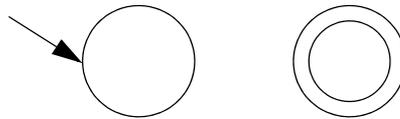
ε -free fa

ε -free normal-form fa at least $3n(n+1)/2$.

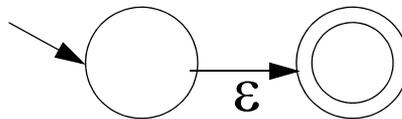
Exercises 3.7

Theorem 3.16 Any regular expression E over Σ can be transformed in time $O(|E|)$ into an equivalent finite automaton $M(E)$ with input alphabet Σ . Moreover $M(E)$ is unambiguous iff E is.

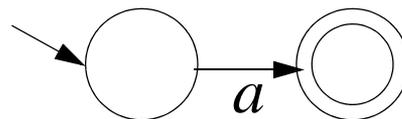
(1a) $M(\emptyset)$



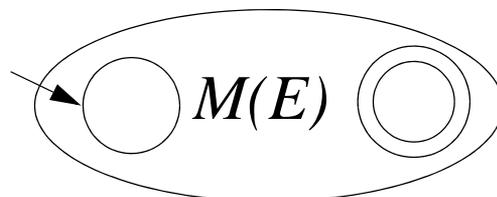
(1b) $M(\varepsilon)$



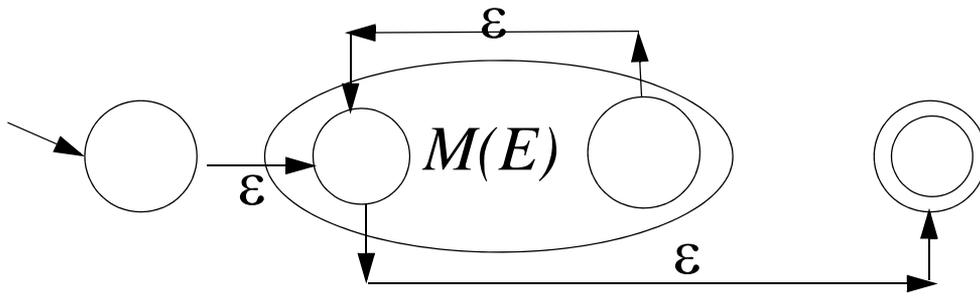
(1c) $M(a)$



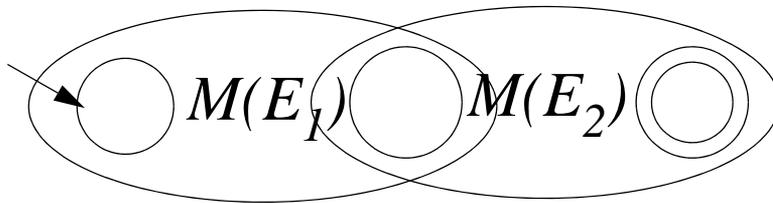
(1d) $M((E))$



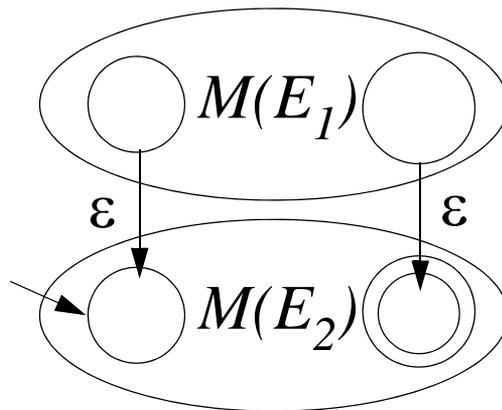
(2) $M(E^*)$



(3) $M(E_1E_2)$



(4) $M(E_1 \cup E_2)$



$L_{match} = \{0^n 1^n \mid n \geq 0\}$ is not regular.

Proof pumping

Assume L_{match} is regular.

ε -free and normal-form automaton M .

assume $n = |Q| + 1$

$$q_s 0^n 1^n \xrightarrow{n} q 1^n \xrightarrow{n} q_f \in F$$

Since $n > |Q|$, $\exists i \geq 0$, $k > 0$, and $\exists p \in Q$.

$$q_s 0^n 1^n \xrightarrow{i} p 0^{n-i} 1^n \xrightarrow{k} p 0^{n-i-k} 1^n \xrightarrow{i-k} q 1^n \xrightarrow{n} q_f \in F..$$

$$\therefore q_s 0^i 0^{j \cdot k} 0^{n-i-k} 1^n (= 0^{n-k+j \cdot k}) \xrightarrow{*} q_f \in F, \forall j \geq 0.$$

but $0^i 0^{j \cdot k} 0^{n-i-k} 1^n \notin L_{match}$.

Theorem 3.17 Any fa $M = (Q, \Sigma, P, q_s, F)$ can be transformed in time $O(|Q| \cdot |M| \cdot 4^{|Q|})$ into an equivalent **regular expression** $E(M)$ over Σ . Moreover M is **ambiguous** iff M is.

Proof Let $Q = \{q_1, \dots, q_n\}$, for $1 \leq i, j \leq n$, $0 \leq k \leq n$

E_{ij}^k a regular expression, $x \in L(E_{ij}^k)$

q_j is reachable^(one or more) from q_i upon reading x without going through any state q_m . \exists . $m > k$.

$$L(E_{ij}^k) = \{x_0 \in \Sigma^* \mid q_{s_0} x_0 \Rightarrow q_{s_1} x_1 \dots \Rightarrow q_{s_m} x_m, m \geq 1,$$

$$q_{s_0} = q_i, q_{s_m} = q_j, x_m = \varepsilon, s_l \leq k, 1 \leq \forall l \leq m-1\}.$$

For $k = 0$, $E_{ij}^0 = x_1 \cup \dots \cup x_m$ where

$$q_i x_l \rightarrow q_j \in P, \text{ for } 1 \leq l \leq m, \\ = \emptyset, \text{ otherwise.}$$

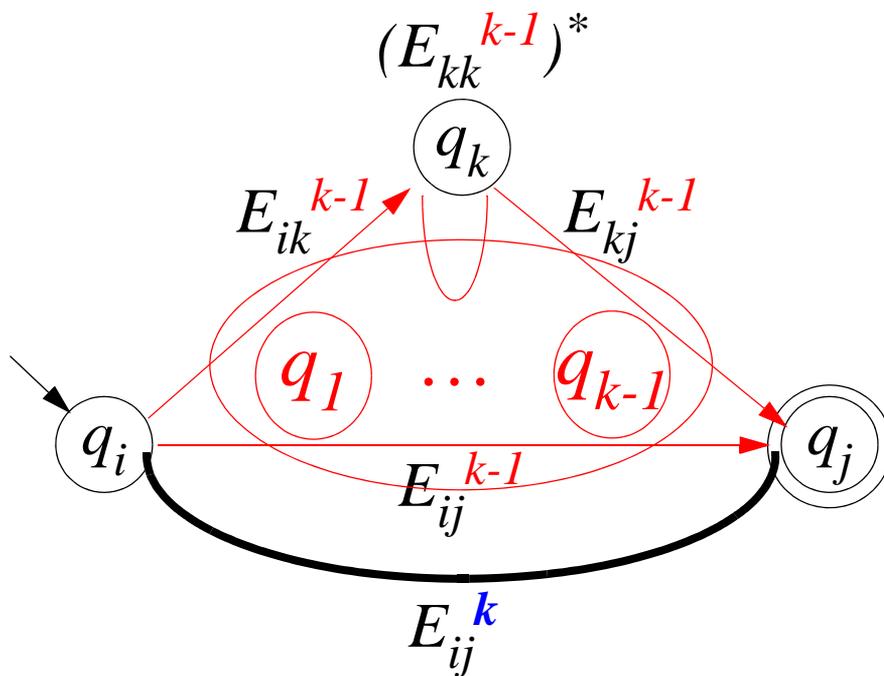
For $k > 0$,

$$(1) E_{ij}^k = (E_{ij}^{k-1})^+, \quad i=j=k.$$

$$(2) E_{ij}^k = E_{ij}^{k-1} \cdot (E_{jk}^{k-1})^* \quad i \neq j = k.$$

$$(3) E_{ij}^k = (E_{ik}^{k-1})^* \cdot E_{kj}^{k-1} \quad i = k \neq j.$$

$$(4) E_{ij}^k = E_{ij}^{k-1} \cup E_{ik}^{k-1} \cdot (E_{kk}^{k-1})^* \cdot E_{kj}^{k-1} \quad i \neq k \neq j.$$



$$E(M) = E_{sf_1}^n \cup \dots \cup E_{sf_m}^n \cup E_\varepsilon.$$

where q_s is a initial state, $\{q_{f_1}, \dots, q_{f_m}\} = F$, and

$$E_\varepsilon = \varepsilon, \text{ if } q_s \in F; E_\varepsilon = \emptyset, q_s \notin F.$$

Theorem 3.18 A language over Σ is **regular**, if and only if it is the language accepted by some **automaton** with input alphabet Σ .

$fa \Rightarrow re$ exponential time bound

$re \Rightarrow fa$ linear time bound

Infinite sequence of regular languages L_1, L_2, \dots

L_n ε -free normal form fa $O(n^2)$,
regular expression 2^n .

fa can be **exponentially more succinct** than re .

finite automaton

\Downarrow $O(n)$ in succinctness

reduced fa

\Downarrow $O(n)$ in succinctness

normal form fa

\Downarrow $O(n^2)$ in succinctness

ε -free fa

\Downarrow $O(2^n)$ in succinctness

regular expression

3.3 Regular Grammars

Let $G = (V, P)$ be a rewriting system.

$$N \cup \Sigma = V, N \cap \Sigma = \emptyset, \text{ and } S \in N.$$

$G = (N, \Sigma, P, S)$ is a **right linear grammar** with

nonterminal alphabet N ,
terminal alphabet Σ ,
start symbol S , and
rules P , if

$$A \rightarrow x, \quad A \rightarrow xB, \quad A, B \in N, x \in \Sigma^*.$$

$G = (N, \Sigma, P, S)$ is a **left linear grammar**, if

$$A \rightarrow x, \quad A \rightarrow Bx, \quad A, B \in N, x \in \Sigma^*.$$

A rewriting system is a **regular grammar**, if

it is either right linear or left linear.

language generated(described) by G

$$L(G) = L_G(S) = \{w \in \Sigma^* \mid S \xRightarrow{*} w \text{ in } G\}.$$

sentential forms in regular grammar

$$S \xRightarrow{*} xA \Rightarrow xyB \xRightarrow{*} xyz, \quad A \rightarrow yB \in P.$$

right linear **single** rightmost nonterminal

$$S \xRightarrow{*} Ax \Rightarrow Byx \xRightarrow{*} zyx, \quad A \rightarrow By \in P.$$

left leaner **single** leftmost nonterminal

$$q_Sxyz \xRightarrow{*} q_Ayz \Rightarrow q_Bz \xRightarrow{*} q_F, \quad q_Ay \rightarrow q_B \in P, \quad q_F \in F.$$

G is **ambiguous**, if some sentence in $L(G)$ has two distinct derivations; otherwise **unambiguous**.

Theorem 3.19 Any fa $M = (Q, \Sigma, P_M, q_s, F)$ can be transformed in time $O(|M|)$ into an equivalent **right-linear grammar** $G(M) = (N, \Sigma, P_G, S)$. Moreover $G(M)$ is unambiguous if and only if M is.

Proof $N = Q, S = q_s,$ and

$$P_G = \{p \rightarrow xq \mid px \rightarrow q \in P_M\} \cup \{f \rightarrow \varepsilon \mid f \in F\}.$$

Theorem 3.20 Any r.l.g. $G = (N, \Sigma, P_G, S)$ can be transformed in time $O(|M|)$ into an equivalent **finite automaton** $M(G) = (Q, \Sigma, P_M, q_s, F)$. Moreover $M(G)$ is unambiguous if and only if G is.

Proof $q_s = [S],$

$$Q = \{[A] \mid A \in N\} \cup \{[xA] \mid A \rightarrow x \in P_G, x \neq \varepsilon\},$$

$$P_M = \{[A]x \rightarrow [B] \mid A \rightarrow xB \in P_G\}$$

$$\cup \{[A]x \rightarrow [xA] \mid A \rightarrow x \in P_G, x \neq \varepsilon\},$$

$$F = \{[A] \mid A \rightarrow \varepsilon \in P_G\} \cup \{[xA] \mid A \rightarrow x \in P_G, x \neq \varepsilon\}.$$

right linear grammar finite automaton

same succinctness and descriptive power

Theorem 3.21 Any language over Σ is **regular** if and only if the language is generated by some **right linear grammar** over Σ .

Theorem 3.23 Any right-linear grammar $G_r = (N_r, \Sigma, P_r, S_r)$ can be transformed in time $O(|M|)$ into an equivalent **left-linear grammar** $G_l = (N_l, \Sigma, P_l, S_l)$. Moreover G_l is unambiguous if and only if G_r is.

Proof. $N_l = N \cup \{S_l\}$

$$P_l = \{B \rightarrow Ax \mid A \rightarrow xB \in P_r\}$$

$$\cup \{S_l \rightarrow Ax \mid A \rightarrow x \in P_r\} \cup \{S_r \rightarrow \varepsilon\}$$

$$S_r = A_0 \Rightarrow x_1 A_1 \Rightarrow \dots \Rightarrow x_1 \dots x_{n-1} A_{n-1} \Rightarrow x_1 \dots x_n \text{ in } G_r.$$

$$A_{i-1} \rightarrow x_i A_i \in P_r \quad 1 \leq i < n, \quad A_{n-1} \rightarrow x_n \in P_r.$$

$$\Downarrow \Uparrow$$

$$S_l \rightarrow A_{n-1} x_n \in P_l, \quad A_i \rightarrow A_{i-1} x_i \in P_l \quad 1 \leq i < n, \quad S_r \rightarrow \varepsilon \in P_l.$$

$$S_l \Rightarrow A_{n-1} x_n \Rightarrow A_{n-2} x_{n-1} x_n \Rightarrow \dots \Rightarrow A_1 x_2 \dots x_n \Rightarrow A_0 x_1 \dots x_n \\ = S_r x_1 \dots x_n \Rightarrow x_1 \dots x_n \text{ in } G_l.$$

$$S_r = A_0 \\ x_1 A_1 \\ x_2 A_2$$

$$S_l \\ A_{n-1} x_n \\ A_{n-2} x_{n-1}$$

$$x_{n-1} A_{n-1} \\ x_n$$

$$A_1 x_2 \\ A_0 = S_r x_1 \\ \varepsilon$$

Theorem 3.22 Any left-linear grammar $G_l = (N_l, \Sigma, P_l, S_l)$ can be transformed in time $O(|M|)$ into an equivalent **right-linear grammar** $G_r = (N_r, \Sigma, P_r, S_r)$. Moreover G_r is unambiguous iff G_l is.

Proof. $N_r = N_l \cup \{S_r\}$

$P_r = \{B \rightarrow xA \mid A \rightarrow Bx \in P_l\}$

$\cup \{S_r \rightarrow xA \mid A \rightarrow x \in P_l\} \cup \{S_l \rightarrow \varepsilon\}$

Theorem 3.24 Any language over an alphabet Σ is **regular** if and only if it is the language generated by some **regular grammar** with input string Σ .

The reversal of a rule $r = \alpha \rightarrow \beta$, $r^R = \alpha^R \rightarrow \beta^R$.

The reversal of G , $G^R = (V, P^R)$.

$$P^R = \{r^R \mid r \in P\}$$

Lemma 3.25 Let $G = (V, P)$ be a rewriting system,

$\pi = r_1 \dots r_n \in P^*$, $\gamma_1, \gamma_2 \in V^*$. Then

$\gamma_1 \xRightarrow{\pi} \gamma_2$ in G , iff $\gamma_1^R \xRightarrow{\pi^R} \gamma_2^R$ in G^R ,

where $\pi^R = r_1^R \dots r_n^R$.

Proof.

"only if"

i) $n=0$, $\pi = \varepsilon$, trivial.

ii) $n>0$, let $\pi = \mu r_n$, $r_n = \omega_1 \rightarrow \omega_2$.

$\gamma_1 \xRightarrow{\mu} \alpha \omega_1 \beta \xRightarrow{r_n} \alpha \omega_2 \beta (= \gamma_2)$ in G .

$\gamma_1^R \xRightarrow{\mu^R} (\alpha \omega_1 \beta)^R (= \beta^R \omega_1^R \alpha^R)$ in G^R by IH.

$\therefore \beta^R \omega_1^R \alpha^R \xRightarrow{r_n^R} \beta^R \omega_2^R \alpha^R = (\alpha \omega_2 \beta)^R = \gamma_2^R$.

$\gamma_1^R \xRightarrow{\mu^R r_n^R} \gamma_2^R$.

"if" trivial, since

if $\gamma_1^R \xRightarrow{\pi^R} \gamma_2^R$ in G^R , $(\gamma_1^R)^R = \gamma_1 \xRightarrow{\pi^{RR}} = \xRightarrow{\pi} (\gamma_2^R)^R = \gamma_2$.

Theorem 3.26 Any rg G can be transformed in time $O(|G|)$ into a **regular grammar** G^R such that

$$L(G^R) = L(G)^R.$$

Proof

G^R is left(right)-linear if G is right(left)-linear.

$|G^R| = |G|$ and G^R can be constructed in time $O(|G|)$.

Theorem 3.27 Family of regular grammar is **effectively closed** under reversal.

3.4 Deterministic Automaton

A finite automaton M is **nondeterministic**, if

$$qw \Rightarrow^{r_1} q_1w_1, qw \Rightarrow^{r_2} q_2w_2, \text{ and } r_1 \neq r_2.$$

M is **deterministic**, if not nondeterministic.

Fact 3.28 A fa is **nondeterministic**, if and only if

$$qx \rightarrow q_1, qy \rightarrow q_2, \text{ where } y \text{ is a prefix of } x.$$

Fact 3.29 A **deterministic fa** is **unambiguous**, provided it has no ε -transitions from final states.

no different accepting states.

deterministic \subset unambiguous

Let $R \subseteq Q$. Then $\delta_x(R) = \{p \mid qx \rightarrow p \in P, q \in R\}$.

$$\delta_x: 2^Q \times \Sigma^* \rightarrow 2^Q.$$

Let $\delta_x(q) = \delta_x(\{q\}) = \{p \mid qx \rightarrow p \in P, q \in R\}$.

Let $\delta_x^n(R) = \{p \mid qx \xrightarrow{n} p \text{ in } M, q \in R\}$.

Let $\delta_x^*(R) = \{p \mid qx \xrightarrow{*} p \text{ in } M, q \in R\}$.

Theorem 3.30 Any fa M can be transformed in time $O(2^{|M|+\log|M|+\log|\Sigma|})$ into an equivalent **deterministic ε -free normal-form fa** \hat{M} of size $O(2^{|M|+\log|\Sigma|})$.

Proof Assume $M = (Q, \Sigma, P, q_s, F)$ is normal-form.

$\hat{M} = (\hat{Q}, \Sigma, \hat{P}, \hat{q}_s, \hat{F})$ where

$$\hat{Q} = 2^Q,$$

$$\hat{q}_s = \{q \in Q \mid q_s \xRightarrow{*} q \text{ in } M\}; \text{ or } \delta_\varepsilon^*(\{q_s\})$$

$$\hat{F} = \{\hat{q} \in \hat{Q} \mid \hat{q} \cap F \neq \emptyset\}, \text{ and}$$

$$\hat{P} = \{\hat{q}_1 a \rightarrow \hat{q}_2 \mid \hat{q}_1 \subseteq Q, a \in \Sigma, \hat{q}_2 = \delta_a^*(\hat{q}_1)\}$$

where

$$\delta_a^*(\hat{q}_1) = \{q_2 \in Q \mid q_1 a \xRightarrow{*} q_2 \text{ in } M, q_1 \in \hat{q}_1\} \in \hat{Q}.$$

$\hat{q}_1 \xRightarrow{*} w\hat{q}_2$ in \hat{M} , iff $\hat{q}_2 = \delta_w^*(\hat{q}_1)$ where

$$\delta_w^*(\hat{q}_1) = \{q_2 \in Q \mid q_1 w \xRightarrow{*} q_2 \text{ in } M, q_1 \in \hat{q}_1\}.$$

$$L(\hat{M}) = \{w \in \Sigma \mid \hat{q}_s w \xRightarrow{*} \hat{q}, \hat{q} \in \hat{F}\}$$

$$= \{w \in \Sigma \mid \delta_w^*(\hat{q}_s) \cap F \neq \emptyset\}$$

$$= \{w \in \Sigma \mid q_s w \xRightarrow{*} q, q \in F\}.$$

$$|\hat{M}| = 3 \cdot |\hat{P}| = 3 \cdot |\hat{Q}| \cdot |\Sigma| = 3 \cdot 2^{|Q|} \cdot |\Sigma| = 3 \cdot 2^{|Q|} + \log|\Sigma|.$$

$$\therefore |\hat{M}| = O(2^{|Q|} + \log|\Sigma|).$$

time complexity

$$O(|\hat{M}| \cdot |Q|) = O(2^{|Q|} + \log|Q| + \log|\Sigma|)$$

$q_1 \xRightarrow{*} q_2$, if and only if $q_1 \delta_{\varepsilon}^* q_2$.

$q_1 a \xRightarrow{*} q_2$, if and only if $q_1 \delta_{\varepsilon}^* \delta_a \delta_{\varepsilon}^* q_2$

$$\delta_a^* = \delta_{\varepsilon}^* \delta_a \delta_{\varepsilon}^*.$$

Furthermore if $w = a_1 \dots a_n$.

$$\begin{aligned} \delta_w^* &= \delta_{\varepsilon}^* \delta_{a_1} \delta_{\varepsilon}^* \delta_{\varepsilon}^* \delta_{a_2} \delta_{\varepsilon}^* \dots \delta_{\varepsilon}^* \delta_{a_n} \delta_{\varepsilon}^*, \\ &= \delta_{\varepsilon}^* \delta_{a_1} \delta_{\varepsilon}^* \delta_{a_2} \dots \delta_{\varepsilon}^* \delta_{a_n} \delta_{\varepsilon}^*. \end{aligned}$$

define $\hat{\delta}_a = \delta_a \delta_{\varepsilon}^*$.

$$\begin{aligned} \delta_w^* &= \delta_{\varepsilon}^* \hat{\delta}_{a_1} \hat{\delta}_{a_2} \dots \hat{\delta}_{a_n} \\ &= \delta_{\varepsilon}^* \hat{\delta}_w. \end{aligned} \quad (Q, \Sigma, P, q_s, F) \text{ is a}$$

$\hat{q}_s := \delta_{\varepsilon}^*(q_s)$;

$\hat{Q} := \{\hat{q}_s\}$; $\hat{P} := \emptyset$;

repeat

for $\hat{q}_1 \in \hat{Q}$ **and** $a \in \Sigma$ **do**

$\hat{Q} := \hat{Q} \cup \hat{\delta}_a(\hat{q}_1)$; ($\hat{\delta}_a = \delta_a \delta_{\varepsilon}^* = \delta_{\varepsilon}^*(\delta_a(\hat{q}_1))$)

$\hat{P} := \hat{P} \cup \hat{q}_1 \cdot a \rightarrow \hat{\delta}_a(\hat{q}_1)$;

od

until no more state is added into \hat{Q} ;

$\hat{F} := \{\hat{q} \in \hat{Q} \mid \hat{q} \cap F \neq \emptyset\}$.

Supplement

Consider a **dfa** $M = (Q, \Sigma, P, q_s, F)$.

$\forall q \in Q, \forall x \in \Sigma^*. \exists! p \in Q$ such that $qx \Rightarrow^* p$.

$\forall q \in Q$, we define $L(q) = \{x \in \Sigma^* \mid q_s x \Rightarrow^* q\}$

Consider a binary relation on Σ^* , $\cong \subseteq \Sigma^* \times \Sigma^*$ as
 $x \cong y$, if $q_s x \Rightarrow^* q$ and $q_s y \Rightarrow^* q$ for some $q \in Q$.

Fact \cong is equivalent.

$\therefore [x]_{\cong} = \{y \in \Sigma^* \mid x \cong y\}$ **equivalent class on Σ^* .**

$\forall q \in Q, \exists x \in \Sigma^*. \exists. L(q) = [x]_{\cong}$.

A state of a **dfa** is an equivalent class of a **partition on Σ^* .**

$$\begin{aligned} q \in Q &\leftrightarrow [x]_{\cong} \in \text{Par}_{\cong}(\Sigma^*) \\ Q &\leftrightarrow \text{Par}_{\cong}(\Sigma^*) \end{aligned}$$

Theorem 3.31**(Characterization of Regular Languages)**

The following statements are logically equivalent for all languages over alphabet Σ .

- (1) L is the language denoted by some **regular expression** over Σ .
- (2) L is the language denoted by some **unambiguous regular expression** over Σ .
- (3) L is the language accepted by some **finite automaton** with input alphabet Σ .
- (4) L is the language accepted by some **deterministic ε -free finite automaton** with
- (5) L is the language generated by some **regular grammar** with terminal alphabet Σ .
- (6) L is the language generated by some **unambiguous right-linear grammar** with
- (7) L is the language generated by some **unambiguous left-linear grammar** with

Moreover, if D is a description of L belonging to any of the above classes of regular language description, then D can be transformed into a equivalent description belonging to any of the other classes.

finite automaton \Leftrightarrow *regular grammar*

$O(n)$ in succinctness

finite automaton \Rightarrow *regular expression*

$O(2^n)$ in succinctness

finite automaton(regular grammar)

$O(n)$ in succinctness

reduced fa(reduced rg)

$O(n)$ in succinctness

normal form fa(normal form rg)

$O(n^2)$ in succinctness

ϵ -free *fa(ϵ -free rg)*

$O(2^n)$ in succinctness

deterministic fa(deterministic rg)

$O(2^n)$ in succinctness

regular expression

$O(2^n)$ in succinctness

unambiguous(deterministic) re

$O(2^{2^n})$

3.5 Decision Problems on Regular Languages

Let \mathbb{D} be a class of regular language description,
 L be a regular language.

$P_{mem}(\mathbb{D})$: "Given $w, D \in \mathbb{D}$; is $w \in L(D)$?"
membership problem for \mathbb{D} . $rep(D)\#w$

$P_{mem}(L)$: "Given w ; is $w \in L$?"
membership problem for L . w

$P_{con}(\mathbb{D})$: "Given $D_1, D_2 \in \mathbb{D}$; is $L(D_1) \subseteq L(D_2)$?"
containment problem for \mathbb{D} . $rep(D_1)\#rep(D_2)$

$P_{ncon}(\mathbb{D})$: "Given $D_1, D_2 \in \mathbb{D}$; is $L(D_1) \not\subseteq L(D_2)$?"
noncontainment problem for \mathbb{D} . $rep(D_1)\#rep(D_2)$

$P_{eq}(\mathbb{D})$: "Given $D_1, D_2 \in \mathbb{D}$; is $L(D_1) = L(D_2)$?"
equivalence problem for \mathbb{D} . $rep(D_1)\#rep(D_2)$

$P_{neq}(\mathbb{D})$: "Given $D_1, D_2 \in \mathbb{D}$; is $L(D_1) \neq L(D_2)$?"
inequivalence problem for \mathbb{D} . $rep(D_1)\#rep(D_2)$

3.6 Applications to Lexical Analysis