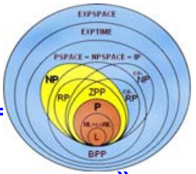


Session 5

- Regular Grammars and Regular Languages
- Identifying Nonregular Languages
- Closure Properties of Regular Languages

Regular Grammars and Regular Languages

Right- and Left-Linear Grammars

A third way of describing the regular languages is by means of certain simple grammars.

A grammar $G = (V, T, P, S)$ is said to be right-linear if all productions are of the form $A \to xB$ and $A \to x$, where $A, B \in V$, and $x \in T^*$. A grammar is said to be left-linear if all productions are of the form $A \to Bx$ and $A \to x$.

A regular grammar is one that is either right-linear or left-linear.

Example The grammar $G_1 = (\{S, A, B\}, \{0, 1\}, P, S)$ with production

$$S \to A01,\ \ A \to A01|B,\ \ B \to 0$$

is left-linear. The sequence $S \Rightarrow A01 \Rightarrow A0101 \Rightarrow B0101 \Rightarrow 00101$ is a derivation with G_1. From this single instance it is to conjecture that $L(G) = L(\mathbf{001(01)}^*)$.

✍ Here we introduce a convenient shorthand notation, we group productions with the same head by |.

Example The grammar $G_2 = (\{S, A, B\}, \{0, 1\}, P, S)$ with production

$$S \to A,\ \ A \to 0B|\epsilon,\ \ B \to A1$$

is not regular. The grammar is an example of a linear grammar.

A linear grammar is a grammar in which at most one variable can occur on the body of any production. Clearly, a regular grammar is always linear, but not all linear grammars are regular.

We will show that regular grammars are another way describing regular languages.

Right-Linear Grammars Generate Regular Languages

Theorem 3.5 *Let $G = (V, T, P, S)$ be a right-linear grammar. Then $L(G)$ is a regular language.*

Proof To do so, we will construct an NFA that mimics the derivations of G. Assume that $V = \{V_0, V_1, \cdots, V_r\}$, that $S = V_0$, and that we have productions of the form

$$V_0 \to v_1 V_i, \ V_i \to v_2 V_j \ \cdots, V_n \to v_l, \cdots.$$

If $w \in L(G)$, then the derivation must have the form

$$V_0 \Rightarrow v_1 V_i \Rightarrow v_1 v_2 V_j \overset{*}{\Rightarrow} v_1 v_2 \cdots v_k V_n \Rightarrow v_1 v_2 \cdots v_k v_l = w.$$

The automaton to be constructed will reproduce the derivation by “consuming” each of these v’s in turn.

The initial state of the automaton will be labeled V_0, and for each variable V_i there will be a nonfinal state labeled V_i.

For each production $V_i \to a_1a_2\cdots a_mV_j$, the automaton will have transitions to connect V_i and V_j that is,

$$\delta^*(V_i, a_1a_2\cdots a_m) = V_j.$$

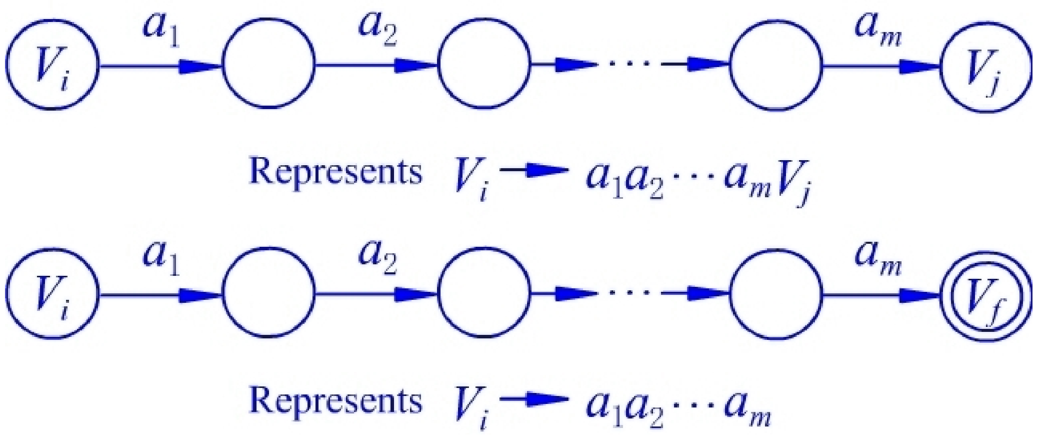
For each production $V_i \to a_1a_2\cdots a_m$, the corresponding transition of the automaton will be

$$\delta^*(V_i, a_1a_2\cdots a_m) = V_f,$$

where V_f is a final state.

The intermediate state that are needed to do this are of no concern and can be given any labels.

The general scheme is shown in following figure. The complete automaton M is assembled from such individual parts.


a_1
a_2
a_m
V_i
V_j
Represents $V_i \to a_1 a_2 \cdots a_m V_j$
a_1
a_2
a_m
V_i
V_f
Represents $V_i \to a_1 a_2 \cdots a_m$

Suppose now that $w \in L(G)$ so that

$$V_0 \Rightarrow v_1 V_i \Rightarrow v_1 v_2 V_j \overset{*}{\Rightarrow} v_1 v_2 \cdots v_k V_n \Rightarrow v_1 v_2 \cdots v_k v_l = w.$$

Clearly $V_f \in \delta^*(V_0, w)$.

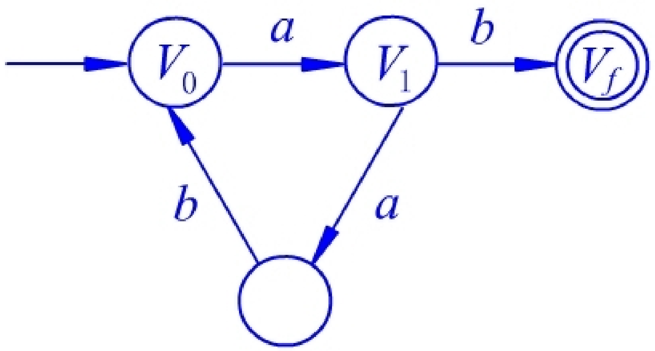
Conversely, assume that w is accepted by M. To accept w the automaton has to pass through a sequence of states $V_0, V_i, \cdots$ to V_f, using paths labeled $v_1, v_2, \cdots$. Therefore, w must have the form $w = v_1 v_2 \cdots v_k v_l$ and the derivation

$$V_0 \Rightarrow v_1 V_i \Rightarrow v_1 v_2 V_j \overset{*}{\Rightarrow} v_1 v_2 \cdots v_k V_n \Rightarrow v_1 v_2 \cdots v_k v_l$$

is possible. Hence $w \in L(G)$. ◀

Example Construct a finite automaton that accepts the language generated by the grammar $V_0 \to aV_1,\ V_1 \to abV_0|b$.

Solution



The language is $L((\mathbf{aab})^*\mathbf{ab})$. ◀

Right-Linear Grammars for Regular Languages

Theorem 3.6 *If L is a regular language on the alphabet Σ, then there exists a right-linear grammar $G = (V, T, P, S)$ such that $L = L(G)$.*

Proof Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that accepts the given language L. Assume that $Q = \{q_0, q_1, \cdots, q_n\}$ and $\Sigma = \{a_1, a_2, \cdots, a_m\}$.

Construct the right-linear grammar $G = (Q, \Sigma, P, q_0)$ with

$$q_i \to a_j q_k, \text{ if } \delta(q_i, a_j) = q_k; \quad q_i \to \epsilon, \text{ if } q_i \in F.$$

Consider $w \in L$ of the form $w = a_i a_j \cdots a_k a_l$. For M to accept this it must make moves via

$$\delta(q_0, a_i) = q_p,\ \delta(q_p, a_j) = q_r,\ \cdots, \delta(q_s, a_k) = q_t,\ \delta(q_t, a_l) = q_f \in F.$$

By construction, the grammar will have one production for each of these δ's. Therefore we can make the derivation

$$q_0 \Rightarrow a_i q_p \Rightarrow a_i a_j q_r \overset{*}{\Rightarrow} a_i a_j \cdots a_k q_t \Rightarrow a_i a_j \cdots a_k a_l q_f \Rightarrow a_i a_j \cdots a_k a_l,$$

with the grammar G, and $w \in L(G)$.

Conversely, if $w \in L(G)$, then its derivation must have the form

$$q_0 \Rightarrow a_i q_p \Rightarrow a_i a_j q_r \overset{*}{\Rightarrow} a_i a_j \cdots a_k q_t \Rightarrow a_i a_j \cdots a_k a_l q_f \Rightarrow a_i a_j \cdots a_k a_l.$$

But this implies that

$$\delta^*(q_0, a_i a_j \cdots a_k a_l) = q_f,$$

completing the proof. ◀

For the purpose of constructing a grammar, it is useful to note that the restriction that M be a DFA is not essential to the proof.

Example Construct a right-linear grammar for $L(\mathbf{aab}^*\mathbf{a})$.

Solution The transition function for NFA is

	a	b
$\rightarrow q_0$	$\{q_1\}$	$\emptyset$
q_1	$\{q_2\}$	$\emptyset$
q_2	$\{q_f\}$	$\{q_2\}$
$\star q_f$	$\emptyset$	$\emptyset$

So the corresponding grammar is $G = (\{q_0, q_1, q_2, q_f\}, \{a, b\}, P, q_0)$ with productions

$$q_0 \rightarrow aq_1,\ \ q_1 \rightarrow aq_2,\ \ q_2 \rightarrow bq_2|aq_f,\ \ q_f \rightarrow \epsilon.$$

◀

Equivalence Between Finite Automata and Regular Grammars

Theorem 3.7 *A language regular if and only if there exists a right-linear grammar* $G = (V, T, P, S)$ *such that* $L = L(G)$.

Similarly, we have

Theorem 3.8 *A language regular if and only if there exists a left-linear grammar* $G = (V, T, P, S)$ *such that* $L = L(G)$.

The proof of Theorem 3.8 depends on the closure under reversal operation for regular languages. We only outline the main idea:

Given any left-linear grammar G with productions $A \to Bv,\ A \to v$, we construct from it a right-linear grammar $\hat{G}$ with $A \to v^R B,\ A \to v^R$, where v^R is the reverse of v. Then we have $L(G) = L(\hat{G})^R$.

Putting above two theorems together, we arrive at our main result.

Theorem 3.9 *A language regular if and only if there exists a regular grammar G such that $L = L(G)$.*

We now have three ways of describing regular languages:

finite automata, **regular expressions,** **regular grammars**

Note that

- While in some instance one or the other of these may be most suitable, they are all equally powerful.
- They all give a complete and unambiguous definition of a regular language.

Identifying Nonregular Languages

The Pumping Lemma for Regular Expressions

Every regular language satisfies the pumping lemma. If somebody presents you with fake regular language, you can use the pumping lemma to show a contradiction.

Example Let's first consider an example and give an informal argument. We claim that the language $L_{01} = \{0^k 1^k \,|\, k \geq 1\}$ is not regular.

Suppose L_{01} were regular. Then it would be recognized by some DFA A, with, say, n states.

Let A read 0^n. On the way it will travel as follows:

$$\begin{array}{cccccc} \epsilon & 0 & 00 & \cdots & \overbrace{00\cdots0}^{n-1} & \overbrace{00\cdots0}^{n} \\ q_0 & q_1 & q_2 & \cdots & q_{n-1} & q_n \end{array}$$

The pigeonhole principle tells us: $\exists\, i < j$ such that $q_i = q_j$. Call this state q.

Now you can fool A:

- If $\hat{\delta}(q, 1^i) \in F$ the machine will foolishly accept $0^j 1^i$.
- If $\hat{\delta}(q, 1^i) \notin F$ the machine will foolishly reject $0^i 1^i$.

Therefore L_{01} cannot be regular!

Theorem 4.1 The Pumping Lemma for Regular Languages *Let L be regular language. Then $\exists\, n, \forall\, w \in L, |w| \geq n$, we can break w into three strings, $w = xyz$ such that*

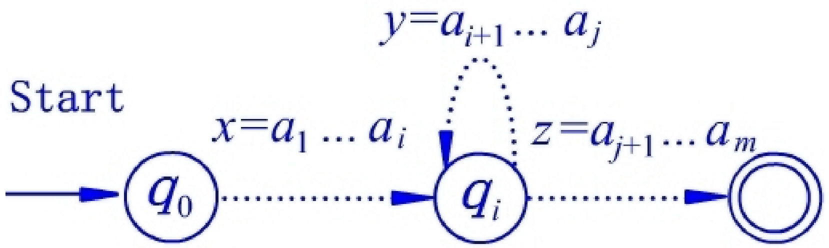
$$1. \quad |y| > 0, \qquad 2. \quad |xy| \leq n, \qquad 3. \quad \forall\, k \geq 0,\ xy^k z \in L.$$

☞ When w is divided into xyz, either x or z may be ϵ, but condition 1 says that $y \neq \epsilon$. Observe that without condition 1 the theorem would be trivially true. Condition 2 states that the pieces x and y together have length at most n. It is an extra technical condition.

Proof Suppose L is regular. Then L is recognized by some DFA A, with, say, n states. Let $w = a_1a_2\cdots a_m \in L$, $m > n$, and let $q_k = \hat{\delta}(q_0, a_1a_2\cdots a_k)$, $k = 1, 2, \cdots, m$. By the pigeonhole principle, $\exists\, i < j$ such that $q_i = q_j$.

Now break $w = xyz$ as follows:

1. $x = a_1 a_2 \cdots a_i$, 2. $y = a_{i+1} a_{i+2} \cdots a_j$, 3. $z = a_{j+1} a_{j+2} \cdots a_m$.



Evidently, $xy^k z \in L$, for any $k \geq 0$. ◄

Example Let L_{eq} be the language of strings with equal number of zero's and one's. Show L_{eq} not to be regular.

Proof Suppose L_{eq} were regular. Then $w = 0^n1^n \in L_{eq}$.

By the pumping lemma $w = xyz$, $|xy| \le n$, $y \ne \epsilon$ and $xy^kz \in L_{eq}$

$$w = \underbrace{00\cdots}_{x}\underbrace{\cdots 00}_{y}\underbrace{111\cdots 11}_{z}$$

In particular, $xz \in L_{eq}$, but xz has fewer 0's than 1's. ◀

Example Let $L_{do} = \{ww \mid w \in \{0,1\}^*\}$. Show L_{do} not to be regular.

Proof Suppose L_{do} were regular. Then $w = 0^n10^n1 \in L_{do}$.

By the pumping lemma $w = xyz$, $|xy| \le n$, $y \neq \epsilon$ and $xy^kz \in L_{do}$

$$w = \underbrace{00\cdots}_{x}\underbrace{\cdots 00}_{y}\underbrace{100\cdots 01}_{z}$$

In particular, $xyyz \in L_{do}$, but $xyyz = 0^m10^n1\,(m > n)$ is not the form ww for any $w \in \{0,1\}^*$. ◀

Example Show that $L_{pr} = \{1^p \mid p\, is prime\}$ is not a regular language.

Proof Suppose L_{pr} were regular. Let n be given by the pumping lemma. Choose a prime $p \geq n+2$, and

$$w = \overbrace{\underbrace{111\cdots\cdot}_{x}\underbrace{\cdots 111}_{y(|y|=m)}\underbrace{111\cdots 11}_{z}}^{p}$$

Now $xy^{p-m}z \in L_{pr}$, $|xy^{p-m}z| = |xz| + (p-m)|y| = p - m + (p-m)m = (1+m)(p-m)$, which is not prime unless one of the factors is 1.

- $y \neq \epsilon \Rightarrow 1 + m > 1$; $m = |y| \leq |xy| \leq n$, $p \geq n+2 \Rightarrow p - m \geq n + 2 - n = 2$.

Example Let $L_{sq} = \{1^{m^2} \mid m \geq 0\}$. Prove L_{sq} not to be a regular language.

Note that the growing gap between successive members of the sequence of perfect squares: $0, 1, 4, 9, 16, 25, 36, 49, \cdots$. Large members of this sequence cannot be near each other.

Consider the two strings xy^iz and $xy^{i+1}z$. If we choose i very large, the lengths of xy^iz and $xy^{i+1}z$ cannot both be perfect squares because they are too close together. Thus these two strings cannot both be in L_{sq}, a contradiction.

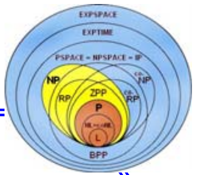
Question Turn this idea into a proof. (Find i that gives the contradiction.)

The pumping lemma is difficult for several reasons.

- Its statement is complicated, and it is easy to go astray in applying it.
- Even if you master the technique, it may still be hard to see exactly how to use it.

The pumping lemma is like a game with complicated rules. Knowledge of the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win!

Break for 15 minutes



Closure Properties of Regular Languages

Closure properties express the idea that when one (or several) languages are regular, then certain related languages are also regular. Let's first summarize the principal closure properties for regular languages.

Let L and M be regular languages. Then the following languages are all regular:

- *Union*: $L \cup M$
- *Intersection*: $L \cap M$
- *Complement*: $\overline{L}$

- *Difference*: $L - M$
- *Closure*: L^*
- *Concatenation*: $L.M$
- *Reversal*: $L^R = \{w^R \mid w \in L\}$
- *Homomorphism*: $h(L) = \{h(w) \mid w \in L,\ h\, is a homomorphism\}$
- *Inverse Homomorphism*: $h^{-1}(L) = \{w \mid h(w) \in L,\ h\, is a homomorphism\}$

Closure Under Boolean Operations

Theorem 4.2 *If L and M are regular languages, then so is $L \cup M$.*

Proof Let $L = L(E)$ and $M = L(F)$. Then $L \cup M = L(E + F)$ by the definition of the + operator for regular expression. ◀

☞ This proof is exceptionally easy because union is one of the three operations that define the regular expressions. The same idea applied to closure and concatenation as well.

Theorem 4.3 *If L is a regular languages over Σ, then so is $\overline{L} = \Sigma^* - L$.*

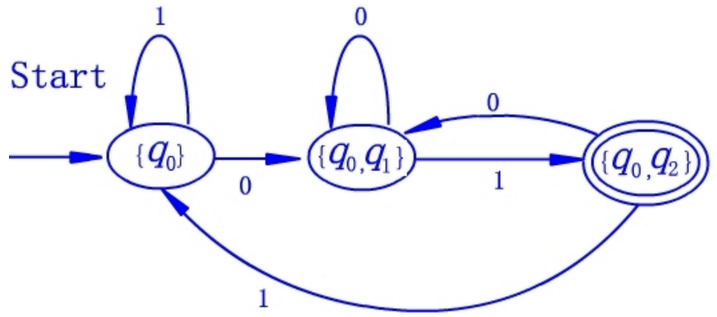
Proof Let L be recognized by a DFA $A = (Q, \Sigma, \delta, q_0, F)$. Let

$$B = (Q, \Sigma, \delta, q_0, Q - F).$$

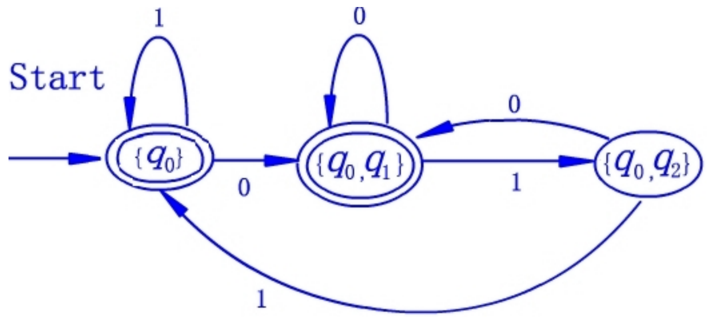
Now $L(B) = \overline{L}$. ◀

☞ Notice that it is important for the above proof that $\hat{\delta}(q_0, w)$ is always some state; i.e. there are no missing transitions in A.

Example Let L be recognized by a DFA



Then $\overline{L}$ is recognized by



Theorem 4.4 *If L and M are regular languages, then so is $L \cap M$.*

Proof By DeMorgan's law $L \cap M = \overline{\overline{L} \cup \overline{M}}$. We already prove that regular languages are closed under complement and union. ◀

Theorem 4.5 *If L and M are regular languages, then so is $L - M$.*

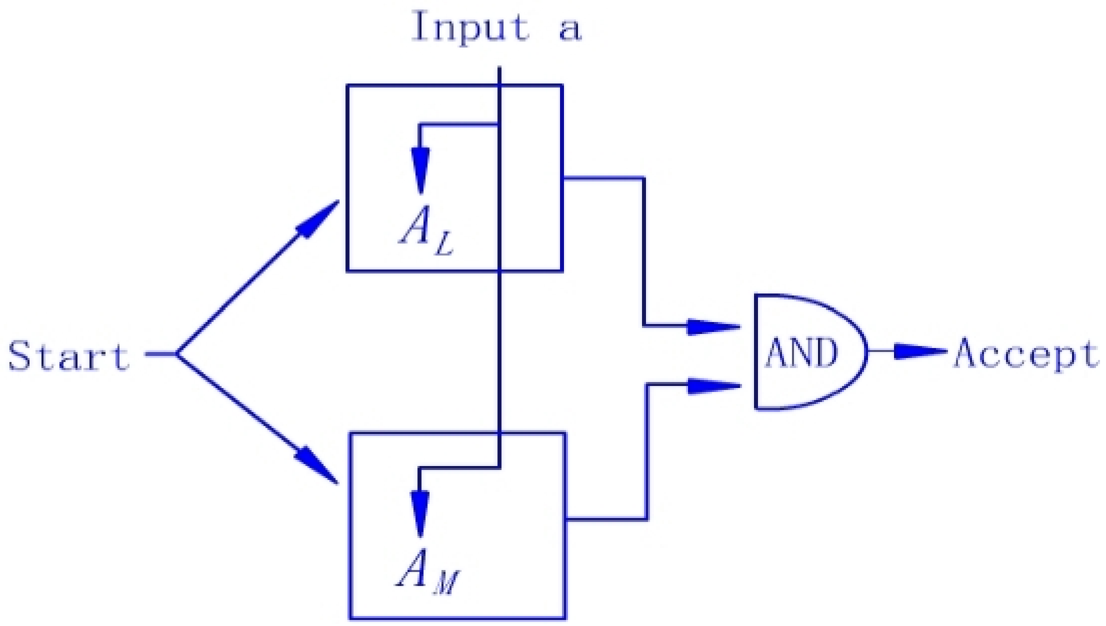
Proof Observe that $L - M = L \cap \overline{M}$. We already prove that regular languages are closed under complement and intersection. ◀

We can also perform a direct construction (product construction) of a DFA for the intersection of two regular languages.

Let L be the language of $A_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$, and M be the language of $A_M = (Q_M, \Sigma, \delta_M, q_M, F_M)$. Assume w.l.o.g. that both automata are deterministic.

We construct an automaton that simulates A_L and A_M in parallel, and accepts if and only if both A_L and A_M accept.

If A_L goes from state p to state s on reading a, and A_M goes from state q to state t on reading a, then $A_{L\cap M}$ will go from state (p, q) to state (s, t) on reading a.


Input a
A_L
A_M
Start
AND
Accept

Formally,

$$A_{L\cap M} = (Q_L \times Q_M, \Sigma, \delta_{L\cap M}, (q_L, q_M), F_L \times F_M),$$

where

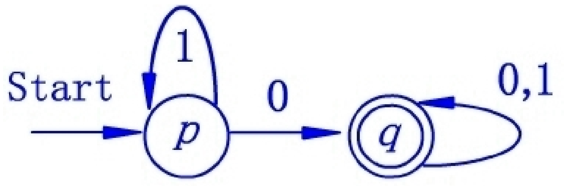
$$\delta_{L\cap M}((p, q), a) = (\delta_L(p, a), \delta_M(q, a)).$$

It will be shown by induction on w that

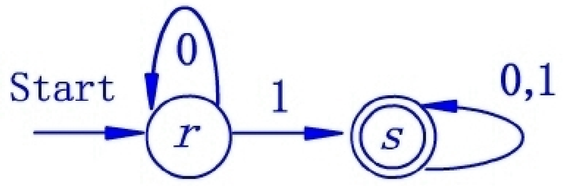
$$\hat{\delta}_{L\cap M}((q_L, q_M), w) = \left(\hat{\delta}_L(q_L, w), \hat{\delta}_M(q_M, w)\right).$$

The claim then follows.

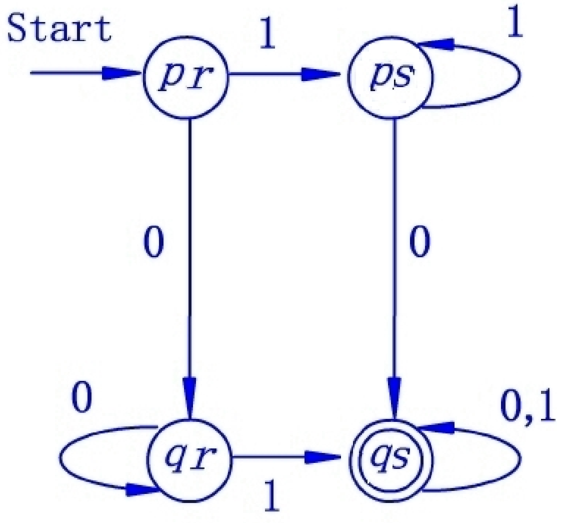
Example $(c) = (a) \times (b)$



(a)



(b)



(c)

Closure Under Reversal Operation

The reversal of a string $a_1a_2 \cdots a_n$ is the string written backwards, that is, $a_na_{n-1} \cdots a_1$. Use w^R for the reversal of w.

Example 0010^R is 0100, and $\epsilon^R = \epsilon$. We say string w is a palindrome, if $w = w^R$.

The reversal of a language L is the language defined by $L^R = \{w^R \mid w \in L\}$.

Theorem 4.5 *If L is a regular language, then so is L^R.*

Proof 1 Let L be recognized by an FA A. Turn A into an FA for L^R, by

- Reversing all arcs.
- Make the old start state the new sole accepting state.
- Create a new start state p_0, with $\delta(p_0, \epsilon) = F$ (F is the set of old accepting states).

Proof 2 Let L be described by a regular expression E. We shall construct a regular expression E^R, such that $L(E^R) = (L(E))^R$.

We proceed by a structural induction on E.

Basis step: If E is ϵ, $\emptyset$, or **a**, then $E^R = E$. That is we know $L(E^R) = L(E)$.

Inductive step:

- $E = F + G$.

  Let $E^R = F^R + G^R$. Since $L(F^R) = (L(F))^R$, $L(G^R) = (L(G))^R$, so

$$L(E^R) = L(F^R) \cup L(G^R) = (L(F))^R \cup L((G))^R = (L(F) + L(G))^R = (L(E))^R.$$

- $E = FG$.

  Let $E^R = G^R F^R$. Note that $w = w_1 w_2$ iff $w^R = w_2^R w_1^R$. By the inductive hypothesis, $L(E^R) = L(G^R).\ L(F^R) = (L(G))^R.\ (L(F))^R = (L(F).\ L(G))^R = (L(E))^R$.

- $E = F^*$.

  Let $E^R = (F^R)^*$. Since $\forall\, n$, $w = w_1 w_2 \cdots w_n$ iff $w^R = w_n^R w_{n-1}^R \cdots w_1^R$. By the inductive hypothesis,

$$L(E^R) = (L(F^R))^* = ((L(F))^R)^* = ((L(F))^*)^R = (L(F^*))^R = (L(E))^R.$$

◀

Closure Under Homomorphism

Let h be a function from Σ to Λ^*, h:a$\mapsto$ y, y is a string of symbols in alphabet Λ.

We can extend function h to Σ^* :

$$\text{If } w = a_1 a_2 \cdots a_n, \text{ then } h(w) = h(a_1)h(a_2)\cdots h(a_n).$$

The extended function h is called a homomorphism on alphabet Σ.

Example The function h: $\{0, 1\} \to \{a, b\}^*$ defined by $h(0) = ab$ and $h(1) = \epsilon$. Now $h(0011) = abab$.

We can apply a homomorphism to a language. If L is a language over alphabet Σ, and h is a homomorphism on Σ, then

$$h(L) = \{h(w) \,|\, w \in L\}.$$

Example The homomorphism h: $\{0, 1\}^* \to \{a, b\}^*$ defined by $h(0) = ab$ and $h(1) = \epsilon$. Then $h(L(\mathbf{10}^*\mathbf{1})) = L((\mathbf{ab})^*)$.

☞ In general, if E is a regular expression with symbols in Σ, let $h(E)$ be the expression obtained by replacing each symbol a of Σ in E by $h(a)$.

Theorem 4.6 *If L is a regular language over Σ and h is a homomorphism on Σ, then $h(L)$ is also regular.*

Proof Let $L = L(E)$ for a regular expression E. We claim that $h(E)$ is also regular expression, and $h(L(E)) = L(h(E))$. The proof is an easy structural induction.

Basis step: If E is ϵ or $\emptyset$. Then $h(E) = E$, and $h(L(E)) = L(E) = L(h(E))$. If E is **a**, then $L(E) = \{a\}$, $h(E)$ is the regular expression that is the string of symbols $h(a)$. So $h(L(E)) = \{h(a)\} = L(h(E))$.

Inductive step:

- $E = F + G$.

  Apply homomorphism to regular expressions assures us that $h(E) = h(F + G) = h(F) + h(G)$. Since $L(E) = L(F) \cup L(G)$, and $h(L(F)) = L(h(F))$, $h(L(G)) = L(h(G))$, so $h(L(E)) = h(L(F) \cup L(G)) = h(L(F)) \cup h(L(G)) = L(h(F)) \cup L(h(G)) = L(h(F) + h(G)) = L(h(E))$.

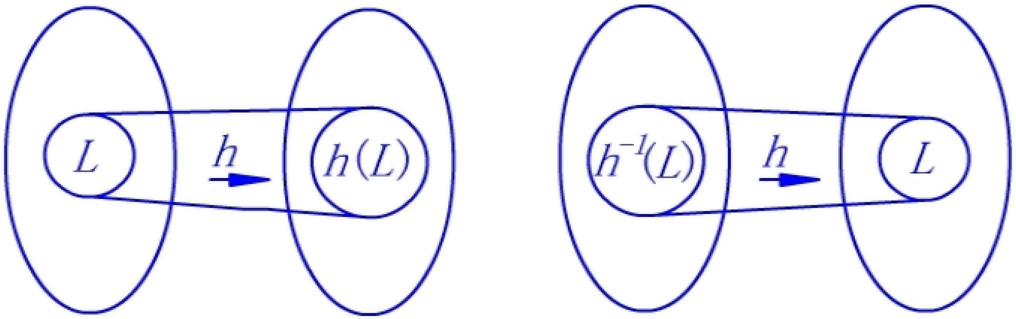
- $E = FG$ and $E = F^*$.

  Similar.

Closure Under Inverse Homomorphism

Let $h : \Sigma^* \to \Lambda^*$ be a homomorphism. Let $L \subseteq \Lambda^*$, and define

$$h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$$



Example Let $h : \{a, b\}^* \to \{0, 1\}^*$ be defined by $h(a) = 01$, and $h(b) = 10$.

Let $L = L((\mathbf{00} + \mathbf{1})^*)$. We claim

$$h^{-1}(L) = L((\mathbf{ba})^*)$$

We shall prove

$$h(w) \in L \;\; if and only if \text{ w=(ba)}^n, (n = 0, 1, 2, \cdots)$$

Proof (If) Let $w = (ba)^n$. Then $h(w) = (1001)^n \in L$.

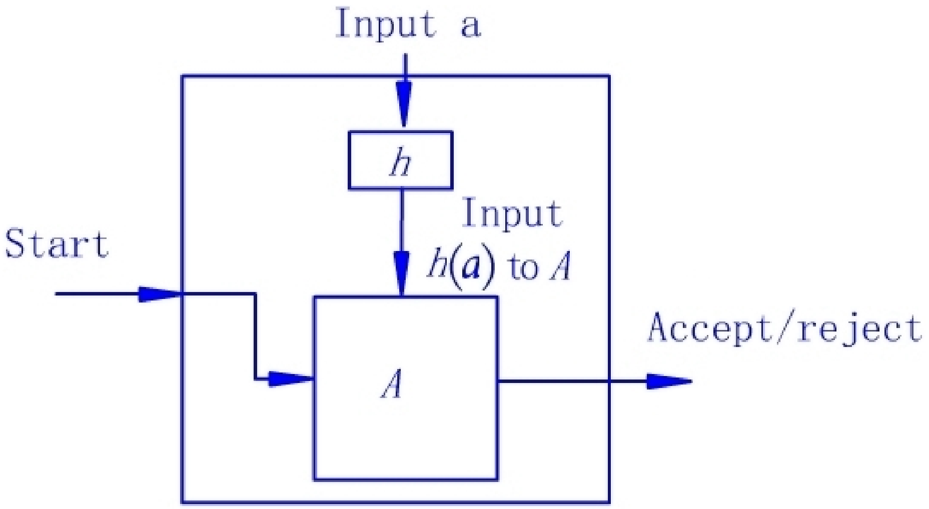
(Only if) Let $h(w) \in L$, and suppose $w \notin L((\mathbf{ba})^*)$. There four cases to consider.

- w begins with a. Then $h(w)$ begins with 01 and is not in $L((\mathbf{00} + \mathbf{1})^*)$.
- w ends in b. Then $h(w)$ ends in 10 and is not in $L((\mathbf{00} + \mathbf{1})^*)$.
- $w = xaay$. Then $h(w) = z0101v$ and is not in $L((\mathbf{00} + \mathbf{1})^*)$.
- $w = xbby$. Then $h(w) = z1010v$ and is not in $L((\mathbf{00} + \mathbf{1})^*)$.

◀

Theorem 4.7 *Let $h : \Sigma^* \to \Lambda^*$ be a homomorphism, and $L \subseteq \Lambda^*$ regular, then $h^{-1}(L)$ is also regular.*

Proof Let $L = L(A)$ for DFA A. We construct from A and h a DFA for $h^{-1}(L)$.



Formally, let L be the language of $A = (Q, \Lambda, \delta, q_0, F)$. Define a DFA

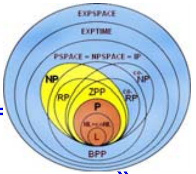
$$B = (Q, \Sigma, \gamma, q_0, F)$$

where

$$\gamma(q, a) = \hat{\delta}(q, h(a))$$

It is an easy induction on $|w|$ to show that $\hat{\gamma}(q_0, w) = \hat{\delta}(q_0, h(w))$. Since the accepting states of A and B are the same, B accepts w if and only if A accepts $h(w)$.

Put another way, B accepts exactly those strings w that are in $h^{-1}(L)$. ◄


EXPSPACE
EXPTIME
PSPACE = NPSPACE = IP
NP
co-NP
RP
ZPP
co-RP
P
L
BPP

Thank you!