Introduction to the Theory of Computation

XU Ming

School of Software Engineering, East China Normal University

April 1, 2025

OUTLINE

- Regular Grammars and Regular Languages
- Identifying Nonregular Languages

Regular Grammars and Regular Languages

Grammars

A grammar is a 4-tuple $G = (V, T, P, S)$, where

- V is a finite set of variables,
- T is a finite set of terminal symbols,
- P is a finite set of productions of the form $x \to y$, where $x \in (V \cup T)^+$ and $y \in (V \cup T)^*$,
- $S \in V$ is a designated variable called the start symbol.

Now we develop the notation for describing the derivations.

Let $G = (V, T, P, S)$ be a grammar, $\alpha, \beta \subset (V \cup T)^*$, and $x \to y \in P$. Then we write

$$\alpha x \beta \underset{G}{\Longrightarrow} \alpha y \beta$$

or, if G is understood

$$\alpha x \beta \Rightarrow \alpha y \beta$$

and say that $\alpha x \beta$ derives $\alpha y \beta$.

Specially, we have $x \Rightarrow y$.

We may extend the $\Rightarrow$ relationship to present zero, one, or many derivation steps.

In other words, we define $\stackrel{*}{\Rightarrow}$ to be the reflexive and transitive closure of $\Rightarrow$, as follows:

Basis step: Let $\alpha \in (V \cup T)^*$. Then $\alpha \stackrel{*}{\Rightarrow} \alpha$.

Inductive step: If $\alpha \stackrel{*}{\Rightarrow} \beta$ and $\beta \Rightarrow \gamma$, then $\alpha \stackrel{*}{\Rightarrow} \gamma$.

Using induction we can prove that if $\alpha \stackrel{*}{\Rightarrow} \beta$ and $\beta \stackrel{*}{\Rightarrow} \gamma$, then $\alpha \stackrel{*}{\Rightarrow} \gamma$.

Let $G(V, T, P, S)$ be a grammar. Then the set

$$L(G) = \{w \in T^* \mid S \overset{*}{\underset{G}{\Longrightarrow}} w\}$$

is the language generated by G.

If $w \in L(G)$, then the sequence $S \Rightarrow w_1 \Rightarrow \cdots \Rightarrow w_n = w$ is a derivation of the sentence w. The strings $S, w_1, \ldots, w_n$, which contain variables as well as terminals, are called sentential forms of the derivation.

In general, we call $L = \{w \in T^* \mid A \overset{*}{\underset{G}{\Longrightarrow}} w\}$ the language of variable A if $A \in V$.

Example

Consider the grammar $G = (V, \{a, b\}, P, S)$ with P given by

$$S \to aSb, S \to \epsilon.$$

The string *aabb* is a sentence in the language generated by G.

A grammar G completely defines $L(G)$, but it may not be easy to get a very explicit description of the language from the grammar.

Here, however, the answer is fairly clear. It is not hard to conjecture that $L(G) = \{a^n b^n \mid n \geq 0\}$, and it is easy to prove it.

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 productions

$$S \to A01, \quad A \to A01 \mid B, \quad B \to 0$$

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

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 productions

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

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 imitates the derivations of G. Assume that $V = \{V_0, V_1, \ldots\}$, that $S = V_0$, and that we have productions of the form

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

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

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

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,

$$V_j \in \hat{\delta}(V_i, a_1a_2\cdots a_m).$$

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

$$V_f \in \hat{\delta}(V_i, a_1a_2\cdots a_m),$$

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 figures. The complete automaton M is assembled from such individual parts.

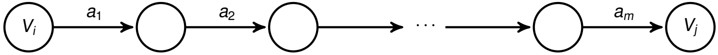


Figure: Represents $V_i \to a_1 a_2 \cdots a_m V_j$

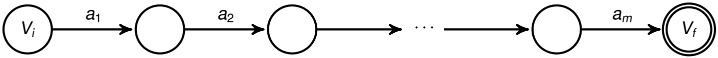


Figure: Represents $V_i \to a_1 a_2 \cdots a_m$

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

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

Clearly $V_f \in \hat{\delta}(V_0, w)$.

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

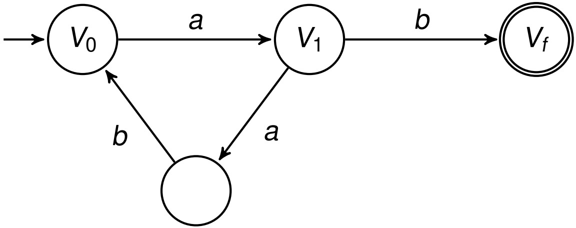
$$V_0 \Rightarrow v_1 V_i \Rightarrow v_1 v_2 V_j \stackrel{*}{\Rightarrow} v_1 v_2 \cdots v_k V_n \Rightarrow v_1 v_2 \cdots v_k v_l$$

exists. Hence $w \in L(G)$. □

Example

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

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 $G = (V, \Sigma, P, S)$ 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 $Q = \{q_0, q_1, \ldots, q_n\}$ and $\Sigma = \{a_1, a_2, \ldots, a_m\}$.

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

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

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

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

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

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

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

$$\hat{\delta}(q_0, a_i a_j \cdots a_k a_l) = q_f,$$

completing the proof. □

For the purpose of constructing a grammar, it is useful to note: 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	$\{q_f\}$	$\{q_2\}$
$\bigstar q_f$	$\emptyset$	$\emptyset$

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

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

Equivalence between Finite Automata and Regular Grammars

Theorem 3.7

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

Proof The proof 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 G' with $A \to v^R B, A \to v^R$, where v^R is the reverse of v. Then we have $L(G) = (L(G'))^R$.

Putting above two theorems together, we arrive at the result.

Theorem 3.8

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

We now have several ways of describing regular languages: finite automata, regular expressions, and regular grammars.

- While in some instance one or the others 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 us first consider an example and give an informal argument. We claim that the language $L_{eq} = \{0^n 1^n \mid n \geq 1\}$ is not regular.

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 us first consider an example and give an informal argument. We claim that the language $L_{eq} = \{0^n 1^n \mid n \geq 1\}$ is not regular.

Suppose L_{eq} is 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:

$$\begin{array}{cccccc} \epsilon & 0 & 00 & \cdots & \overbrace{00\cdots 0}^{n-1} & \overbrace{00\cdots 0}^{n} \\ p_0 & p_1 & p_2 & \cdots & p_{n-1} & p_n \end{array}$$

The pigeonhole principle tells us: $\exists\, i < j$ such that $p_i = p_j$. 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_{eq} cannot be regular!

Theorem 4.1 (Pumping Lemma for Regular Languages)

Let L be regular language. Then there is an $n \in \mathbb{N}$ such that any $w \in L$ with $|w| \geq n$ can be divided into three strings, $w = xyz$, satisfying:

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

That is,

$$\exists\, n \in \mathbb{N} : \forall\, w \in L : \left[|w| \geq n \to \exists \text{ split } w = xyz : \left(\begin{array}{l} |y| > 0 \wedge \\ |xy| \leq n \wedge \\ \forall\, k \in \mathbb{Z}_{\geq 0} : xy^k z \in L \end{array} \right) \right].$$

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.

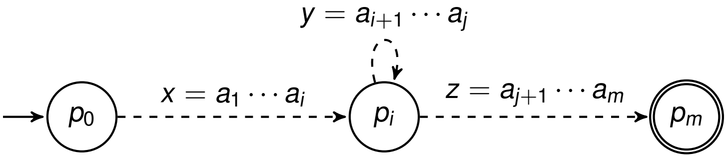
Proof Suppose L is regular. Then L is recognized by some DFA A, with, say, n states. Let $w = a_1 a_2 \cdots a_m \in L$, $m > n$, and let $p_k = \hat{\delta}(q_0, a_1 a_2 \cdots a_k), k = 1, 2, \cdots, m$. By the pigeonhole principle, $\exists\, i < j$ such that $p_i = p_j$. In other words, p_j is the first repeated state along with $p_0, p_1, \ldots, p_m$.

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 $w = a_1 a_2 \cdots a_m \in L, m > n$, and let $p_k = \hat{\delta}(q_0, a_1 a_2 \cdots a_k), k = 1, 2, \cdots, m$. By the pigeonhole principle, $\exists\, i < j$ such that $p_i = p_j$. In other words, p_j is the first repeated state along with $p_0, p_1, \ldots, p_m$.

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, $xy^k z \in L$ holds for any $k \geq 0$; while $|xy| \leq n$ follows the fact there is no repeated state in $p_0, p_1, \ldots, p_{j-1}$. □

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} is regular. Let n be the constant mentioned in the pumping lemma. Then $w = 0^n 1^n \in L_{eq}$.

By the pumping lemma $w = xyz, |xy| \leq n, y \neq \epsilon$ and $xy^k z \in L_{eq}$

$$w = \underbrace{00\cdots}_{x}\underbrace{\cdots 00}_{y}\underbrace{0111\cdots 11}_{z}$$

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

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} is regular. Let n be the constant mentioned in the pumping lemma. Then $w = 0^n 1^n \in L_{eq}$.

By the pumping lemma $w = xyz, |xy| \leq n, y \neq \epsilon$ and $xy^k z \in L_{eq}$

$$w = \underbrace{00\cdots\cdot}_{x}\underbrace{\cdots 00}_{y}\underbrace{0111\cdots 11}_{z}$$

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

Example

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

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

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

$$w = \underbrace{00\cdots}_{x}\underbrace{\cdots 00}_{y}\underbrace{0100\cdots 01}_{z}$$

In particular, $xyyz \in L_{do}$, but $xyyz = 0^m10^n1$ $(m \ge n+2)$ is not the form ww for this w. □

Example

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

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

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

$$w = \underbrace{00\cdots}_{x}\underbrace{\cdots 00}_{y}\underbrace{0100\cdots 01}_{z}$$

In particular, $xyyz \in L_{do}$, but $xyyz = 0^m10^n1$ $(m \geq n+2)$ is not the form ww for this w. □

Example

Show that $L_{pr} = \{1^p \mid p \text{ is prime}\}$ 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

$$w = \overbrace{\underbrace{111\cdots}_{x}\ \underbrace{\cdots111}_{y\ (|y|=m)}\ \underbrace{111\cdots11}_{z}}^{p}$$

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 only if one of the factors is 1.

- $y \neq \epsilon$ implies $1 + m \geq 2$;
- $m = |y| \leq |xy| \leq n$ and $p \geq n + 2$ imply $p - m \geq n + 2 - n = 2$.

Example

Show that $L_{pr} = \{1^p \mid p \text{ is prime}\}$ 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

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

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 only if one of the factors is 1.

- $y \neq \epsilon$ implies $1 + m \geq 2$;
- $m = |y| \leq |xy| \leq n$ and $p \geq n+2$ imply $p - m \geq n + 2 - n = 2$.

$\square$

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: $0, 1, 4, 9, 16, 25, 36, 49, \ldots$. 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 since 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.)

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: $0, 1, 4, 9, 16, 25, 36, 49, \ldots$. 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 since 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!

Homework

Exercises 4.1.2(b) & (f)

A nonregular language meets the pumping lemma

Consider the language $L = L_1 \cup L_2$ over alphabet $\Sigma = \{a, b, c, d\}$, where

$$L_1 = \left\{uvwxy : u, y \in \Sigma^* \wedge v, w, x \in \Sigma \wedge (v = w \vee v = x \vee w = x)\right\}$$
$$L_2 = \left\{w \in \Sigma^* : \text{exactly } \tfrac{1}{6} \text{ positions in } w \text{ are occupied by } d\right\}$$
$$L_3 = \left\{(abc)^i(abd)^i : i \geq 0\right\}.$$

L_2 can be simplified to L_3. In the following, we will show:

1. L meets the pumping lemma, but
2. it is not regular.

Thereby, the counter-example evidences that the pumping lemma is a necessary condition to regular languages, but not a sufficient one!

A nonregular language meets the pumping lemma

Claim 1: L meets the pumping lemma.

Let $n = 5$. For any $w = w_1 w_2 \ldots w_m \in L$ with length $m \geq 5$, there exist two indices $1 \leq i < j \leq 5$ such that $w_i = w_j$ as $|\Sigma| = 4$. Then we make a discussion based on the gap $j - i$, which ranges over $\{1, 2, 3, 4\}$.

- If $j - i = 1$ or 2, those strings pumped with $(w_{i-1})^k$ or $(w_{j+1})^k$ are in L_1, for any $k \geq 0$.
- If $j - i = 3$ or 4, those strings pumped with $(w_{i+1} w_{i+2})^k$ are in L_1.

A nonregular language meets the pumping lemma

Claim 2: L is not regular.

It is not hard to see that the string $(abc)^i(abd)^j$ is in L if and only if $i = j$. Particularly, $(abc)^i(abd)^i$ is in L_2, not L_1.

Recall that $L_{eq} = \{A^iB^i : i \geq 0\}$ is not regular.

So, our claim follows by choosing $A = abc$ and $B = abd$.

A nonregular language meets the pumping lemma

Another example is the language $L' = L'_1 \cup L'_2$ over alphabet $\Sigma = \{0, 1, 2\}$, where

$$L'_1 = \left\{01^i2^i : i \geq 0\right\}$$
$$L'_2 = \left\{0^j w : j \neq 1 \wedge w \in \{1, 2\}^*\right\}.$$