Grammar

Regular Grammar

Context-sensitive Grammar

Context-free Grammar

Languages

Regular Languages

Regular Language if and only if there exist a finite automaton
\[ M = (Q, \Sigma, \delta, q_0, F) \]
such that:
\[ L = L(M) = \{ w \in \Sigma^* : \delta(q_0, w) \in F \} \]

Equivalence between Regular Grammars and Regular Languages

Theorem 1

If \( L \) is a regular language then there is a right-linear grammar
\[ G = (V, \Sigma, S, P) \] such that \( L = L(G) \).

Proof. \( L \) is a regular implies (by def.) there exist a finite automaton
\[ M = (Q, \Sigma, \delta, q_0, F) \] such that \( L(M) = L \). Now we construct the equivalent grammar \( G \) as follows:
- Variables are the states: \( V = Q \)
- Start symbol is the start state: \( S = q_0 \)
- Same alphabet of terminals \( \Sigma \)
- A transition \( \delta(q_1, a) = q_2 \) becomes the rule \( q_1 \rightarrow aq_2 \)
- Accept states \( q \in F \) define the \( \rightarrow \) productions \( q \rightarrow \) ?

Accepted paths give rise to terminating derivations and vice versa. \( L(G) = L(M) \).

Example 1

The DFA above can be simulated by the grammar
\[ x \rightarrow 0x | 1y \\
y \rightarrow 0x | 1z \\
z \rightarrow 0x | 1z | \lambda \]

Example 1

\[ x \Rightarrow 1y \]

\[ 10011 \]
Example 1

\[ x \rightarrow 0x \mid 1y \]
\[ y \rightarrow 0x \mid 1z \]
\[ z \rightarrow 0x \mid 1z \mid \lambda \]

\[ x \Rightarrow 1y \Rightarrow 10x \Rightarrow 1001y \]

10011

**Theorem 2**

If \( G = (V, T, S, P) \) is a right-linear grammar then \( L(G) \) is a regular language.

Proof. Define a FA \( M = (Q, \Sigma, \delta, q_0, F) \) as follows:
- Start state \( q_0 \) correspond to start symbol \( S \)
- A non-final state \( q \) corresponds to a variables symbol \( V_i \)
- Same alphabet of terminals \( \Sigma = \) \( T \)
- For every rule \( V_i \rightarrow a_1 \ldots a_m V_j \), define a transition \( \delta(q, a_1 \ldots a_m) = q \)
- For every rule \( V_i \rightarrow a_1 \ldots a_m \), define a transition \( \delta(q, a_1 \ldots a_m) = q \) final state

Terminating derivations give rise to accepted paths and vice versa. So \( L(M) = L(G) \). Hence (by def.) \( L(G) \) is a regular language.

\[ \square \]
Equivalence between Regular Grammars and Regular Languages

Example 2

Construct an FA that is equivalent to the right-linear grammar:

\[ S \rightarrow aA \\
A \rightarrow abS \\
A \rightarrow b \\
S \rightarrow a \\
A \rightarrow a \]

Answer:

\[ \]

THEOREM 1 and THEOREM 2 show that right-linear grammars and regular languages are equivalent.

Similarly we can show that left-linear grammars and regular languages are equivalent.

Hence we conclude that Regular Grammars and Regular Languages are equivalent.

Comments

Can every CFG be converted into a right linear grammar?

No! This would mean that all context free languages are regular.

For example:

\[ S \rightarrow ? | aSb \]

Cannot be converted because \( \{a^n b^n \} \) is not regular.

How we can identify non-regular languages?

By using a technique called “Pumping Lemma”

Consider the language \( L_1 = 01^* = \{0, 01, 011, ... \} \)

The string 011 is said to be pumpable in \( L_1 \)

Because can take the underlined portion, and pump it up (i.e. repeat) as much as desired while always getting elements in \( L_1 \).

Which of the following are pumpable?

1. 01111
2. 01
3. 0

A:

1. Pumpable: 01111, 01111, 01111, 01111, etc.
2. Pumpable: Q
3. 0 not pumpable because most of 0* not in \( L_1 \)
Define \( L_2 \) by the following automaton:

Is 01010 pumpable?

\[ L_2 = \{010, 01010\} \]

Pumpable: 01010, 010. Underlined substrings correspond to cycles in the FA!

Cycles in the FA can be repeated arbitrarily often, hence pumpable.

Let \( L_3 = \{011, 1010, 000, \lambda\} \)

Which strings are pumpable?

None! When pumping any string non-trivially, always result in infinitely many possible strings. So no pumping can go on inside a finite set.

Pumping Lemma give a criterion for when strings can be pumped.

We have: \( ababaaab \in L(M) \)

Because:

\[
q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_3 \rightarrow q_2 \rightarrow q_0 \rightarrow q_1 \rightarrow q_3
\]

Theorem

- Given an (infinite) regular language \( L \), there is a number \( p \) (called the \textbf{pumping number}) such that any string in \( L \) of length \( \geq p \) is pumpable within its first \( p \) letters.
- In other words, for all \( u \in L \) with \( |u| \geq p \) we can write:
  - \( u = xyz \) (\( x \) is a prefix, \( z \) is a suffix)
  - \(|y| \geq 1\) (mid-portion \( y \) is non-empty)
  - \(|xy| \leq p\) (pumping occurs in first \( p \) letters)
  - \(xyz \in L \) for all \( i \geq 0 \) (can pump \( y \)-portion)

To prove the Pumping Lemma we need to know the \textbf{Pigeonhole Principle}.

...
Given a “sufficiently” long string, the states of a DFA must repeat in an accepting computation. These cycles can then be used to predict (generate) infinitely many other strings in (of) the language.

Pigeon-Hole Principle

Now consider an accepted string $u$. By assumption $L$ is regular so let $M$ be the FA accepting it.

Let $p = |Q| = \text{no. of states in } M$.

Suppose $|u| \geq p$.

The path labeled by $u$ visits $p+1$ states in its first $p$ letters.

Thus (by pigeonhole principle) $u$ must visit some state twice.

The sub-path of $u$ connecting the first and second visit of the vertex is a loop, and gives the claimed string $y$ that can be pumped within the first $p$ letters.

Proof

Pumping Lemma (PL)

Notes:

• It is a necessary condition.
  – Every regular language satisfies it.
  – If a language violates it, it is not regular.
    • RL $\implies$ PL
    not PL $\implies$ not RL

• It is not a sufficient condition.
  – Not every non-regular language violates it.
    • not RL $\implies$? PL or not PL (no conclusion)

Negation of the necessary condition:

$\exists u \in L: |u| \geq p \land$
$\forall x, y, z: (xyz = u) \land (|xy| \leq p) \land (|y| \geq 1) \land (\forall i: i \geq 0 \implies xy^i z \in L)$

Pumping Lemma (PL)

Notes:

For all sufficiently long strings $u$:

There exists a non-null prefix $(xy)$ and substring $(y)$

For all repetitions of the substring $(y)$, we get strings in the language.

In general, to prove that $L$ isn’t regular:

1. Assume $L$ were regular
2. Therefore it has a pumping no. $p$
3. Find a string pattern involving the length $p$ in some clever way, and which cannot be pumped. This is the hard part.
4. $(2) \implies (3)$ <contradiction> Therefore our assumption (1) was wrong and conclude that $L$ is not a regular language
Show that $L = \{a^nb^n \mid n = 0, 1, 2, \ldots \}$ is not regular.

1. Assume $L$ were regular.
2. Therefore it has a pumping number $p$, say $p=2$.
3. But... consider the string $u = a^2b^2$. We have $|u|=4 > p=2$, let $x=a, y=a$, and $z=bb \Rightarrow u=xyz, |y|=1, \text{ and } |xy|=p=2$. PL: $xy^i z \in L$ for all $i \geq 0$.
   Taking $i=0 \Rightarrow xz \in L \Rightarrow abb \in L$.
4. $(2) \rightarrow \leftarrow (3)$ <contradiction> Therefore our assumption (1) was wrong and conclude that $L$ is not a regular language.

Example 1

**Pumping Lemma (PL)**

Show that the following languages are not regular:

- $L_p = \{a^p \mid p \text{ is a prime number} \}$
- $L_c = \{a^c \mid c \text{ is a composite number} \}$
- $L = \{\omega \in \{a, b\}^* \mid \#a's \text{ in } \omega = \#b's \text{ in } \omega \}$
- $L_{pal} = \{x \in \{a, b\}^* \mid x = x^R \}$