Context-free Grammars

Narges Khakpour

Department of Computer Science,
Linnaeus University

Table of Content

Recall...

- Alphabet, e.g. $\Sigma = \{a, b, c\}$
- Strings, e.g. *abbb*
- Σ^* all possible strings defined over Σ
- Languages, i.e. $L \subseteq \Sigma^*$, e.g. $L = \{a, ac, bbc\}$
- Regular expressions to represent languages (the operators of $+$ and $*$)
- Regular languages vs irregular languages
- Language equivalence of NFA, DFA and RE
- Pumping lemma

Introduction

- Regular languages are efficient but very limited in power
- Can we still come up with a recognizer to express non-regular languages?
- The topic of today's lecture.

Table of Content

Context-free Grammars

Definition

A context-free grammar (CFG) G is a quadruple $\langle V, \Sigma, R, S \rangle$ where

- V: a set of non-terminal symbols
- Σ: a set of terminals ($V \cap \Sigma = \phi$)
- R: a set of production rules ($R : V \rightarrow (V \cup \Sigma)^*$)
- $S \in V$: a start symbol.

Context-free Grammars

Example

$$
\begin{aligned}
S &\rightarrow aSb \qquad\qquad\qquad or \;\; S \rightarrow aSb \mid \emptyset \\
S &\rightarrow \emptyset
\end{aligned}
$$

- $V = \{S\}$
- $\Sigma = \{a, b\}$
- $R = \{S \rightarrow aSb, S \rightarrow \emptyset\}$
- The left-hand-side of a production rule (e.g. S) is called head, and the right-hand-side (e.g. aSb) is called body
- S is the start symbol.
- Convention: We always use capital letters for non-terminals.

Context-free Grammars: Language

- Derivations, using productions from head to body.
- Let $A \to B$ be a production rule.
- If we apply this rule on a string xAy, then we obtain xBy. We say that xAy derivatives xBy and denote it by $xAy \to xBy$.
- In each step, we rewrite one symbol
- If $A_1 \to A_2 \to \dots \to A_n$, then we write $A_1 \to^* A_n$.

Example 1

$$S \rightarrow aSb \mid \emptyset$$

- Does S derivative $a^3 b^3$?

$$
\begin{aligned}
S \quad &\rightarrow aSb \\
&\rightarrow aaSbb \\
&\rightarrow aaaSbbb \\
&\rightarrow aaabbb
\end{aligned}
$$

Example 2

Let the grammar $G = (\{E, F\}, \{+, (,), *, a, b, 0, 1\}, P, E)$ that generates some arithmetic expressions where P is the following set of productions. **Derive $(\mathbf{b1}) * \mathbf{a} + \mathbf{ba1}$.**

$$1 : E \rightarrow F$$
$$2 : E \rightarrow E + E$$
$$3 : E \rightarrow E * E$$
$$4 : E \rightarrow (E)$$
$$5 : F \rightarrow a$$
$$6 : F \rightarrow b$$
$$7 : F \rightarrow Fa$$
$$8 : F \rightarrow Fb$$
$$9 : F \rightarrow F0$$
$$10 : F \rightarrow F1$$

Example 2

Let the grammar $G = (\{E, F\}, \{+, (,), *, a, b, 0, 1\}, P, E)$ that generates some arithmetic expressions where P is the following set of productions.
Derive $(\mathbf{b1}) * \mathbf{a} + \mathbf{ba1}.$

$$
\begin{aligned}
&E &&\rightarrow E * \mathbf{E} &&(3)\\
&1 : E \rightarrow F &&\rightarrow \mathbf{E} * E + E &&(2)\\
&2 : E \rightarrow E + E &&\rightarrow (\mathbf{E}) * E + E &&(4)\\
&3 : E \rightarrow E * E &&\rightarrow (\mathbf{F}) * E + E &&(1)\\
&4 : E \rightarrow (E) &&\rightarrow (\mathbf{F}1) * E + E &&(10)\\
&5 : F \rightarrow a &&\rightarrow (b1) * \mathbf{E} + E &&(6)\\
&6 : F \rightarrow b &&\rightarrow (b1) * F + \mathbf{E} &&(1)\\
&7 : F \rightarrow Fa &&\rightarrow (b1) * \mathbf{F} + F &&(1)\\
&8 : F \rightarrow Fb &&\rightarrow (b1) * a + \mathbf{F} &&(5)\\
&9 : F \rightarrow F0 &&\rightarrow (b1) * a + \mathbf{F}1 &&(10)\\
&10 : F \rightarrow F1 &&\rightarrow (b1) * a + \mathbf{F}a1 &&(7)\\
& &&\rightarrow (b1) * a + ba1 &&(6)
\end{aligned}
$$

Example 2

Let the grammar $G = (\{E, F\}, \{+, (,), *, a, b, 0, 1\}, P, E)$ that generates some arithmetic expressions where P is the following set of productions.
Derive $(\mathbf{b1}) * \mathbf{a} + \mathbf{ba1}$.

$$
\begin{aligned}
1 &: E \rightarrow F \\
2 &: E \rightarrow E + E \\
3 &: E \rightarrow E * E \\
4 &: E \rightarrow (E) \\
5 &: F \rightarrow a \\
6 &: F \rightarrow b \\
7 &: F \rightarrow Fa \\
8 &: F \rightarrow Fb \\
9 &: F \rightarrow F0 \\
10 &: F \rightarrow F1
\end{aligned}
$$

$$
\begin{aligned}
E \quad &\rightarrow E + \mathbf{E} & (2) \\
&\rightarrow E + \mathbf{F} & (5) \\
&\rightarrow E + \mathbf{F}1 & (10) \\
&\rightarrow E + \mathbf{F}a1 & (7) \\
&\rightarrow \mathbf{E} + ba1 & (6) \\
&\rightarrow \mathbf{E} * E + ba1 & (2) \\
&\rightarrow (\mathbf{E}) * E + ba1 & (4) \\
&\rightarrow (\mathbf{F}) * E + ba1 & (1) \\
&\rightarrow (\mathbf{F}1) * E + ba1 & (10) \\
&\rightarrow (b1) * \mathbf{E} + ba1 & (6) \\
&\rightarrow (b1) * \mathbf{F} + ba1 & (1) \\
&\rightarrow (b1) * a + ba1 & (5)
\end{aligned}
$$

Derivations

- At each step we might have several applicable rules,
- Not all choices lead to successful derivations of a particular string
- Leftmost derivation: always replace the leftmost variable by one of its rule-bodies.
- Rightmost derivation: always replace the rightmost variable by one of its rule

Table of Content

Context-Free Language (CFL)

- A string s is generated by a grammar $G = \langle V, \Sigma, R, S \rangle$, if $S \to^* s$.
- The languages of G is $L(G) = \{s \in \Sigma^* | S \to^* s\}$.
- The languages generated by a context-free grammar is called a context-free language, or,
- A language L' is context-free, if and only if there is a grammar G such that $L(G) = L'$

Example

Example

- Consider the following grammar:

$$S \rightarrow abSc \mid c$$

- The derivations are of the following forms:

  $S \rightarrow c$

  or

  $S \rightarrow abSc \rightarrow ababScc \rightarrow abababSccc \rightarrow ababababScccc \rightarrow$
  $.... \rightarrow (ab)^n c(c)^n$

- The language is strings with the pattern $(ab)^n c(c)^n$ where $n \geq 0$

$$L(G) = \{s | s = (ab)^n c(c)^n, n \geq 0\}$$

Example

Example

- Consider the following grammar:

$$S \rightarrow aSc \mid A$$
$$A \rightarrow bA \mid d$$

- The derivations are of the following forms:

$$S \rightarrow A \rightarrow bA \rightarrow bbA \rightarrow \dots \rightarrow b^n A \rightarrow b^n d$$

$$\text{or} \quad S \rightarrow A \rightarrow d$$

$$\text{or} \quad S \rightarrow aSc \rightarrow aaScc \rightarrow \dots \rightarrow a^m Sc^m \rightarrow a^m Ac^m \rightarrow a^m dc^m$$

$$\text{or} \quad S \rightarrow aSc \rightarrow aaScc \rightarrow \dots \rightarrow a^m Sc^m \rightarrow a^m Ac^m \rightarrow a^m bAc^m$$
$$\dots \rightarrow a^m b^k Ac^m \rightarrow a^m b^k dc^m$$

$$L(G) = \{s \mid s = a^n b^m dc^n, m, n \geq 0\}$$

Table of Content

Parse Tree

- Graphical representation of derivations
- Non-terminal are the interior nodes
- Terminals are the leaves
- The yield of a parse tree is the string of leaves from left to right or the concatenation of its children's yields from left to right.
- A string *s* belongs to a language, if we can produce a parse tree for it with the root labeled by the start symbol

Parse Tree: Example

$$1 : E \rightarrow F$$
$$2 : E \rightarrow E + E$$
$$3 : E \rightarrow E * E$$
$$4 : E \rightarrow (E)$$
$$5 : F \rightarrow a$$
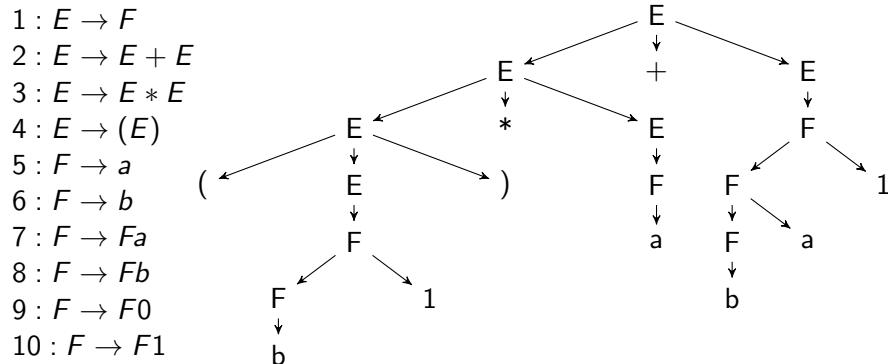$$6 : F \rightarrow b$$
$$7 : F \rightarrow Fa$$
$$8 : F \rightarrow Fb$$
$$9 : F \rightarrow F0$$
$$10 : F \rightarrow F1$$

Parse Tree: Example

The parse tree for $(b1) * a + ba1$

1 : $E \to F$

2 : $E \to E + E$

3 : $E \to E * E$

4 : $E \to (E)$

5 : $F \to a$

6 : $F \to b$

7 : $F \to Fa$

8 : $F \to Fb$

9 : $F \to F0$

10 : $F \to F1$

From Parse Tree to the String

Yields of a parse tree
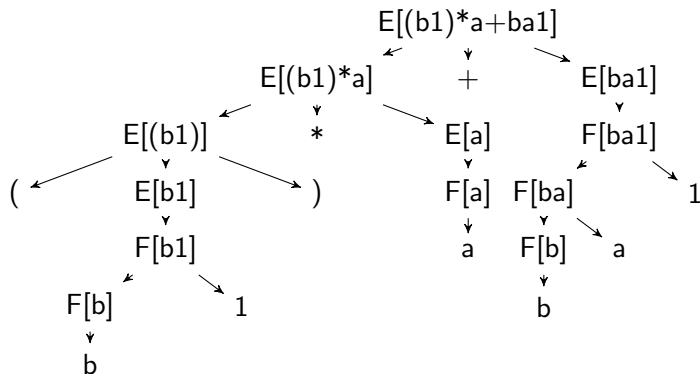
Table of Content

Language Ambiguity

- Consider the following grammar wit the start symbol S

$$S \rightarrow S + S \mid S * S \mid a \mid b$$

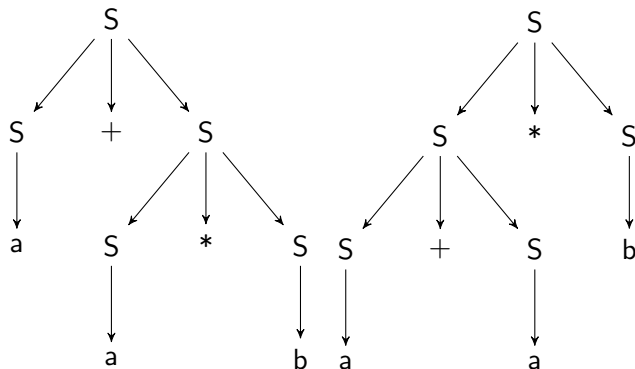- Derivative the string $a + a * b$

-
$$S \xrightarrow{1} S + S \xrightarrow{3} a + S \xrightarrow{2} a + S * S \xrightarrow{3} a + a * S \xrightarrow{4} a + a * b$$

Language Ambiguity

$$S \rightarrow S + S \mid S * S \mid a \mid b$$

Language Ambiguity

- A language is ambiguous, if it has a string with two different parse trees.
- No general algorithm to check ambiguity
- No general algorithm to resolve ambiguity
- An unambiguous grammar has unique left-most derivations
- There are some methods to remove ambiguity in **some** languages.

Removing Language Ambiguity: Left-factoring

- $$A \to \alpha\beta_1 | \alpha\beta_2 | \ldots | \alpha\beta_n | \omega_1 | \ldots | \omega_m$$

- ω is those rhs that does not start with α
- Rewrite it as the following where A' is non-terminal

$$A \to \alpha A' | \omega_1 | \ldots | \omega_m$$

$$A' \to \beta_1 | \beta_2 | \ldots | \beta_n$$

Removing Language Ambiguity: Left-factoring

Example

$$ifStmt \rightarrow IF\ boolExpr\ THEN\ stmts\ ELSE\ stmts$$

$$ifStmt \rightarrow IF\ boolExpr\ THEN\ stmts$$

can be rewritten as

$$ifStmt \rightarrow IF\ boolExpr\ THEN\ stmts\ elsePart$$

$$elsePart \rightarrow ELSE\ stmts\ |\ \epsilon$$

Removing Language Ambiguity: Left-Recursion

- $$A \rightarrow A\alpha \mid \omega_1 \mid \ldots \mid \omega_m$$

- Rewrite it as the following where A' is non-terminal

$$A \rightarrow \omega_1 A' \mid \ldots \mid \omega_m A'$$

$$A' \rightarrow \alpha A' \mid \epsilon$$

Example

$$E \rightarrow x \mid y \mid z \mid (E) \mid E + E$$

$x + y + z$ has more than one parse tree

Removing Language Ambiguity: Left-Recursion

- Indirect recursion, e.g.

$$A \rightarrow Bw \mid x$$

$$B \rightarrow Ay \mid z$$

after two steps of rewriting

$$A \rightarrow Bw \mid Ayw$$

Table of Content

Conclusions

- Context-free grammars and languages
- Parse trees and language ambiguity
- Next lecture is on CFGs vs REs and parsing