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 ($R : V \to (V \cup \Sigma)^*$)
- $S \in V$: a start symbol.

Context-free Grammars

Example

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

- $V = \{S\}$
- $\Sigma = \{a, b\}$
- $R = \{S \rightarrow aSb, S \rightarrow \emptyset\}$
- 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 $A_1 \to A_2 \to .... \to A_n$, then we write $A_1 \to^* A_n$.

Example 1

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

- Does S derivative a^3b^3?

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

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 $\mathbf{(b1) * a + ba1}$.**

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

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 $(\mathbf{b1}) * \mathbf{a} + \mathbf{ba1}$.**

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

$$
\begin{aligned}
E &\to E * \mathbf{E} && (3) \\
&\to \mathbf{E} * E + E && (2) \\
&\to (\mathbf{E}) * E + E && (4) \\
&\to (\mathbf{F}) * E + E && (1) \\
&\to (\mathbf{F}1) * E + E && (10) \\
&\to (b1) * \mathbf{E} + E && (6) \\
&\to (b1) * F + \mathbf{E} && (1) \\
&\to (b1) * \mathbf{F} + F && (1) \\
&\to (b1) * a + \mathbf{F} && (5) \\
&\to (b1) * a + \mathbf{F}1 && (10) \\
&\to (b1) * a + \mathbf{F}a1 && (7) \\
&\to (b1) * a + ba1 && (6)
\end{aligned}
$$

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 $\mathbf{(b1) * a + ba1}$.**

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

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

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 $S \rightarrow^* s$.
- The languages of G is $L(G) = \{s \in \Sigma^* | S \rightarrow^* s\}$.
- 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:

$$S \to abSc \mid c$$

- The derivations are of the following forms:

  $S \to c$
  or
  $S \to abSc \to ababScc \to abababSccc \to ababababScccc \to \ldots. \to (ab)^n c(c)^n$

- 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:

$$\begin{aligned} S &\to aSc \mid A \\ A &\to bA \mid d \end{aligned}$$

- The derivations are of the following forms:

$$S \to A \to bA \to bbA \to .... \to b^n A \to b^n d$$

$$or \qquad S \to A \to d$$

$$or \qquad S \to aSc \to aaScc \to .... \to a^m Sc^m \to a^m Ac^m \to a^m dc^m$$

$$or \qquad \begin{array}{l} S \to aSc \to aaScc \to .... \to a^m Sc^m \to a^m Ac^m \to a^m bAc^m \\ .... \to a^m b^k Ac^m \to a^m b^k dc^m \end{array}$$

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

Example 3

Example

- **BonusPoint** What is the language of the following grammar? S is the start symbol.

$$S \to A \mid cS \mid a \mid b$$
$$A \to aSb \mid baS \mid abS \mid Sab \mid bSa \mid Sba$$

where $\Sigma = \{a, b, c\}$

Table of Content

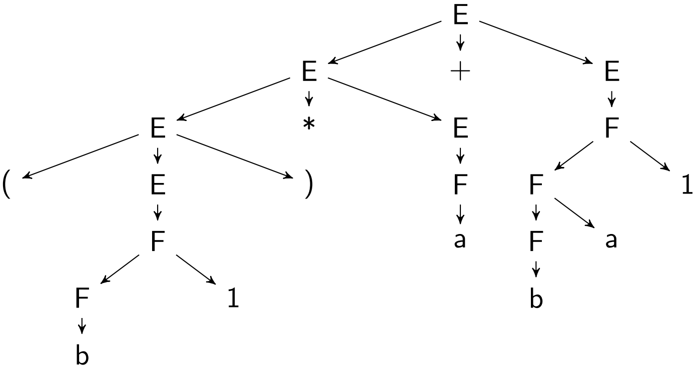
Parse Tree: Example

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

Parse Tree: Example

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

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



Parse Tree

- Graphical representation of derivations
- Non-terminal are the interior nodes
- Terminals are the leaves
- A string s belongs to a language, if we can produce a parse tree for it with the root labeled by the start symbol
- 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.

From Parse Tree to the String

Yields of a parse tree

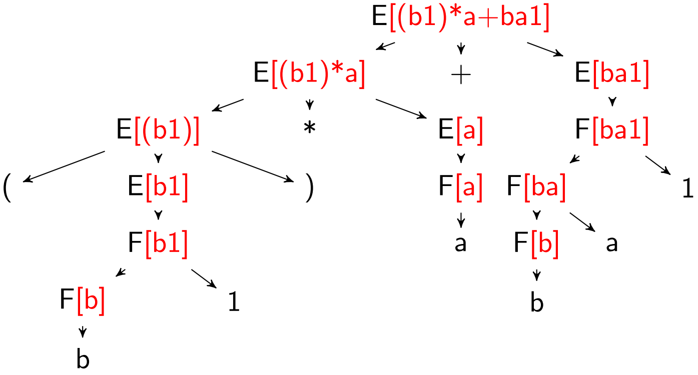


Table of Content

Language Ambiguity

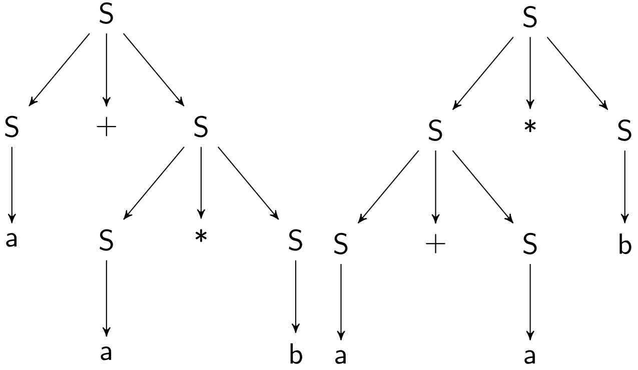
- Consider the following grammar with the start symbol S

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

- 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

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



Language Ambiguity

- A language is ambiguous, if it has a string with two different parse trees.
- **BonusPoint** Is the following statement correct?
  "if a language contains a string that can be obtained using two different derivations, it's ambiguous"
- 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

- $$A \rightarrow \alpha\beta_1|\alpha\beta_2|\ldots|\alpha\beta_n|\omega_1|\ldots|\omega_m$$
- ω is those rh's that do not start with α
- Rewrite it as the following where A' is non-terminal

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

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

Removing Language Ambiguity: Left-factoring

Example

$$S \rightarrow if \ E \ then \ S' \ else \ S'$$

$$S \rightarrow if \ E \ then \ S'$$

can be rewritten as

$$S \rightarrow if \ E \ then \ S' \ S''$$

$$S'' \rightarrow else \ S' \mid \epsilon$$

Removing Language Ambiguity: Left-Recursion

- $$A \to A\alpha \mid \omega_1 | \ldots | \omega_m$$
- Rewrite it as the following where A' is non-terminal

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

$$A' \to \alpha A' | \epsilon$$

Example

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

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

After removing left-recursion

$$E \to xE' \mid yE' \mid zE' \mid (E)E'$$

$$E' \to +EE' \mid \epsilon$$

Removing Language Ambiguity: Left-Recursion

BonusPoint Is the following grammar ambiguous? Why? If yes, remove ambiguity.

$$A \rightarrow Bw \mid x$$

$$B \rightarrow Ct|zC$$

$$C \rightarrow AxA \mid z$$

Removing Language Ambiguity: Indirect Left-Recursion

- Indirect recursion, e.g.

$$A \rightarrow Bw \mid x$$

$$B \rightarrow Ct|zC$$

$$C \rightarrow AxA \mid z$$

after rewriting and removing C

$$B \rightarrow AxAt \mid zt \mid zAxA \mid zz$$

after rewriting and removing B

$$A \rightarrow AxAtw \mid ztw \mid zAxAw \mid zzw \mid x$$

that has left recursion!

Removing Language Ambiguity: Indirect Left-Recursion

After removing left-recursion

$$A \to ztwA' \mid zAxAwA' \mid zzwA' \mid xA'$$

$$A' \to xAtwA' \mid \epsilon$$

Needs left-factoring z in the next step

$$A \to zZ \mid xA'$$

$$Z \to twA' \mid AxAwA' \mid zwA'$$

$$A' \to xAtwA' \mid \epsilon$$

Needs one further step rewriting to left-factorize z in the rules for Z (via A and the third rule)....

$$Z \to twA' \mid zZxAwA' \mid xA'xAwA' \mid zwA'$$

to

$$Z \to twA' \mid zZ' \mid xA'xAwA'$$

$$Z' \to ZxAwA' \mid wA'$$

Removing Language Ambiguity: Indirect Left-Recursion

Final grammar

$$A \to zZ \mid xA'$$

$$A' \to xAtwA' \mid \epsilon$$

$$Z \to twA' \mid zZ' \mid xA'xAwA'$$

$$Z' \to ZxAwA' \mid wA'$$

Table of Content

Conclusions

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