

CS420

Introduction to the Theory of Computation

Lecture 15: Context-free grammars

Tiago Cogumbreiro

Today we will learn...

- Context-Free Language
- Context-Free Grammar (CFG)
- Derivation
- Parse tree
- Writing context-free grammars
- Left-most derivations
- Ambiguous grammars

Section 2.1

Context-free grammars

Why do we use/need them?

Context-free grammars

- Appear in the context of natural languages
- Allows the formalization of a syntactic structure of terms
- Context-free grammars introduce recursive definition
- Context-free grammars are widely used in the specification of protocols, file formats, compilers, and interpreters

Use-case

Parsing JSON

Grammar for JSON

ANTLR is a **parser generator**.

- **Input:** a *grammar*; **Output:** a parser, and data-structures that represent the parse tree (known as a Concrete Syntax Tree)
- The HTML DOM is an example of an **Abstract** Syntax Tree

```

json: value; // initial rule
obj: '{' pair (',' pair)* '}' | '{' '}' ; // a sequence of comma-separated pairs
pair: STRING ':' value; // Example: "foo": 1
array: '[' value (',' value)* ']' | '[' ']' ; // a sequence of comma-separated values
value: STRING | NUMBER | obj | array | 'true' | 'false' | 'null';
// ...
  
```

Source: raw.githubusercontent.com/antlr/grammars-v4/master/json/JSON.g4

A grammar for JSON integers

```
NUMBER: '-'? INT ('.' [0-9] +)? EXP?;
```

```
fragment INT: '0' | [1-9] [0-9]*; // fragment means do not generate code for this rule
```

```
fragment EXP : [Ee] [+|-]? INT; // fragment means do not generate code for this rule
```

Source: raw.githubusercontent.com/antlr/grammars-v4/master/json/JSON.g4

A grammar for JSON

```

> ls *.java
JSONBaseListener.java JSONParser.java JSONVisitor.java
JSONBaseVisitor.java JSONLexer.java JSONListener.java
> cat JSONBaseListener.java
// Generated from ../JSON.g4 by ANTLR 4.7.2
import org.antlr.v4.runtime.tree.ParseTreeListener;

/**
 * This interface defines a complete listener for a parse tree produced by
 * {@link JSONParser}.
 */
public interface JSONListener extends ParseTreeListener {
    /**
     * Enter a parse tree produced by {@link JSONParser#json}.
     * @param ctx the parse tree
     */
    void enterJson(JSONParser.JsonContext ctx);
    /**
     * Exit a parse tree produced by {@link JSONParser#json}.
     * @param ctx the parse tree
     */
    void exitJson(JSONParser.JsonContext ctx);

```


An example of a context-free grammar

An example of a context-free grammar

Example grammar

- A boolean expression B can be either an and-operation, an or-operation, or a boolean literal.
- A boolean literal is either t or f

$$B \rightarrow B \text{ and } B$$

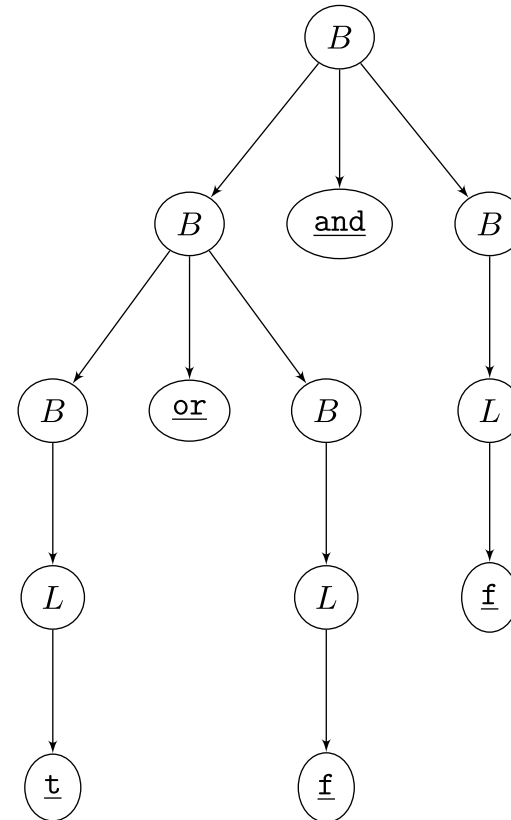
$$B \rightarrow B \text{ or } B$$

$$B \rightarrow L$$

$$L \rightarrow \text{t}$$

$$L \rightarrow \text{f}$$

Example: t or f and f



What is a grammar?

Grammar

- **Format:** A grammar G consists of a sequence of productions.
- **Start variable:** Every grammar has exactly one start variable. By **convention** the start variable is the first variable in the right-hand side of the first production.

Examples

Let grammar G consist of the following 5 productions:

- Production #1: $B \rightarrow B$ and B
- Production #2: $B \rightarrow B$ or B
- Production #3: $B \rightarrow L$
- Production #4: $L \rightarrow \mathbf{t}$
- Production #5: $L \rightarrow \mathbf{f}$

Productions

- **Also Known As:** substitution rule, or just a rule.
- **Format:** a **variable**, say A , followed by an arrow \rightarrow , and then a possibly-empty sequence of **terminals** / variables
- **Starts from:** A production **starts from** the variable on the left-hand side of the production. Example, production $B \rightarrow L$ starts from B (and not from L)
- **Variables** a symbol distinguished by *an italic font*, often capital letters. Examples: B or L .
- **Terminals** a symbol distinguished by a **mono** type font, often lower-case letters / numbers

Example

$$\underbrace{B}_{\text{variable}} \rightarrow \underbrace{B}_{\text{var.}} \text{ and } \underbrace{B}_{\text{var.}}$$

Generating strings

Generating strings

Yield $u \Rightarrow v$

Operation yield, given a string in the form $u\underline{A}v$ returns $u\underline{w}v$ if there is some rule $A \rightarrow w$ in the grammar.

Example

Grammar

$$B \rightarrow B \text{ and } B \text{ (1)}$$

$$B \rightarrow B \text{ or } B$$

$$B \rightarrow L \text{ (2), (4)}$$

$$L \rightarrow \mathbf{t} \text{ (3)}$$

$$L \rightarrow \mathbf{f} \text{ (5)}$$

Derivation

$$B \Rightarrow B \text{ and } B$$



Example

Grammar

$$\begin{aligned}
 B &\rightarrow B \text{ and } B \text{ (1)} \\
 B &\rightarrow B \text{ or } B \\
 B &\rightarrow L \text{ (2), (4)} \\
 L &\rightarrow \mathbf{t} \text{ (3)} \\
 L &\rightarrow \mathbf{f} \text{ (5)}
 \end{aligned}$$

Derivation

$$\begin{aligned}
 &\underbrace{B \Rightarrow B}_{1} \text{ and } B \\
 &\underbrace{\Rightarrow}_{2} L \text{ and } B
 \end{aligned}$$

Example

Grammar

$$\begin{aligned}
 B &\rightarrow B \text{ and } B \text{ (1)} \\
 B &\rightarrow B \text{ or } B \\
 B &\rightarrow L \text{ (2), (4)} \\
 L &\rightarrow \mathbf{t} \text{ (3)} \\
 L &\rightarrow \mathbf{f} \text{ (5)}
 \end{aligned}$$

Derivation

$$\begin{aligned}
 & \underbrace{B \Rightarrow B}_{1} \text{ and } B \\
 & \underbrace{\Rightarrow}_{2} L \text{ and } B \\
 & \underbrace{\Rightarrow}_{3} \mathbf{t} \text{ and } B
 \end{aligned}$$

Example

Grammar

$$\begin{aligned}
 B &\rightarrow B \text{ and } B \text{ (1)} \\
 B &\rightarrow B \text{ or } B \\
 B &\rightarrow L \text{ (2), (4)} \\
 L &\rightarrow \mathbf{t} \text{ (3)} \\
 L &\rightarrow \mathbf{f} \text{ (5)}
 \end{aligned}$$

Derivation

$$\begin{aligned}
 &\underbrace{B \Rightarrow B}_{1} \text{ and } B \\
 &\underbrace{\Rightarrow}_{2} L \text{ and } B \\
 &\underbrace{\Rightarrow}_{3} \mathbf{t} \text{ and } B \\
 &\underbrace{\Rightarrow}_{4} \mathbf{t} \text{ and } L
 \end{aligned}$$

Example

Grammar

$$\begin{aligned}
 B &\rightarrow B \text{ and } B \text{ (1)} \\
 B &\rightarrow B \text{ or } B \\
 B &\rightarrow L \text{ (2), (4)} \\
 L &\rightarrow \mathbf{t} \text{ (3)} \\
 L &\rightarrow \mathbf{f} \text{ (5)}
 \end{aligned}$$

Derivation

$$\begin{aligned}
 & B \Rightarrow B \text{ and } B \\
 & \underbrace{\hspace{1.5cm}}_1 \\
 & \Rightarrow L \text{ and } B \\
 & \underbrace{\hspace{1.5cm}}_2 \\
 & \Rightarrow \mathbf{t} \text{ and } B \\
 & \underbrace{\hspace{1.5cm}}_3 \\
 & \Rightarrow \mathbf{t} \text{ and } L \\
 & \underbrace{\hspace{1.5cm}}_4 \\
 & \Rightarrow \mathbf{t} \text{ and } \mathbf{f} \\
 & \underbrace{\hspace{1.5cm}}_5
 \end{aligned}$$

Thus, $B \Rightarrow^* \mathbf{t} \text{ and } \mathbf{f}$

Example

Grammar that generates well-balanced braces.

$$C \rightarrow \{ C \}$$

$$C \rightarrow CC$$

$$C \rightarrow \epsilon$$

Derivation

Build a derivation for $\{\{\}\}\{\}$.

Example

Grammar that generates well-balanced braces.

$$C \rightarrow \{ C \}$$

$$C \rightarrow CC$$

$$C \rightarrow \epsilon$$

Derivation

Build a derivation for $\{\{\}\}\{\}$.

$$\underline{C} \Rightarrow \underline{C}C \Rightarrow \{C\}\underline{C} \Rightarrow \{\underline{C}\}\{C\} \Rightarrow \{\{\underline{C}\}\}\{C\} \Rightarrow \{\{\epsilon\}\}\{\underline{C}\} \Rightarrow \{\{\}\}\{\}$$

Shorthand notation For grammars

Shorthand notation

Instead of writing $A \rightarrow w_1, \dots, A \rightarrow w_n$ can be **abbreviated** as $A \rightarrow w_1 \mid \dots \mid w_n$.

Example

$$C \rightarrow \{ C \}$$

$$C \rightarrow CC$$

$$C \rightarrow \epsilon$$

can be abbreviated as

$$C \rightarrow \{ C \} \mid CC \mid \epsilon$$

Example

Build a grammar from a regex.

Write a CFG that recognizes $L(10^*1)$.

Example

Build a grammar from a regex.

Write a CFG that recognizes $L(10^*1)$.

$$C \rightarrow 1D$$

$$D \rightarrow 0D \mid E$$

$$E \rightarrow 1$$

Example

Write a CFG that recognizes language $\{0^n 1^n \mid n \geq 0\}$.

Example

Write a CFG that recognizes language $\{0^n 1^n \mid n \geq 0\}$.

Solution

$$A \rightarrow 0A1$$
$$A \rightarrow \epsilon$$

Example

Write a CFG that recognizes language $\{0^n 1^m \mid n \leq m\}$.

Example

Write a CFG that recognizes language $\{0^n 1^m \mid n \leq m\}$.

Solution

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow 1B$$

$$B \rightarrow \epsilon$$

Parse tree examples

Parse tree examples

- CFG's may process a string in any order (not just from left-to-right)

Derive: $8 \div 2 \times 4$

Left-to-right derivation example.

$$E \rightarrow E \times E \mid E \div E \mid L$$

$$L \rightarrow 2 \mid 4 \mid 8$$

Derive: $8 \div 2 \times 4$

Left-to-right derivation example.

$$E \rightarrow E \times E \mid E \div E \mid L$$

$$L \rightarrow 2 \mid 4 \mid 8$$

Derivation $D_1: (8 \div 2) \times 4 = 16$

$$\begin{aligned} E &\Rightarrow \underline{E} \times E \\ &\Rightarrow \underline{E} \div E \times E \\ &\Rightarrow \underline{L} \div E \times E \\ &\Rightarrow 8 \div \underline{E} \times E \\ &\Rightarrow 8 \div \underline{L} \times E \\ &\Rightarrow 8 \div 2 \times \underline{E} \\ &\Rightarrow 8 \div 2 \times \underline{L} \\ &\Rightarrow 8 \div 2 \times 4 \end{aligned}$$

Derive: $8 \div 2 \times 4$

Left-to-right derivation example.

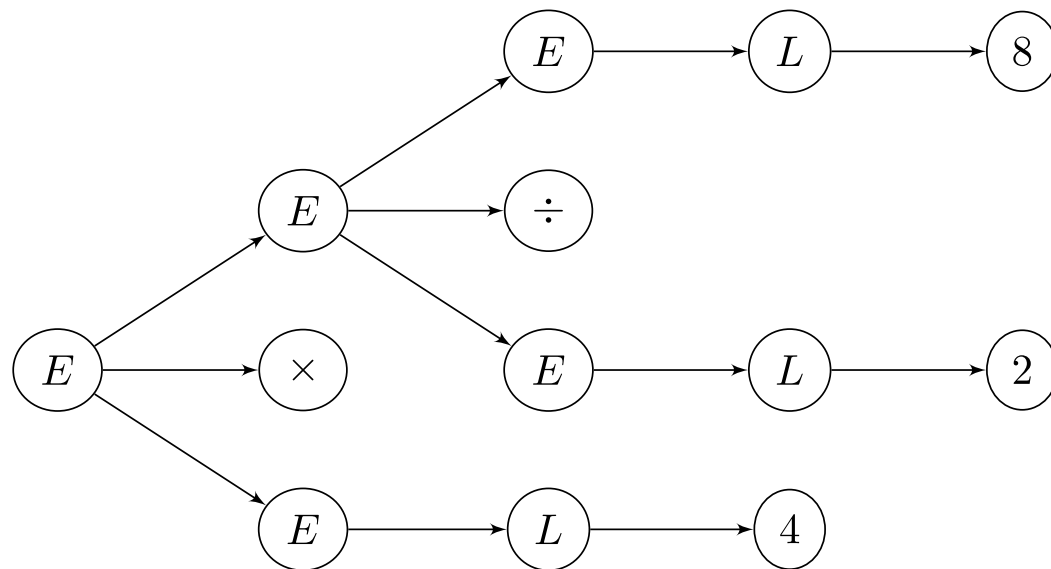
$$E \rightarrow E \times E \mid E \div E \mid L$$

$$L \rightarrow 2 \mid 4 \mid 8$$

Derivation $D_1: (8 \div 2) \times 4 = 16$

$$\begin{aligned} E &\Rightarrow \underline{E} \times E \\ &\Rightarrow \underline{E} \div E \times E \\ &\Rightarrow \underline{L} \div E \times E \\ &\Rightarrow 8 \div \underline{E} \times E \\ &\Rightarrow 8 \div \underline{L} \times E \\ &\Rightarrow 8 \div 2 \times \underline{E} \\ &\Rightarrow 8 \div 2 \times \underline{L} \\ &\Rightarrow 8 \div 2 \times 4 \end{aligned}$$

Parse Tree



Derive: $8 \div 2 \times 4$

Right-to-left derivation example.

$$E \rightarrow E \times E \mid E \div E \mid L$$

$$L \rightarrow 2 \mid 4 \mid 8$$

Derive: $8 \div 2 \times 4$

Right-to-left derivation example.

$$E \rightarrow E \times E \mid E \div E \mid L$$

$$L \rightarrow 2 \mid 4 \mid 8$$

Derivation D_2 : $8 \div (2 \times 4) = 1$

$$\begin{aligned} \underline{E} &\Rightarrow E \div \underline{E} \\ \Rightarrow \underline{E} \div E \times E \\ \Rightarrow \underline{L} \div E \times E \\ \Rightarrow 8 \div \underline{E} \times E \\ \Rightarrow 8 \div \underline{L} \times E \\ \Rightarrow 8 \div 2 \times \underline{E} \\ \Rightarrow 8 \div 2 \times \underline{L} \\ \Rightarrow 8 \div 2 \times 4 \end{aligned}$$

Derive: $8 \div 2 \times 4$

Right-to-left derivation example.

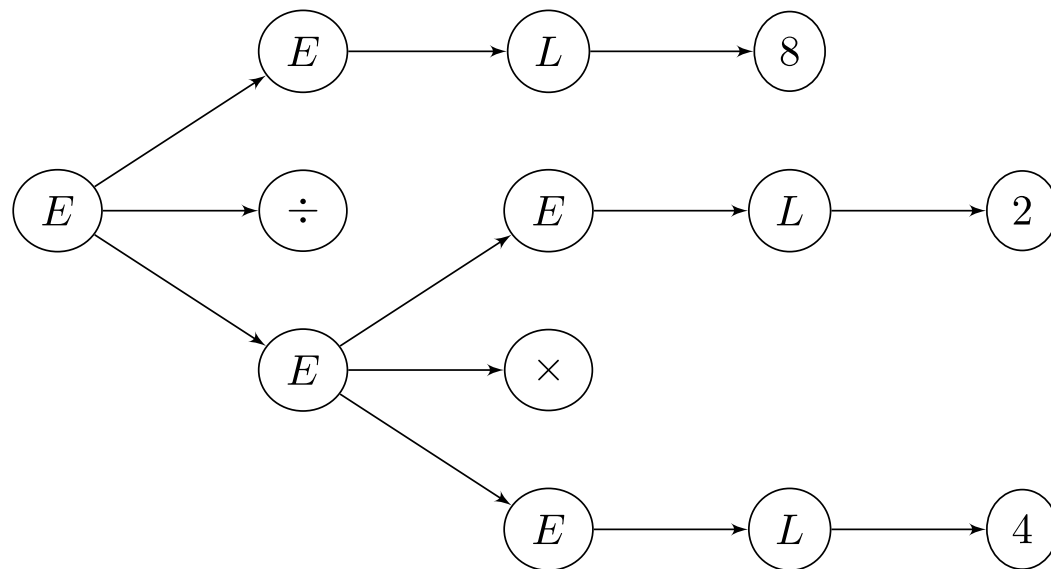
$$E \rightarrow E \times E \mid E \div E \mid L$$

$$L \rightarrow 2 \mid 4 \mid 8$$

Derivation $D_2: 8 \div (2 \times 4) = 1$

$$\begin{aligned} \underline{E} &\Rightarrow E \div \underline{E} \\ \Rightarrow \underline{E} \div E \times E \\ \Rightarrow \underline{L} \div E \times E \\ \Rightarrow 8 \div \underline{E} \times E \\ \Rightarrow 8 \div \underline{L} \times E \\ \Rightarrow 8 \div 2 \times \underline{E} \\ \Rightarrow 8 \div 2 \times \underline{L} \\ \Rightarrow 8 \div 2 \times 4 \end{aligned}$$

Parse Tree



Ambiguity

$$E \rightarrow E \times E \mid E \div E \mid L$$

$$L \rightarrow 2 \mid 4 \mid 8$$

Admits two different parse trees for the same string!

Formalizing CFGs

Context-free grammar

$$G = (V, \Sigma, R, S)$$

1. V is a finite set of **variables**
2. Σ is a finite set of **terminals**; Σ is disjoint from V
3. R is a set of rules $V \times V \cup \Sigma$
4. S is the **start variable**; $S \in V$

Generating strings

Yield

A string u yields a string v according to grammar G , notation $u \xRightarrow{G} v$, defined as follows. When there is no ambiguity we may omit the grammar and just write $u \Rightarrow v$.

$$\frac{A \rightarrow w \in R \quad G = (V, \Sigma, R, S)}{uAv \xRightarrow{G} uww}$$

Generating strings

Derivation

Since, \xRightarrow{G} is a binary relation, we call the reflexive transitive closure a **derivation**, notation $\xRightarrow{G^*}$, defined as follows:

$$\frac{u \xRightarrow{G^*} v \quad v \xRightarrow{G} w}{u \xRightarrow{G^*} w} \qquad \frac{}{u \xRightarrow{G^*} u}$$

Language of a CFG

Let $G = (V, \Sigma, R, S)$ be a context-free grammar. We define the language of G , notation $L(G)$ below.

$$L(G) = \{w \mid S \Rightarrow^* w\}$$

The language of a CFG consists of every word that can be derived from the start variable where all the letters are terminals.

Context-Free Language (CFL)

Definition. We say that a language L is context-free if there exists a CFG G such that $L(G) = L$

Ambiguity

Ambiguity

Note that we do not formalize parse trees, so we cannot define ambiguity in terms of a parse tree.

Definition

A **leftmost** derivation if at every step the leftmost remaining variable is the one replaced.

Definition 2.7

A string is derived **ambiguously** in context-free grammar G if it has two or more different leftmost derivations. Grammar G is ambiguous if it generates some string ambiguously.

Leftmost/non-leftmost example

Leftmost derivation

$$\begin{aligned}
 \underline{E} &\Rightarrow \underline{E} \times E \\
 \Rightarrow \underline{E} \div E \times E \\
 \Rightarrow \underline{L} \div E \times E \\
 \Rightarrow 8 \div \underline{E} \times E \\
 \Rightarrow 8 \div \underline{L} \times E \\
 \Rightarrow 8 \div 2 \times \underline{E} \\
 \Rightarrow 8 \div 2 \times \underline{L} \\
 \Rightarrow 8 \div 2 \times 4
 \end{aligned}$$

Non-leftmost derivation

$$\begin{aligned}
 \underline{E} &\Rightarrow E \div \underline{E} \\
 \Rightarrow \underline{E} \div E \times E \\
 \Rightarrow \underline{L} \div E \times E \\
 \Rightarrow 8 \div \underline{E} \times E \\
 \Rightarrow 8 \div \underline{L} \times E \\
 \Rightarrow 8 \div 2 \times \underline{E} \\
 \Rightarrow 8 \div 2 \times \underline{L} \\
 \Rightarrow 8 \div 2 \times 4
 \end{aligned}$$

Ambiguous grammar example

Claim: The grammar below is ambiguous.

$$E \rightarrow E \times E \mid E \div E \mid L$$

$$L \rightarrow 2 \mid 4 \mid 8$$

Can we convert D_2 into a leftmost derivation?

$$\begin{aligned} \underline{E} &\Rightarrow E \div \underline{E} \\ \Rightarrow \underline{E} \div E \times E \\ \Rightarrow \underline{L} \div E \times E \\ \Rightarrow 8 \div \underline{E} \times E \\ \Rightarrow 8 \div \underline{L} \times E \\ \Rightarrow 8 \div 2 \times \underline{E} \\ \Rightarrow 8 \div 2 \times \underline{L} \\ \Rightarrow 8 \div 2 \times 4 \end{aligned}$$

Ambiguous grammar example

Claim: The grammar below is ambiguous.

$$E \rightarrow E \times E \mid E \div E \mid L$$

$$L \rightarrow 2 \mid 4 \mid 8$$

Ambiguous grammar example

Claim: The grammar below is ambiguous.

$$E \rightarrow E \times E \mid E \div E \mid L$$

$$L \rightarrow 2 \mid 4 \mid 8$$

$$\begin{aligned}
 & (D_1) \\
 & \underline{E} \Rightarrow \underline{E} \times E \\
 \Rightarrow & \underline{E} \div E \times E \\
 \Rightarrow & \underline{L} \div E \times E \\
 \Rightarrow & 8 \div \underline{E} \times E \\
 \Rightarrow & 8 \div \underline{L} \times E \\
 \Rightarrow & 8 \div 2 \times \underline{E} \\
 \Rightarrow & 8 \div 2 \times \underline{L} \\
 \Rightarrow & 8 \div 2 \times 4
 \end{aligned}$$

$$\begin{aligned}
 & (D'_2) \\
 & \underline{E} \Rightarrow \underline{E} \div E \\
 \Rightarrow & \underline{L} \div E \\
 \Rightarrow & 8 \div \underline{E} \\
 \Rightarrow & 8 \div \underline{E} \times E \\
 \Rightarrow & 8 \div \underline{L} \times E \\
 \Rightarrow & 8 \div 2 \times \underline{E} \\
 \Rightarrow & 8 \div 2 \times \underline{L} \\
 \Rightarrow & 8 \div 2 \times 4
 \end{aligned}$$

Proof. String $8 \div 2 \times 4$ is derived ambiguously, since there are at least two distinct leftmost derivation (see slides before).