CS420

Introduction to the Theory of Computation

Lecture 16: 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

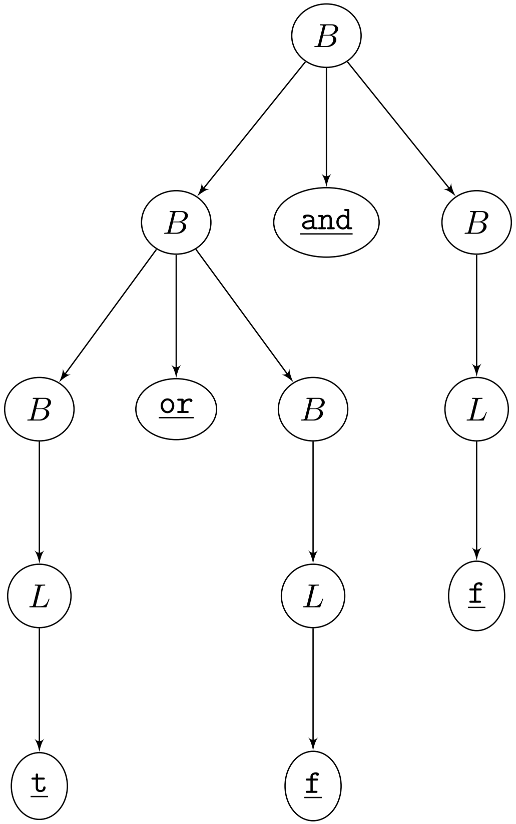
An example of a context-free grammar

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

$$B \rightarrow B \texttt{ and } B$$
$$B \rightarrow B \texttt{ or } B$$
$$B \rightarrow L$$
$$L \rightarrow \texttt{t}$$
$$L \rightarrow \texttt{f}$$

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: $B \to B\ \mathtt{and}\ B$
- Production #2: $B \to B\ \mathtt{or}\ B$
- Production #3: $B \to L$
- Production #4: $L \to \mathtt{t}$
- Production #5: $L \to \mathtt{f}$

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

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

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 $A \to w$ in the grammar.

Example

Grammar

$B \rightarrow B$ **and** B (1)

$B \rightarrow B$ **or** B

$B \rightarrow L$ (2), (4)

$L \rightarrow$ **t** (3)

$L \rightarrow$ **f** (5)

Derivation

$$\underbrace{B \Rightarrow}_{1} B \texttt{ and } B$$

Example

Grammar

$B \rightarrow B$ **and** B (1)

$B \rightarrow B$ **or** B

$B \rightarrow L$ (2), (4)

$L \rightarrow$ **t** (3)

$L \rightarrow$ **f** (5)

Derivation

$$B \underbrace{\Rightarrow}_{1} B \texttt{ and } B$$

$$\underbrace{\Rightarrow}_{2} L \texttt{ and } B$$

Example

Grammar

$B \rightarrow B$ **and** B (1)

$B \rightarrow B$ **or** B

$B \rightarrow L$ (2), (4)

$L \rightarrow$ **t** (3)

$L \rightarrow$ **f** (5)

Derivation

$$B \underbrace{\Rightarrow}_{1} B \texttt{ and } B$$

$$\underbrace{\Rightarrow}_{2} L \texttt{ and } B$$

$$\underbrace{\Rightarrow}_{3} \texttt{t and } B$$

Example

Grammar

$B \rightarrow B$ **and** B (1)

$B \rightarrow B$ **or** B

$B \rightarrow L$ (2), (4)

$L \rightarrow$ **t** (3)

$L \rightarrow$ **f** (5)

Derivation

$$B \underbrace{\Rightarrow}_{1} B \texttt{ and } B$$

$$\underbrace{\Rightarrow}_{2} L \texttt{ and } B$$

$$\underbrace{\Rightarrow}_{3} \texttt{t and } B$$

$$\underbrace{\Rightarrow}_{4} \texttt{t and } L$$

Example

Grammar

$B \rightarrow B$ `and` B (1)

$B \rightarrow B$ `or` B

$B \rightarrow L$ (2), (4)

$L \rightarrow$ `t` (3)

$L \rightarrow$ `f` (5)

Derivation

$$\underbrace{B \Rightarrow}_{1} B \texttt{ and } B$$

$$\underbrace{\Rightarrow}_{2} L \texttt{ and } B$$

$$\underbrace{\Rightarrow}_{3} \texttt{t and } B$$

$$\underbrace{\Rightarrow}_{4} \texttt{t and } L$$

$$\underbrace{\Rightarrow}_{5} \texttt{t and f}$$

Thus, $B \Rightarrow^{\star} \texttt{t and f}$

Example

Grammar that generates well-balanced braces.

$$C \to \{\ C\ \}$$
$$C \to CC$$
$$C \to \epsilon$$

Derivation

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

Example

Grammar that generates well-balanced braces.

$$C \to \{\ C\ \}$$
$$C \to CC$$
$$C \to \epsilon$$

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 $A \to w_1, \ldots, A \to w_n$ can be **abbreviated** as $A \to w_1 \mid \cdots \mid w_n$.

Example

$$C \to \{\ C\ \}$$
$$C \to CC$$
$$C \to \epsilon$$

can be abbreviated as

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

Example

Build a grammar from a regex.

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

Example

Build a grammar from a regex.

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

$$C \rightarrow 1D$$
$$D \rightarrow 0D \mid E$$
$$E \rightarrow 1$$

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 ÷ 2 × 4

Left-to-right derivation example.

$$E \to E \times E \mid E \div E \mid L$$
$$L \to 2 \mid 4 \mid 8$$

Derive: 8 ÷ 2 × 4

Left-to-right derivation example.

$$E \to E \times E \mid E \div E \mid L$$
$$L \to 2 \mid 4 \mid 8$$

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

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

Derive: 8 ÷ 2 × 4

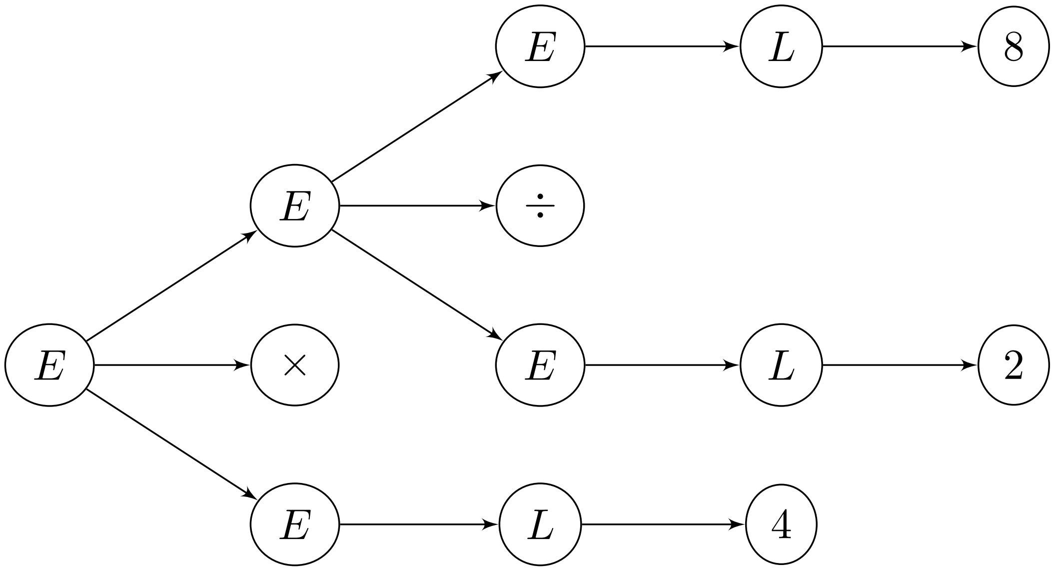
Left-to-right derivation example.

$$E \to E \times E \mid E \div E \mid L$$
$$L \to 2 \mid 4 \mid 8$$

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

$$\underline{E} \Rightarrow \underline{E} \times E$$
$$\Rightarrow \underline{E} \div E \times E$$
$$\Rightarrow \underline{L} \div E \times E$$
$$\Rightarrow 8 \div \underline{E} \times E$$
$$\Rightarrow 8 \div \underline{L} \times E$$
$$\Rightarrow 8 \div 2 \times \underline{E}$$
$$\Rightarrow 8 \div 2 \times \underline{L}$$
$$\Rightarrow 8 \div 2 \times 4$$

Parse Tree



Derive: 8 ÷ 2 × 4

Right-to-left derivation example.

$$E \to E \times E \mid E \div E \mid L$$
$$L \to 2 \mid 4 \mid 8$$

Derive: 8 ÷ 2 × 4

Right-to-left derivation example.

$$E \to E \times E \mid E \div E \mid L$$
$$L \to 2 \mid 4 \mid 8$$

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 ÷ 2 × 4

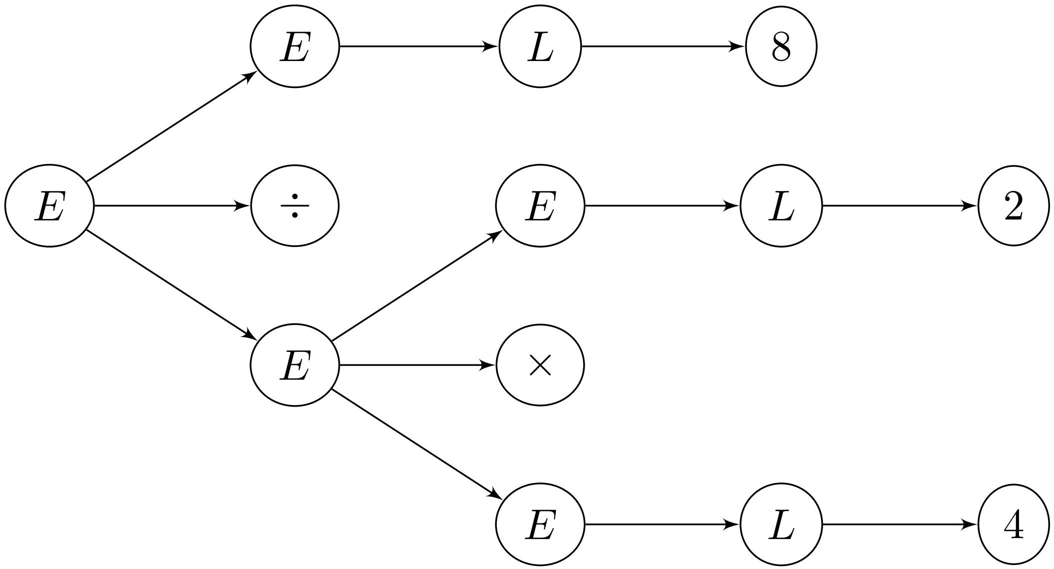
Right-to-left derivation example.

$$E \to E \times E \mid E \div E \mid L$$
$$L \to 2 \mid 4 \mid 8$$

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

$$\underline{E} \Rightarrow E \div \underline{E}$$
$$\Rightarrow \underline{E} \div E \times E$$
$$\Rightarrow \underline{L} \div E \times E$$
$$\Rightarrow 8 \div \underline{E} \times E$$
$$\Rightarrow 8 \div \underline{L} \times E$$
$$\Rightarrow 8 \div 2 \times \underline{E}$$
$$\Rightarrow 8 \div 2 \times \underline{L}$$
$$\Rightarrow 8 \div 2 \times 4$$

Parse Tree



Ambiguity

$$E \to E \times E \mid E \div E \mid L$$
$$L \to 2 \mid 4 \mid 8$$

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 $u \overset{G}{\Longrightarrow} v$, defined as follows. When there is no ambiguity we may omit the grammar and just write $u \Rightarrow v$.

$$\frac{A \rightarrow w \in R \qquad G = (V, \Sigma, R, S)}{uAv \overset{G}{\Longrightarrow} uwv}$$

Generating strings

Derivation

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

$$\frac{u \stackrel{G}{\Longrightarrow}^{\star} v \qquad v \stackrel{G}{\Longrightarrow} w}{u \stackrel{G}{\Longrightarrow}^{\star} w} \qquad \frac{}{u \stackrel{G}{\Longrightarrow}^{\star} u}$$

Language of a CFG

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

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

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.

$$E \to E \times E \mid E \div E \mid L$$
$$L \to 2 \mid 4 \mid 8$$

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.

$$E \to E \times E \mid E \div E \mid L$$
$$L \to 2 \mid 4 \mid 8$$

Ambiguous grammar example

Claim: The grammar below is ambiguous.

$$E \to E \times E \mid E \div E \mid L$$
$$L \to 2 \mid 4 \mid 8$$

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

$$(D_2')$$
$$\begin{aligned} \underline{E} &\Rightarrow \underline{E} \div E \\ &\Rightarrow \underline{L} \div E \\ &\Rightarrow 8 \div \underline{E} \\ &\Rightarrow 8 \div \underline{E} \times E \\ &\Rightarrow 8 \div \underline{L} \times E \\ &\Rightarrow 8 \div 2 \times \underline{E} \\ &\Rightarrow 8 \div 2 \times \underline{L} \\ &\Rightarrow 8 \div 2 \times 4 \end{aligned}$$

Proof. String `8 ÷ 2 × 4` is derived ambiguously, since there are at least two distinct leftmost derivation (see slides before).