CS112: Theory of Computation (LFA)

Lecture9: Context-free Grammars

Dumitru Bogdan

Faculty of Computer Science
University of Bucharest

April 15, 2021

Table of contents

Section 1

Previously on CS112

Intuition

- A grammar consists of a collection of **substitution rules**, also called **productions**
- Each rule appears as a line in the grammar, comprising a symbol and a string separated by an arrow
- The symbol is called a **variable**. The string consists of variables and other symbols called **terminals**
- The variable symbols often are represented by capital letters
- The terminals are analogous to the input alphabet and often are represented by lowercase letters, numbers, or special symbols
- One variable is designated as the start variable
- It usually occurs on the left-hand side of the topmost rule

Example

For example, grammar G_1 contains three rules. G_1's variables are A and B, where A is the start variable. Its terminals are 0, 1, and $\#$

$$A \rightarrow 0A1$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

Intuition

You use a grammar to describe a language by generating each string of that language in the following manner:

1. Write down the start variable. It is the variable on the left-hand side of the top rule, unless specified otherwise
2. Find a variable that is written down and a rule that starts with that variable. Replace the written down variable with the right-hand side of that rule
3. Repeat step 2 until no variables remain

Example

For example, grammar G_1 generates the string $000\#111$. The sequence of substitutions to obtain a string is called a **derivation**. A derivation of string $000\#111$ in grammar G_1 is:

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$$

Example

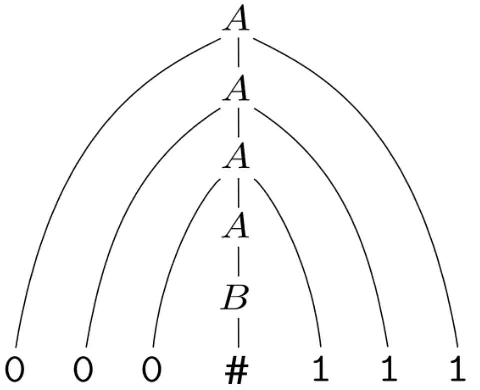

Figure: Parse tree for 000#111 in grammar G_1

Section 2

Context setup

Context setup

Corresponding to Sipser 2.1

Section 3

Ambiguity

Ambiguity

- Sometimes a grammar can generate the same string in several different ways
- Such a string will have several different parse trees and thus several different meanings
- This result may be undesirable for certain applications, such as programming languages, where a program should have a unique interpretation

Ambiguity

- If a grammar generates the same string in several different ways, we say that the string is derived *ambiguously* in that grammar
- If a grammar generates some string ambiguously, we say that the grammar is *ambiguous*
- If we consider grammar G_5

$$< EXPR > \rightarrow < EXPR > + < EXPR > | < EXPR > x < EXPR > |(< EXPR >)|a$$

we observe that it generates the string $a + a \times a$ ambiguously

Ambiguity

Grammar G_5 doesn't capture the usual precedence relations and so may group the $+$ before the $\times$ or vice versa.

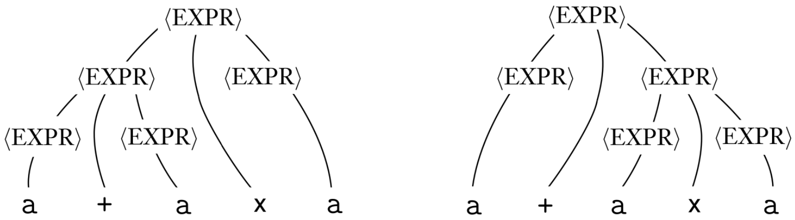

Figure: The two parse trees for the string $a + a \times a$ in grammar G_5

Ambiguity

- Grammar G_2 (see previous slides) is another example of an ambiguous grammar
- The sentence *the girl touches the boy with the flower* has two different derivations
- Give the two parse trees and observe their correspondence with the two different ways to read that sentence ($\Leftarrow$ get a CS112 T-Shirt)

Ambiguity

Now we formalize the notion of ambiguity

- When we say that a grammar generates a string ambiguously, we mean that the string has two different parse trees, not two different derivations
- Two derivations may differ merely in the order in which they replace variables yet not in their overall structure
- A derivation of a string w in a grammar G is a **leftmost derivation** if at every step the leftmost remaining variable is the one replaced

Ambiguity

Definition

A string w 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

- Sometimes when we have an ambiguous grammar we can find an unambiguous grammar that generates the same language
- Some context-free languages, however, can be generated only by ambiguous grammars. Such languages are called **inherently ambiguous**
- Study why $\{a^i b^j c^k | i = j or j = k\}$ is inherently ambiguous ($\Leftarrow$ and get a CS112 T-Shirt)

Section 4

Chomsky normal form

Noam Chomsky

Avram Noam Chomsky is an American linguist, **philosopher**, **political activist**, author, and lecturer. He is an Institute Professor and professor emeritus of linguistics at the Massachusetts Institute of Technology

- According to The New York Times, Noam Chomsky is **arguably the most important intellectual alive**
- https://www.goodreads.com/author/list/2476.Noam_Chomsky
- If in doubt, start with this book: **How the World Works**

Chomsky normal form

- Working with context-free grammars, it is often convenient to have them in simplified form
- The simplest and most useful forms is called **the Chomsky normal form**
- Chomsky normal form is useful in giving algorithms for working with context-free grammars

Formal definition

Definition

A context-free grammar is in Chomsky normal form if every rule is of the form

$$A \to BC$$
$$A \to a$$

where a is any terminal and A, B, and C are any variables — except that B and C may not be the start variable. In addition, we allow the rule $S \to \epsilon$, where S is the start variable

Converting to Chomsky normal form

Theorem

Any context-free language is generated by a context-free grammar in Chomsky normal form

- Proof idea is to convert any grammar G into Chomsky normal form
- The conversion has several stages wherein rules that violate the conditions are replaced with equivalent ones that are satisfactory

Proof idea

- First, we add a new start variable
- Then, we eliminate all ϵ-rules of the form $A \to \epsilon$
- We also eliminate all unit rules of the form $A \to B$
- In both cases we patch up the grammar to be sure that it still generates the same language
- Finally, we convert the remaining rules into the proper form

Formal proof

Proof.

First, we add a new start variable S_0 and the rule $S_0 \rightarrow S$, where S was the original start variable. This change guarantees that the start variable doesn't occur on the right-hand side of a rule

Second, we take care of all ϵ-rules. We remove an ϵ-rule $A \rightarrow \epsilon$, where A is not the start variable. Then for each occurrence of an A on the right-hand side of a rule, we add a new rule with that occurrence deleted. In other words, if $R \rightarrow uAv$ is a rule in which u and v are strings of variables and terminals, we add rule $R \rightarrow uv$. We do so for each occurrence of an A, so the rule $R \rightarrow uAvAw$ causes us to add $R \rightarrow uvAw$, $R \rightarrow uAvw$ and $R \rightarrow uvw$. If we have the rule $R \rightarrow A$, we add $R \rightarrow \epsilon$ unless we had previously removed the rule $R \rightarrow \epsilon$. We repeat these steps until we eliminate all ϵ-rules not involving the start variable.

Formal proof

Proof.

Third, we handle all unit rules. We remove a unit rule $A \to B$. Then, whenever a rule $B \to u$ appears, we add the rule $A \to u$ unless this was a unit rule previously removed. As before, u is a string of variables and terminals. We repeat these steps until we eliminate all unit rules.
Finally, we convert all remaining rules into the proper form. We replace each rule $A \to u_1 u_2 \dots u_k$, where $k \geq 3$ and each u_i is a variable or terminal symbol, with the rules $A \to u_1 A_1, A_1 \to u_2 A_2, A_2 \to u_3 A_3, \dots, A_{k-2} \to u_{k-1} u_k$ The A_i's are new variables. We replace any terminal u_i in the preceding rule(s) with the new variable U_i and add the rule $U_i \to u_i$ □

Section 5

Examples

Example

- Let G_6 be the following CFG and convert it to Chomsky normal form by using the conversion procedure just given.
- The series of grammars presented illustrates the steps in the conversion.

$$
\begin{aligned}
S &\to ASA|aB \\
A &\to B|S \\
B &\to b|\epsilon
\end{aligned}
$$

Example

1. The result of applying the first step to make a new start variable appears on the right

$$
\begin{aligned}
S &\rightarrow ASA|aB \\
A &\rightarrow B|S \\
B &\rightarrow b|\epsilon
\end{aligned}
\qquad
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASA|aB \\
A &\rightarrow B|S \\
B &\rightarrow b|\epsilon
\end{aligned}
$$

2. Remove ϵ-rules $B \rightarrow \epsilon$, shown on the left and $A \rightarrow \epsilon$, shown on the right

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASA|aB|a \\
A &\rightarrow B|S|\epsilon \\
B &\rightarrow b|\epsilon
\end{aligned}
\qquad
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASA|aB|a|SA|AS|S \\
A &\rightarrow B|S \\
B &\rightarrow b
\end{aligned}
$$

Example

3. Remove unit rules $S \to S$, shown on the left and $S_0 \to S$, shown on the right

$$S_0 \to S$$
$$S \to ASA|aB|a|SA|AS$$
$$A \to B|S$$
$$B \to b$$

$$S_0 \to ASA|aB|a|SA|AS$$
$$S \to ASA|aB|a|SA|AS$$
$$A \to B|S$$
$$B \to b$$

4. Remove unit rules $A \to B$ and $A \to S$

$$S_0 \to ASA|aB|a|SA|AS$$
$$S \to ASA|aB|a|SA|AS$$
$$A \to S|b$$
$$B \to b$$

$$S_0 \to ASA|aB|a|SA|AS$$
$$S \to ASA|aB|a|SA|AS$$
$$A \to b|ASA|aB|a|SA|AS$$
$$B \to b$$

Example

5. Convert the remaining rules into the proper form by adding additional variables and rules. The final grammar in Chomsky normal form is equivalent to G_6. We simplified the resulting grammar by using a single variable U and rule $U \to a$

$$
\begin{aligned}
S_0 &\to AA_1 | UB | a | SA | AS \\
S &\to AA_1 | UB | a | SA | AS \\
A &\to b | AA_1 | UB | a | SA | AS \\
A_1 &\to SA \\
U &\to a \\
B &\to b
\end{aligned}
$$