Alex Hubers
CSC:341 Spring 2024

Lecture Notes #4B

#1 Context-free grammars

Today we will be studying context-free grammars. Grammars can be used to specify both (i) languages, that is, sets of strings, and (ii) languages such as regular expressions, that is, "programming languages". Actually, (ii) is just a special case of (i), but there are practical distinction between the two. Grammars can also describe a billion other things. In fact, grammars have their origin in linguistics, hence the name. So they can generate natural language, too. (This is what Noam Chomsky studied before he switched to politics.) Here is a simple grammar:

$$\begin{aligned} A &\to 0A1 \mid B \\ B &\to \# \end{aligned}$$

We have (formally) four components.

Definition 1 (Context-Free Grammars)**.** *A context-free grammar is a 4-tuple (V, Σ, R, S) where*

1. *V is a finite set of symbols declared on the LHS, called variables or non-terminals;*
2. *Σ is a finite set of symbols that are terminal (they don't incur further substitutions);*
3. *$S \in V$ is designated start symbol; and*
4. *R is list of "rules" (above) that specify substitutions, called substitution or production rules.*

In the example above:

1. $V = \{A, B\}$,
2. $\Sigma = \{0, 1, \#\}$,
3. R is the set $\{A \to 0A1, A \to B, B \to \#\}$, and
4. S is the symbol A.

What is context-free about context-free grammars? Context-free grammars are named as such because any of the production rules in the grammar can be applied regardless of context—it does not depend on any other symbols that may or may not be around a given symbol that is having a rule applied to it. In other words, the left-hand side of each production rule is just a symbol in V. A production rule like $ABA \to 01A$ would not be context-free.

#2 Derivations

A "proof" that a given string is generated by a grammar is called a derivation. Here is an derivation showing that ABA generates $0\#1$, written $ABA \Rightarrow^* 0\#1$.

$$ABA \to 0A1 \to 0\#1$$

See that each arrow takes a "step" by applying a substitution rule until we get to the target string of terminals.

#3 Parse trees and ambiguity

Context-free grammars can also be thought of as defining trees. In PL, these are called abstract syntax trees. For example, arithmetic strings have a tree-like structure (which is why you in fact can represent arithmetic languages as trees in compilers.)

Consider a grammar for strings of arithmetic expressions.

$$\begin{aligned} E &\to E + E \mid E \times E \mid E \mid T \\ T &\to 0 \mid 1 \end{aligned}$$

In class I will ask for some example strings and then we will draw trees together.

This simple grammar can generate ambiguous strings.

Now, consider the string $1 + 1 \times 0$. Note that the meaning here is ambiguous: this could be either

$$(1 + 1) \times 0$$

which equals 0, or

$$1 + (1 \times 0)$$

which equals 1. I will draw trees of each in class (Or, see Figure 2.6 in Sipser).

$\langle$ Drawing goes here :) $\rangle$

Because this string corresponds to two parse trees, it is called ambiguous. More formally, we call a string ambiguous if it has two or more different leftmost derivations.

Definition 2 (Leftmost Derivation)**.** *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.*

Here is an example showing the example above is ambiguous.

For $(1 + 1) \times 0$, we have

$$E \to E \times E \to T + T \times E \to 1 + T \times E \to 1 + 1 \times E \to 1 + 1 \times T \to 1 + 1 \times 0$$

and for $1 + (1 \times 0)$ we have

$$E \to E + E \to T + E \to 1 + E \to 1 + E \times E \to 1 + 1 \times E \to 1 + 1 \times T \to 1 + 1 \times 0$$

Parse trees correspond to leftmost derivations—that is, if a string has two distinct leftmost derivations, those derivations will each have differing parse trees. It is often simpler thus to think of the parse trees as differing.

#4 Fixing ambiguity

Let's try to make arithmetic strings unamibugous.

$$
\begin{aligned}
E &\to E + T \mid T \\
T &\to T \times F \mid F \\
F &\to (E) \mid 0 \mid 1
\end{aligned}
$$

where the E is the start symbol.

See that our hand is forced: we must first transform E into either $+$ or T. Once it is T, we may turn it to multiplication. Further, we have parentheses to disambiguate the order of operations once we hit the variable F.

For $(1+1) \times 0$, we have the leftmost derivation

$$E \to T \to T \times F \to (E) \times F \to (E + T) \times F \to (T + T) \times F \to^2 (F + F) \times F \to^3 (1+1) \times 0$$

and for $1 + (1 \times 0)$ we have:

$$E \to E + T \to^2 1 + T \to 1 + (T \times F) \to 1 + (F \times F) \to^2 1 + (1 \times 0)$$

N.b. the notation $X \to^n Y$ is shorthand for "X steps to Y in n steps". I am just skipping some of those steps when I feel they are obvious. For example, I would rather write $(F + F) + F \to^3 (1 + 1) + 0$ than

$$(F + F) + F \to (1 + F) + F \to (1 + 1) + F \to (1 + 1) + 0$$