CSE396 Lecture Thu. Week 6: Context-Free Grammars [new material not on Prelim I]

These came out in papers by Noam Chomsky and then in his 1957 book *Syntactic Structures*. Let's have a blast of syntax before we see examples and larger motivations.

Definition: A **context-free grammar** (**CFG**) is an object $G = (V, \Sigma, \mathcal{R}, S)$ where:

1. V is a finite alphabet of *variables*, aka. *non-terminals*.
2. Σ is the *terminal alphabet*.
3. S, a member of V, is the *start symbol*.
4. $\mathcal{R}$ is a finite set of *rules*, each of the form $A \to X$, where $A \in V$ and $X \in (V \cup \Sigma)^*$. More simply put, $\mathcal{R}$ is a subset of $V \times (V \cup \Sigma)^*$.

Common alternative notations are T in place of Σ and $\mathcal{P}$ for "productions" in place of $\mathcal{R}$ for "rules." Variables are commonly denoted by capital letters $A, B, C, D, \ldots$ This can confuse with languages but hopefully not in-context, and besides, each variable A will stand for the language L_A of strings that can be derived from that variable, with $L(G) = L_S$. A common alternate notation is to put variables in angle-brackets, such as $\langle sentence \rangle$ in place of S. Then also the arrow is often written ":=" or "::=" which goes into **Backus-Naur Form** (**BNF**) grammars. BNF is more liberal than CFG notation but stays equivalent in the languages that can be represented.

Ordinary strings in Σ^* will be put in lowercase as $x, y, z, w, \ldots$ as usual. But strings that can include variables will be emphasized by putting them in uppercase as $X, Y, Z, W, \ldots$ etc. Two or more rules for the same variable are often grouped using | between the alternatives, for instance $A \to X \mid Y \mid Z$. This hints right away that CFGs are fundamentally nondeterministic and that the class of languages we get will be closed under union. Usually V, Σ, and the start symbol can be inferred from how the rules are laid out, so it is only necessary to state the rules in order to specify a context-free grammar G. Here is how to define the language $L(G)$:

Definition: Given a CFG G and two strings $Y, Z \in (\Sigma \cup V)^*$, we write $Y \Longrightarrow_G Z$ and say that Y derives Z in one step if there are strings $U, W \in (\Sigma \cup V)^*$ and a rule $A \to X$ in $\mathcal{R}$ such that

$$Y = UAW \quad \text{and} \quad Z = UXW.$$

We say that A was re-written to X by the rule. This is called "context-free" because none of the letters next to A in Y matter. We drop the subscript G as is usually clear. Now one more "inductive escalation":

Definition:

- Any string $X \in (\Sigma \cup V)^*$ is considered to "derive itself in zero steps": $X \Longrightarrow^0 X$.
- For $k \geq 1$, write $X \Longrightarrow^k Z$ if there is a Y such that $X \Longrightarrow^{k-1} Y$ and $Y \Longrightarrow Z$.
- Then write $X \Longrightarrow^* Z$ if there is a $k \geq 0$ such that $X \Longrightarrow^k Z$.
- Finally, $L(G) = \left\{ x \in \Sigma^* : S \Longrightarrow^* x \right\}$.

An $X \in (V \cup \Sigma)^*$ such that $S \Longrightarrow^* X$ is called a **sentential form**. Sometimes we speak of "the language of sentential forms", but strictly speaking the language of the grammar is the set of derivable terminal strings only. Note that the superscript * keeps its reading of "zero or more" (steps). Now let's see some basic examples of grammars and derivations.

Example 1

$G = S \to 0S1 \mid \epsilon$ Here the start symbol S is the only variable, and $\Sigma = \{0, 1\}$. Some derivations:

- $S \Longrightarrow \epsilon$. This is a one-step derivation using the rule $S \to \epsilon$. The empty string counts as a terminal string since it belongs to Σ^* (as well as to $(\Sigma \cup V)^*$). So $\epsilon \in L(G)$.
- $S \Longrightarrow 0S1 \Longrightarrow 01$. The first step used the rule $S \to 0S1$, then we used the ϵ-rule. (Not all grammars have ϵ-rules, and we will later want to eliminate them from those that do.)
- $S \Longrightarrow 0S1 \Longrightarrow 00S11 \Longrightarrow 0011$.

This is enough to give the idea of why $L(G) = \left\{0^n 1^n : n \geq 0\right\}$.

Example 2

$G = S \to \epsilon \mid (S) \mid SS$. Again S is the only variable, but $\Sigma = \{(,)\}$ instead of $\{0, 1\}$.

The first two rules are much the same as in the first grammar (the order of writing the possible right-hand sides does not matter, and the only reason the order of writing rules for different variables might matter is if you need to put rule(s) for the start symbol first in order to say which it is). But the third rule "expands" by having two (or more) variables on the RHS.

- We can derive ϵ, $()$, $(())$, $((()))$, etc., much as in the previous grammar.
- But we can also do $S \Longrightarrow SS \Longrightarrow (S)S \Longrightarrow ()S \Longrightarrow ()(S) \Longrightarrow ()()$ to get other kinds of balanced-parenthesis expressions.
- In fact, $L(G) = BAL$.

$x = (( ) (( ) (( ))))$

$S \Longrightarrow (S) \Longrightarrow (SS) \Longrightarrow ((S)S) \Longrightarrow (()S) \Longrightarrow (()(S)) \Longrightarrow (()(SS)) \Longrightarrow^2 (()(()S)) \Longrightarrow^2 (()(()((S)))) \Longrightarrow x.$

Example 3

$G' = S \to \epsilon \mid (S) \mid (S)S$ is another grammar involving parentheses.

- Anything G' can derive can be derived in G because the rule $S \to (S)S$ in G' can be **simulated** by the two steps $S \Longrightarrow SS \Longrightarrow (S)S$ in G.
- Hence $L(G') \subseteq L(G)$. Since we already asserted that $L(G) = BAL$, this means $L(G') \subseteq BAL$. We therefore say that G' is **sound** for BAL.

- Is $L(G') = BAL$, which would follow if $L(G') \supseteq BAL$? The latter I call G' being **comprehensive** for BAL.
- The combination of being sound and comprehensive just means being *correct*.
- In fact, yes, G' is comprehensive---but that is often not as easy to prove as soundness.

Example 4

$G'' = S \to \epsilon \mid \$S \mid \$SDS$ generates all strings in the spears-and-dragons game with unlimited spears in which the "Player" survives. We can make it look more like G' by changing "spear" to (and "dragon" to) and ignoring 0 for "empty room":

$$G'' = S \to \epsilon \mid (S \mid (S)S.$$

Then $L(G'')$ is the language of parenthesis expressions that can be properly closed by appending zero or more right parens.

Example 5 (with more than one variable):

$$\begin{aligned} S &\to \epsilon \mid aB \mid bA \\ A &\to a \mid aS \mid bAA \\ B &\to b \mid bS \mid aBB \end{aligned}$$

What is the language? Think of the variables as saying:

- S: "Every string I derive has equal #s of a's and b's---and I derive all such strings."
- A: "Every string I derive has one more a than b---and I derive all such strings."
- B: "Every string I derive has one more b than a---and I derive all such strings."

Every rule is **sound** in the sense that if each variable on the right-hand side does what it says, then the variable on the left-hand side fulfills its promise. This means in particular that L_S, which is the language of G, is a subset of $EQ = \{x \in \{a,b\}^* : \#a(x) = \#b(x)\}$. Thus we can say that G is **sound** for the target language EQ. If it is also **comprehensive**, meaning $L(G) \supseteq EQ$, then in this case it will be **correct**, of course meaning $L(G) = EQ$. Yes, G is correct---but comprehensiveness is often a lot harder to prove than soundness, because you can't just examine individual rules but have to use all the rules in concert. [One fact that helps in this case is that G is in **Greibach normal form** (Sheila Greibach, UCLA), meaning that the RHS of every rule begins with a terminal letter---with $S \to \epsilon$ as an allowed exception. This form helps you **parse** strings in a left-to-right manner.]

Example 6: Palindromes and Even Palindromes

The CFG $G = S \rightarrow aSa \mid bSb \mid \epsilon \mid a \mid b$ (with $\Sigma = \{a, b\}$) makes $L(G) = PAL$. If we take away the terminal rules $S \rightarrow a$ and $S \rightarrow b$, leaving $G' = S \rightarrow aSa \mid bSb \mid \epsilon$, then we get $L(G')$ equal to the language of even-length palindromes: EVENPAL = $\left\{x \cdot x^R : x \in \{a, b\}^*\right\}$.

Non-Example: The Double-Word Language

Recall DOUBLEWORD = $\left\{x \cdot x : x \in \{a, b\}^*\right\}$. This looks even simpler than EVENPAL. However, there does not exist a CFG G such that $L(G) =$ DOUBLEWORD. We will later prove this via the CFL Pumping Lemma when we hit section 2.3 (after jumping over most of section 2.2). As a self-study challenge, however, see if you can figure out the language of the grammar

$$
\begin{aligned}
S &\rightarrow AB \mid BA \\
A &\rightarrow aAa \mid aAb \mid bAa \mid bAb \mid a \\
B &\rightarrow aBa \mid aBb \mid bBa \mid bBb \mid b \quad .
\end{aligned}
$$

(The final period is just punctuation, not part of Σ.) As a warmup, can this grammar ever derive a string of odd length? We will come back to this later.