CS 4510 Automata and Complexity Janurary 25th 2023

Lecture 5: Context-Free Grammars

Lecturer: Abrahim Ladha *Scribe(s): Rishabh Singhal*

1 Background

Automata So far we have only looked at automata. These are usually tasked with **Decision or Recognition**. It's a fairly mechanical model, a decision procedure. You look at the input scanning left to right and do something.

- Given $w \in \Sigma$, is $w \in L(D)$? This is not that hard, you just run the automata on the input.
- Characterizing all of $L(D)$? This is much harder for an automata. If I give you a DFA or NFA and ask you to describe exactly the strings it accepts, this is not as easy.

Grammars In contrast, a grammar is tasked with **Production or Generation**. A grammar will non-deterministically produce only the correct strings, like a flower blooming. It doesn't start with an input to look at, it starts with nothing. Defined with the rules we give it, it will produce a string according to those rules.

- Given $w \in \Sigma$, is $w \in L(G)$? This is surprisingly non-trivial
- Characterizing all of $L(G)$? This is surprisingly easier.

2 Formal Definition of Context-Free Grammar

We represent a context-free grammar (CFG) as a four tuple (V, Σ, R, S) such that:

- V Non-Terminals or Variables. These are always capitalized like $\{S, A, B, ...\}$
- Σ Terminals or our alphabet. These are always lower-case like $\{a, b, c, ...\}$
- R Productions or Rules. Each are of them will be of form $V \to (V \cup \Sigma)^*$. The left-hand side of the production will be a single non-terminal and the right-hand side will be a string of terminals and non-terminals.
- $S \in V$ is our designated start non-terminal.

For $A \in V, w \in (V \cup \Sigma)^*$, with production of the form $A \to w$, we apply a production as a substring replacement of a "working string" like $xAz \implies xwz$, for $x, z \in (V \cup \Sigma)^*$. When we write $w_i \implies w_{i+1}$ we mean that w_i "yields" w_{i+1} after application of one production. If $S \implies w_1 \implies w_2 \implies w_3 ... \implies w_n$ with $w \in \Sigma^*$ we say that $w \in L(G)$ and may write $S \overset{*}{\implies} w$.

For a context-free grammar G, we characterize the set of strings in $L(G)$ as those and only those produced non-deterministically starting from S. Observe that a production halts when there are no more non-terminals in the working string. We say that a language L is context-free if ther exists a context-free grammar G such that $L = L(G)$.

2.1 Examples

Like a state diagram, you can give all parts of the CFG by just giving the set of productions. It implicitly gives the terminals and non-terminals, and we always denote S as the start non-terminal.

2.1.1 $\{a^n b^n \mid n \in \mathbb{N}\}$

We write $\{S \to aSb, S \to \varepsilon\}$ or just $\{S \to aSb \mid \varepsilon\}$. If we have two or more productions with the same beginning non-terminal, we may use "|" as a shorthand to "or" those productions together. Let us say we want to produce a^3b^3 the process we follow is

$$S \implies aSb \implies a(aSb)b \implies aaSbb \implies aaaSbbb \implies aaabbb \implies a^3b^3$$

We repeatedly apply the first production, and terminate when we have no more non-terminals in our working string. This occurs when we apply the second rule, $S \to \varepsilon$. Notice that it has to produce exactly the strings of the form a^nb^n. This was our canonical example of a non-regular language, the first one we used for pumping. This should convince you atleast, that the languages produced by context-free grammars, $\mathscr{L}(CFG)$, is not equal to the regular languages. Later we will show it is a strict super set.

2.1.2 $\{ww^R \mid w \in \Sigma^*\}$

Our productions are similar. $\{S \to aSa \mid bSb \mid \varepsilon\}$. This generates even length palindromes. As we apply productions, the left and right of our primary recursive production effectively act like two stacks, mirrors of each other. This generates the string which is a palindrome and these strings are also even in length. We can conserve the same idea, to generate palindrome of odd length.

2.1.3 $\{w\Sigma w^R \mid w \in \Sigma^*\}$

We write this as $\{S \to aSa \mid bSb \mid a \mid b\}$. We may combine ideas from the previous two examples to show the set of all palindromes is a context-free language, with the grammar $\{S \to aSa \mid bSb \mid a \mid b \mid \varepsilon\}$. We pumped a third language, $\{ww \mid w \in \Sigma^*\}$. As some foreshadowing, this language is not regular, but it is also not context free.

2.1.4 Σ^*

There exist many equivalent grammars for this language. These may include

- $S \to aSa \mid bSb \mid aSb \mid bSa \mid a \mid b \mid \varepsilon$
- $S \to aaS \mid abS \mid baS \mid bbS \mid a \mid b \mid \varepsilon$
- $S \to aS \mid bS \mid \varepsilon$

2.1.5 $\emptyset$

If a grammar produces no strings, not even ε, it is either trivial, or some how does not have a halting condition. There are a few you could come up with, but a non-trivial grammar for this would be $\{S \to A, A \to S\}$. No production of this terminates.

2.1.6 Dyck Language

Consider the grammar $\{S \to (S) \mid SS \mid \varepsilon\}$. This language is the set of balanced, or matching paranthesis. It has a special name, called the Dyck language.

We can prove it is not regular by closure. Assume to the contrary $L(G)$ was regular. Then by closure, so must be $L(G) \cap (^*)^*$. The left side enforces that the number of opens equals the number of closes, and the right hand side enforces that all the opens come before all the closes. The intersection is the logical and of these, so we see this intersection must be equal to $\{(^n)^n \mid n \in \mathbb{N}\}$, our canonical non-regular language, a contradiction. Therefore, the Dyck language is not regular.

2.1.7 Arithmetic Expressions

Consider the following grammar:

$$\begin{aligned} S &\to S + T \mid T \\ T &\to T \times F \mid F \\ F &\to (S) \quad\ \mid a \end{aligned}$$

with $V = \{S, T, F\}, \Sigma = \{(,), \times, +, a\}$. Lets do an example of a long production to show this grammar generates $(a + a) \times a$

$$\begin{gathered} S \implies T \implies T \times F \implies F \times F \implies (S) \times F \implies \\ (S) \times a \implies (S + T) \times a \implies (T + T) \times a \implies (F + T) \times a \implies \\ (F + F) \times a \implies (F + a) \times a \implies (a + a) \times a \end{gathered}$$

2.1.8 One last example

On the homework, you were asked to pump the language $\{a^n ba^m b^{n+m} \mid n, m \in \mathbb{N}\}$. First notice that for some n, m that $a^n ba^m b^{n+m} = a^n ba^m b^n b^m$. We have matching blocks of the same size, but we can't pair them up as written. We notice that letters of the same kind obviously commute, so we see $a^n ba^m b^n b^m = a^n ba^m b^m b^n = a^n (ba^m b^m) b^n$. This gives us the intuition on how we would build our grammar as $\{S \to aSb \mid bR, R \to aRb \mid \varepsilon\}$. Just to work out some productions, they may look like

$$S \overset{*}{\implies} a^n Sb^n \overset{*}{\implies} a^n bRb^n \overset{*}{\implies} a^n ba^m Rb^m b^n \overset{*}{\implies} a^n ba^m b^m b^n = a^n ba^m b^{m+n}$$

3 Relationship with Regular Languages

We say a grammar is right-regular if it only has productions of the form $A \to aB$ or $A \to a$ or $A \to \varepsilon$, where A, B are any non-terminals, and a is any terminal. Certainly every right-regular grammar is also context-free, we claim that the right-regular grammars decide exactly the regular languages. The proof of this characterization is not complicated, but tedious[1]. Instead we will highlight just the part of given a DFA, how one might construct a right-regular grammar. This should convince you that we are working with a strictly more powerful computational model, $\mathscr{L}(DFA) \subsetneq \mathscr{L}(CFG)$. For a DFA of the form $(Q, \Sigma, q_0, \delta, F)$ we construct a grammar (V, Σ, R, S).

- For $Q = \{q_0, ..., q_k\}$ we have non-terminals $V = \{Q_0, ..., Q_k\}$
- The set of terminals for our grammar is identical to the alphabet for our DFA: $\Sigma = \Sigma$
- For q_0 the start state of our DFA, we designate our start non-terminal as Q_0
- For every transition of the form $\delta(q_i, a) = q_j$, we add production $Q_i \to aQ_j$
- For every $q_f \in Q$, we add production $Q_f \to \varepsilon$

Convince yourself of its correctness.

4 Closure of Context-Free Language

We will prove that CFLs are closed under union, concatenation, and star. Let G_1, G_2 be two CFGs to produce $L(G_1)$ and $L(G_2)$ with start non-terminals S_1, S_2 respectively.

$L(G_1) \cup L(G_2)$	Copy all productions, add new start state S, and a new production $S \to S_1 \mid S_2$
$L(G_1)L(G_2)$	Similarly, with new production $S \to S_1S_2$
$L(G_1)^*$	Add new productions $S \to S_1S \mid \varepsilon$

Later we will show CFLs are not closed under intersection or complement. This may be intuitive, if you observe the behavior of a CFG. It only knows how to grow correct strings.

[1]I have a more detailed proof here https://ladha.me/files/sectionX/regulargrammars.pdf