CS 4510 Automata and Complexity	Sept 14
Lecture 7	
Lecturer: Frederic Faulkner	*Scribe(s):*

1 Context-Free Grammars

1.1 A new type of model

Believe it or not, at this point we have said pretty much everything interesting there is to say about regular languages (at least in this class). Finite automata have some power, as we have seen, but it is quite limited; so now we will begin to explore other types of models to see if they can handle languages that finite automata could not.

The model that we will cover next is called a *context-free grammar*. At first glance, it looks very different from the models we have seen so far: there are no states, just rules. Here is an example grammar:

$$S \to 0S1$$
$$S \to \varepsilon$$

Each line with an "$\to$" is a rule which says that you may replace the character on the left hand side with the string on the right hand side. Of course, we also need to know what we start with. In this case, assume that we start with just a single S. Then we could make a string like follows: $S \to 0S1 \to 00S11 \to 000S111$. (Each new color corresponds to another application of the first rule.) We now have the string $000S111$. However, we cannot stop there, because S is a *nonterminal*, which means that we cannot stop applying rules while it is present. However, we can now apply the second rule to get $000S111 \to 000\varepsilon111 = 000111$. Now there are no rules left that can apply and the only characters in the string are the *terminals* 0 and 1 so we are done. We see that this grammar generates the string 000111. You can probably see that you can also make the strings $\varepsilon, 01, 0011, 00001111$, and so on. So the language of this grammar is the non-regular language $\{0^n1^n | n \in \mathbb{Z}^{\geq 0}\}$.

We see then that context-free grammars seem to be more powerful than DFA's, since we have shown that some grammar generates a language that no DFA accepts.

1.2 Being Formal

Definition 1 *A Context-Free Grammar is a 4-tuple (V, Σ, R, S) where:*

- *V is the set of non-terminals (or variables). These are the characters that are to be replaced; they cannot be present in the final string.*
- *Σ is the set of terminals, also called the alphabet. The final string generated must consist only of terminals.*

- $R \subseteq (V \times (V \cup \Sigma)^*)$ *is the set of rules. They are typically written* $A \to w$*, where* A *is a single non-terminal, and* w *is a string of terminals and non-terminals.*
- $S \in V$ *is the starting non-terminal. All strings are derived by starting with a single "S".*

We typically use capital letters like A and B to represent non-terminals, and small letters like a and b to represent terminals. Note from the definition that the left hand side of a rule is *always* a single non-terminal, but the right hand side can contain both terminals and non-terminals (or neither). Thus $A \to B$,$A \to aBaAAab$, and $A \to \varepsilon$ are all legal rules, but some thing like $AB \to a$ or $a \to b$ are not.

For the previous grammar, the only non-terminal was S, the terminals (or alphabet) were 0 and 1, the starting non-terminal was S, and there were 2 rules.

Definition 2 *Let* u, v, w *be strings of terminals and non-terminals, let* A *be a non-terminal, and let* $A \to w$ *be a rule. Then we say that* uAv *yields* uwv*, written* $uAv \Rightarrow uwv$*.*

We say that u *derives* v*, written* $u \overset{*}{\Rightarrow} v$ *if there exists a sequence* $u_1, ..., u_k$ *such that* $u \Rightarrow u_1 \Rightarrow ... \Rightarrow u_k \Rightarrow v$*.*

If G *is a grammar with start symbol* S*, then we say that the language of* G *is the set of all strings derived from* S*, i.e.* $L(G) = \{w \mid S \overset{*}{\Rightarrow} w\}$*. We say a language is context-free if it is generated by some grammar* G*.*

This is just a formal way of explaining how one generates strings from a grammar: start from the start symbol and apply one rule at a time until there are no non-terminals left.

1.3 More examples

Prove that $L = \{w \mid w$ *is a palindrome*$\}$ *is context-free.* (A palindrome is a string that is the same forwards and backwards, such as 101 or *racecar*.)

We can prove that a language is context-free by giving a grammar for it. Here is a first attempt at a grammar:

$$S \to 0S0$$
$$S \to 1S1$$
$$S \to \varepsilon$$

This seems reasonable, as it allows us to generate strings like 0110, 0000, 1001, ... However, closer inspection reveals that all of these strings are even! Still, we do not need to start over from scratch; we just need to allow for a middle character:

$$S \to 0S0$$
$$S \to 0$$
$$S \to 1S1$$
$$S \to 1$$
$$S \to \varepsilon$$

Success! Now we can also generate odd length strings. We are up to five rules, which can start to make the grammar feel a bit unwieldy. To save space we use | to mean "or". It doesn't change the grammar at all, but it looks nicer to read:

$$S \to 0S0 \mid 1S1$$
$$S \to 0 \mid 1 \mid \varepsilon$$

2 Ambiguity

Consider the following grammar:

$$S \to AB \mid BC$$
$$A \to aA \mid a$$
$$B \to a \mid b \mid c$$
$$C \to cC \mid c$$

Note that there are multiple ways to derive the string ac. You could go $S \Rightarrow AB \Rightarrow aB \Rightarrow ac$, but you could also derive it as $S \Rightarrow BC \Rightarrow aC \Rightarrow ac$. Thus, we say that this grammar is *ambiguous*, since given a string from this language it might not be clear how this string was derived.

However, what about ab? It also might feel like this string is ambiguously derived, since you could choose to go $S \Rightarrow AB \Rightarrow aB \Rightarrow ab$ or $S \Rightarrow AB \Rightarrow Ab \Rightarrow ab$. However, note that, unlike the previous case, both of these derivations are effectively the same derivation. Each terminal comes from the same nonterminal in both derivations (a from A, b from B), each of those nonterminals A and B comes from the same nonterminal in both derivations, etc. Whereas, in one derivation for ac, the final a comes from an A, whereas in the other derivation it comes from a B! Thus we have the following definitions.

Definition 3 *A leftmost derivation for a context-free grammar G is a derivation in which the leftmost non-terminal at each step is the one to which a rule is applied.*

Definition 4 *A string x is said to be ambiguously derived by some grammar G if there are two different leftmost derivations of x by G.*

Definition 5 *A context-free grammar G is said to be ambiguous if it ambiguously generates some string x.*

Then we see that $S \Rightarrow AB \Rightarrow Ab \Rightarrow ab$ is not a leftmost derivation of ab, (we replace B while there is an A to the left of it), so we do not count it when deciding whether ab is ambiguously derived. However, the two derivations given for ac are both leftmost derivations, so the given grammar is indeed ambiguous. (Note that even if a grammar only generates one string ambiguously, we still label the whole grammar as ambiguous.)

It turns that in real-life applications, ambiguity in grammars can cause a lot of inconvenience. It is often preferable to replace an ambiguous grammar by an equivalent

unambiguous one. Unfortunately, that is not always possible: some context-free languages are inherently ambiguous, meaning that *every* grammar that generates them is ambiguous. The proof of this fact is beyond the scope of this course, but an example of such a language is $\{a^i b^j c^k \mid i = j \text{ or } j = k \text{ or both}\}$.