Notes for Friday, May 7DH Last time, we looked at a few examples of grammars:
$L_1 = \{0^n 1^n | 1 \geq 0\}$, and grammar G_1 :

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

and $L_2 = \{w|w$ contains an even number of 0's$\}$, and grammar G_2 :

$$S \to A0A0S|A$$

$$A \to 1A|\varepsilon$$

We call these Context-Free Grammars (CFG's). Why "context-free?"

Consider a "Context-Sensitive Grammar." In this case, "context" refers to surrounding variables. So any example of a Context-Sensitive Grammar might be

$$0A1 \to 00A1|01$$

$$1A0 \to 11A0|10$$

Recall that the formal definition of a CFG is $G = (V, \Sigma, R, S)$, with V the variables, Σ the terminals (or alphabet), R the rules, and S the start symbol. The rules are of the form $V \to (V \cup \Sigma)^*$ and $V \cap \Sigma = \emptyset$.

So a formal definition of G_1 above would be $G_1 = (\{S\}, \{0, 1\}, R, S)$ with $R = \{S \to 0S1, S \to \varepsilon\}$.

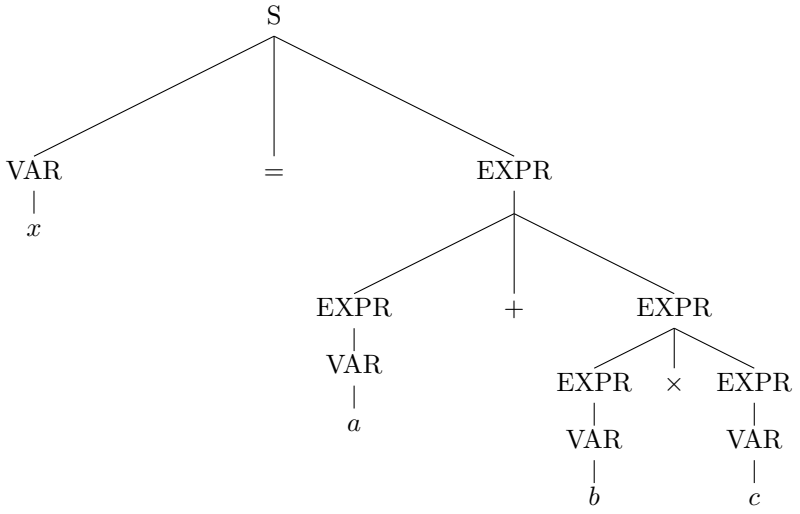
Example: CFG for assignment statements in a programming language (for example, $x = a + b \times c$. We can construct a CFG G_A with rules:

$$S \to VAR = EXPR$$

$$VAR \to a|b|c| \cdots |z$$

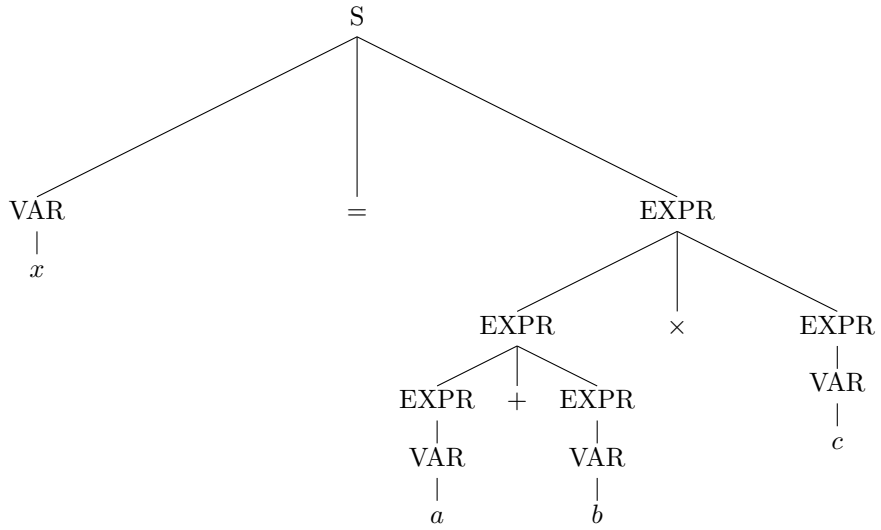
$$EXPR \to EXPR + EXPR|EXPR \times EXPR|VAR$$

We can create a parse tree for the string $x = a + b \times c$:



The leaves of the tree correspond to the string $x = a + b \times c$, as expected. This tree captures what we would expect with order of operations: it multiplies first, then adds, then assigns with the equals.

There is a different parse tree for this string:



This is also a valid parse tree, but this one does not follow the rules for evaluating an expression.

The two parse trees above have two different meanings. This is an example of an ambiguous grammar: an ambiguous grammar G is one for which there exists are least two parse trees for some $w \in L(G)$.

An ambiguous grammar is a bad thing in terms of writing a compiler for a programming language. We want to stay away from ambiguous grammars.

Some examples in English: "HOMER HIT THE GUY WITH THE BEER BOTTLE." There are two different ways of interpreting this:

Homer hit whom? The guy with the bottle

OR

Homer hit some guy with what? A bottle.

Another example: "THE GIRL TOUCHES THE BOY WITH THE FLOWER;" and "TIME FLIES LIKE AN ARROW, FRUIT FLIES LIKE A BANANA" (here, "flies" is interpreted as both a verb and a noun, and "like" is interpreted as an adjective or a verb.)

So how do we get rid of an ambiguity?

We can fix G_A. We want to multiply first, then add:

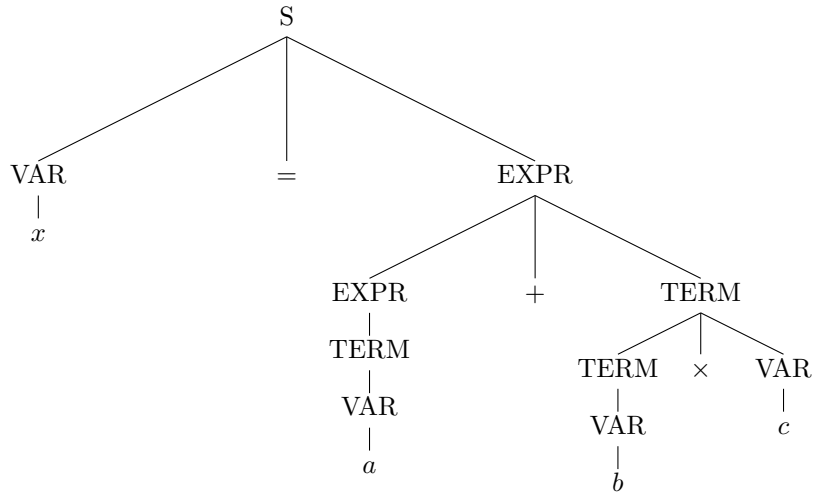
$$S \to VAR = EXPR$$

$$VAR \to a|b|c|\cdots|z$$

$$EXPR \to EXPR + TERM|TERM$$

$$TERM \to TERM \times VAR|VAR$$

A parse tree for $x = a + b \times c$ with this grammar:



The question arises: can you always convert an ambiguous grammar to an unambiguous one? In fact, there are examples of languages that will inherently be ambiguous.

Example: := $\{0^i1^j2^k | i = j \text{ or } j = k\}$.

The string $0^n1^n2^n$ can always be parsed as an equal number of 0's and 1's, OR as an equal number of 1's and 2's.

Programming languages are made to avoid this inherent ambiguity problem.

Now we ask the question: What is the relationship between CFLs (context-free languages) and regular languages? Recall:

$$REG = \{L | L \text{ is regular}\}$$

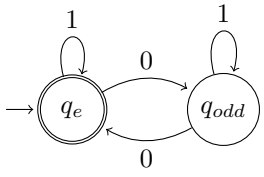
$$CFL = \{L = L(G) \text{ for some CFG } G\}$$

Is one a subset of the other? Are they equal?

The counterexample 0^n1^n shows that they are not equal.

What about $REG \subset CFL$? Are $CFLs$ more powerful? Can you always convert a DFA into a grammar?

Example DFA M:



$L(M) = \{w | w \text{ contains an even number of 0's}\}$.

Can we create a CFG G_M that mimics M? If we think of this relationship between variables and states: $S \leftrightarrow q_e$ and $A \leftrightarrow q_{odd}$, then we create the rules:

$$S \to 1S|0A|\varepsilon$$

$$A \to 1A|0S$$

which follows the transition rules of the DFA above (ε corresponds to an accept state, because it terminates the string).

This is different from the example G_2 at the very beginning of these notes. It is the same language, but uses different approaches.

Can we generalize this construction? Is there anything special about M that allowed us to do this? The answer is no: this construction should be valid for any DFA. The fuller proof will be given next time.

This construction allows us to conclude that the regular languages are a proper subset of CFLs!