Lecture 14

Homework #14: 3.1.2, 3.1.3, 3.1.4, 3.1.5, 3.1.7, 3.1.9 a & d

In our consideration of the pumping theorem, we saw that many "simple" languages are not regular.

Now let's begin to move beyond the realm of the regular languages:

Context-Free Languages - a step up in the Chomsky hierarchy.

As with regular languages, we'll study both
(1) methods for generating them and
(2) machines that accept them

We'll start with the method for generating them: Context-Free Grammars (CFGs)

CFGs:

(1) specify a starting point;
(2) specify rules to follow for generating strings;
(3) may provide a number of alternative rules to apply in the generation (derivation) of a string.

Here's **an example.**

Let S be our starting point.

And let these be the rules to follow for generating strings:
$S \rightarrow aMb$
$M \rightarrow aMb$
$M \rightarrow e$

These are "replacement rules". Each rule indicates that the symbol on the left-hand side may be replaced by the string on the right-hand side.

A sample derivation of a string according to these rules is:

$S \Rightarrow aMb \Rightarrow aaMbb \Rightarrow aaaMbbb \Rightarrow aaabbb$

[If you've seen how grammars of programming languages are specified, this will be familiar.]

This generator (or grammar) is called Context-Free because you can replace the left-hand-side symbol (for example, M) regardless of the rest of the string (i.e., regardless of the context in which M is found).

Formally:

A **CFG** G is a quadruple (V, Σ, R, S), where

V is an alphabet,
Σ (the set of terminals) is a subset of V
R (the set of rules) is a finite subset of $(V-\Sigma) \times V^*$
S (the start symbol) is an element of $V-\Sigma$

$V-\Sigma$ is the set of non-terminals (i.e., symbols that derive something)

Notation:

We write $A \rightarrow u$, whenver $(A, u) \in R$.
$A \in V-\Sigma$
$u \in V^*$

$u \Rightarrow_G v$ iff $u=xAy$,
$v=xv'y$,
$A \rightarrow_G v'$,
$A \in V-\Sigma$
$u,x,y,v,v' \in V^*$

$\Rightarrow^*$ is the reflexive transitive closure of $\Rightarrow$
[Note that in general we can omit the G indicating the grammar]

$L(G)$ is the language generated by $G = \{w \in \Sigma^* : S \Rightarrow^* w\}$

L is a context-free language iff it is generated by a context-free grammar.

A derivation is any sequence
$w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n$

The length of a derivation n is ≥ 0. (can also say that the derivation takes n steps)

Example.
Write a grammar that generates strings that are legal boolean expressions.

[Note that in genl CFG are sufficient to represent programming language constructs; They're not enough to represent natural language, but they are used as a starting point for many NLP algorithms.]

For this example, let

Σ = {(,), ||, &&, !, true, false, x1, x2}, where x1 and x2 are boolean vars.

V = {E, B, N, (,), ||, &&, !, true, false, x1, x2}

S = E

R: E → E || E
E → E && E
E → ! E
E → (E)
E → B (boolean value)
E → N (Named boolean var)
B → true | false
N → x1 | x2

How would the grammar generate (i.e. derive)

(! (x1 && x2))

E ⇒ (E)
(!E)
(! (E))
(! (E && E))
(! (N && E))
(! (x1 && E))
(! (x1 && N))

(! (x1 && x2))

It will often **not** be obvious that a given CFG generates a particular language. So, we need to know how to prove that it does:

Consider the grammar $G = (V, \Sigma, R, S)$, where

$V = \{S, a, b\}$
$\Sigma = \{a, b\}$
$R = \{S \rightarrow aSb, S \rightarrow e\}$

which generates the language $\{a^n b^n : n \geq 0\}$

This one is fairly obvious, but if it weren't we'd need to show:

(1) That the grammar generates only strings of the form specified for the language;
(2) That all strings in the language are generated by the grammar.

For this particular example.

(1) Show that every string generated by G contains an equal number of a's and b's and that all a's in the string precede all b's in the string.

We'll actually show that this is true at **every step** in a derivation; we'll also show that the a's and b's are either separated by nothing or are separated by S.

The proof will proceed by induction on the length (k) of a derivation:

$S = w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_k = w, k \geq 1$

Basis. $k = 1$.

We need to consider 2 possible derivations of length 1:

$S \Rightarrow aSb$
$S \Rightarrow e$

In either case, all a's precede all b's and the number of a's and b's is equal (1 in the first case, 0 in the second); furthermore, the a's and b's are separated either by S (in the first case) or by nothing (in the second).

Inductive Hypothesis. Assume that the claim is true for all derivations of length k.

Now say that $S \Rightarrow^+ w$ in (k+1) steps.

The derivation can be written as

$S \Rightarrow^* w' \Rightarrow w$, where $w', w \in V^*$

By the IH, w' has an equal number of a's and b's (and all a's precede all b's). Now we must consider the step from $w' \Rightarrow w$. This implies that w' must contain S, which (by the IH) separates the a's in the string from the b's. Now consider the rules that might be applied at this point in the derivation:

$S \Rightarrow aSb$ This adds an equal number of a's and b's; it preserves the property that a's precede b's and that they are separated by S.

$S \Rightarrow e$ This adds an equal number of a's and b's; preserves the existing order of a's and b's and results in their being separated by e.

(2) Now we need to show that every string of the form $a^n b^n$ can be generated by the grammar.

To do this, we need to give an algorithm for generating strings from the grammar:

a string w of the form $a^n b^n$ can be generated as follows:

Apply the rule
$S \rightarrow aSb$ n times

Then apply the rule
$S \rightarrow e$ one time.