CS 341 Homework 15
Parsing

1. Show that the following languages are deterministic context free.

(a) $\{a^m b^n : m \neq n\}$

(b) $\{wcw^R : w \in \{a, b\}^*\}$

(c) $\{ca^m b^m : m \geq 0\} \cup \{da^m b^{2m} : m \geq 0\}$

(d) $(a^m cb^m : m \geq 0\} \cup \{a^m db^{2m} : m \geq 0\}$

2. Consider the context-free grammar: $G = (V, \Sigma, R, S)$, where $V = \{(,), ., a, S, A\}$, $\Sigma = \{(,), .\}$, and $R =$

{ $S \rightarrow ()$,
$S \rightarrow a$,
$S \rightarrow (A)$,
$A \rightarrow S$,
$A \rightarrow A.S$} (If you are familiar with the programming language LISP, notice that L(G) contains all atoms and lists, where the symbol a stands for any non-null atom.)

(a) Apply left factoring and the rule for getting rid of left recursion to G. Let G' be the resulting grammar. Argue that G' is LL(1). Construct a deterministic pushdown automaton M that accepts L(G)\$ by doing a top down parse. Study the computation of M on the string ((()).a).

(b) Repeat Part (a) for the grammar resulting from G if one replaces the first rule by $A \rightarrow \varepsilon$.

(c) Repeat Part (a) for the grammar resulting from G if one replaces the last rule by $A \rightarrow S.A$.

3. Answer each of the following questions True or False. If you choose false, you should be able to state a counterexample.

(a) If a language L can be described by a regular expression, we can be sure it is a context-free language.

(b) If a language L cannot be described by a regular expression, we can be sure it is not a context-free language.

(c) If L is generated by a context-free grammar, then L cannot be regular.

(d) If there is no pushdown automaton accepting L, then L cannot be regular.

(e) If L is accepted by a nondeterministic finite automaton, then there is some deterministic PDA accepting L.

(f) If L is accepted by a deterministic PDA, then L' (the complement of L) must be regular.

(g) If L is accepted by a deterministic PDA, then L' must be context free.

(h) If, for a given L in $\{a, b\}^*$, there exist x, y, z, such that $y \neq \varepsilon$ and $xy^n z \in L$ for all $n \geq 0$, then L must be regular.

(i) If, for a given L in $\{a, b\}^*$, there do not exist u, v, x, y, z such that $|vy| \geq 1$ and $uv^n xy^n z \in L$ for all $n \geq 0$, then L cannot be regular.

(j) If L is regular and $L = L1 \cap L2$ for some L1 and L2, then at least one of L1 and L2 must be regular.

(k) If L is context free and L = L1L2 for some L1 and L2, then L1 and L2 must both be context free.

(l) If L is context free, then L* must be regular.

(m) If L is an infinite context-free language, then in any context-free grammar generating L there exists at least one recursive rule.

(n) If L is an infinite context-free language, then there is some context-free grammar generating L that has no rule of the form $A \rightarrow B$, where A and B are nonterminal symbols.

(o) Every context-free grammar can be converted into an equivalent regular grammar.

(p) Given a context-free grammar generating L, every string in L has a right-most derivation.

4. Recall problem 4 from Homework 12. It asked you to consider the following grammar for a language L (the start symbol is S; the alphabets are implicit in the rules):

$S \rightarrow SS \mid AAA \mid \varepsilon$
$A \rightarrow aA \mid Aa \mid b$

(a) It is not possible to convert this grammar to an equivalent one in Chomsky Normal Form. Why not?

(b) Modify the grammar as little as possible so that it generates L - ε. Now convert this new grammar to Chomsky Normal Form. Is the resulting grammar still ambiguous? Why or why not?

(c) From either the original grammar for L - ε or the one in Chomsky Normal Form, construct a PDA that accepts L - ε.

5. Consider the following language : L = $\{w^R w'' : w \in \{a, b\}^*$ and w'' indicates w with each occurrence of a replaced by b, and vice versa$\}$. In Homework 12, problem 5, you wrote a context-free grammar for L. Then, in Homework 13, problem 3, you wrote a PDA M that accepts L and traced one of its computations. Now decide whether you think L is deterministic context free. Defend your answer.

6. Convert the following grammar for arithmetic expressions to Chomsky Normal Form:

$E \rightarrow E + T$
$E \rightarrow T$
$T \rightarrow T * F$
$T \rightarrow F$
$F \rightarrow (E)$
$F \rightarrow \text{id}$

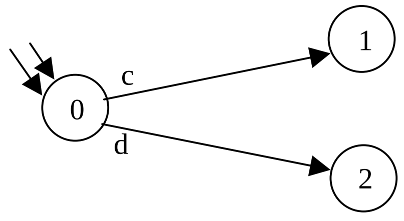
7. Again, consider the grammar for arithmetic expressions given in Problem 6. Walk through the process of doing a top down parse of the following strings using that grammar. Point out the places where a decision has to be made about what to do.
(a) id * id + id
(b) id * id * id

Solutions

1. (a) L = $\{a^m b^n : m \neq n\}$. To show that a language L is deterministic context free, we need to show a deterministic PDA that accepts L\$. We did that for L = $\{a^m b^n : m \neq n\}$ in class. (See Lecture Notes 14).

(b) L = $\{wcw^R : w \in \{a, b\}^*\}$. In class (again see Lecture Notes 14), we built a deterministic PDA to accept L = $\{wcw^R : w \in \{a, b\}^*\}$. It's easy to turn it into a deterministic PDA that accepts L\$.

(c) L = $\{ca^m b^m : m \geq 0\} \cup \{da^m b^{2m} : m \geq 0\}$. Often it's hard to build a deterministic PDA for a language that is formed by taking the union of two other languages. For example, $\{a^m b^m : m \geq 0\} \cup \{a^m b^{2m} : m \geq 0\}$ would be hard (in fact it's impossible) because we have no way of knowing, until we run out of b's, whether we're expecting two b's for each a or just one. However, $\{ca^m b^m : m \geq 0\} \cup \{da^m b^{2m} : m \geq 0\}$ is actually quite easy. Every string starts with a c or a d. If it's a c, then we know to look for one b for each a; if it's a d, then we know to look for two. So the first thing we do is to start our machine like this:



The machine that starts in state 1 is our classic machine for $a^n b^n$, except of course that it must have a final transition on \$ to its final state.

We have two choices for the machine that starts in state 2. It can either push one a for every a it sees, and then pop an a for every pair of b's, or it can push two a's for every a it sees, and then pop one a for every b.

(d) L = (a^mcb^m : $m \geq 0$} $\cup$ {a^mdb^{2m} : $m \geq 0$}. Now we've got another unioned language. But this time we don't get a clue from the first character which part of the language we're dealing with. That turns out to be okay though, because we do find out before we have to start processing the b's whether we've got two b's for each a or just one. Recall the two approaches we mentioned for machine 2 in the last problem. What we need here is the first, the one that pushes a single a for each a it sees. Then, when we see a c or d, we branch and either pop an a for each b or pop an a for every two b's.

2. (a) We need to apply left factoring to the two rules S $\rightarrow$ () and S $\rightarrow$ (A). We also need to eliminate the left recursion from A $\rightarrow$ A . S. Applying left factoring, we get the first column shown here. Then getting rid of left recursion gets us the second column:

S $\rightarrow$ (S′	S $\rightarrow$ (S′
S′ $\rightarrow$)	S′ $\rightarrow$)
S′ $\rightarrow$ A)	S′ $\rightarrow$ A)
S $\rightarrow$ a	S $\rightarrow$ a
A $\rightarrow$ S	A $\rightarrow$ SA′
A $\rightarrow$ A.S	A′ $\rightarrow$ .SA′
	A′ $\rightarrow$ ε

(b) Notice that the point of the first rule, which was S $\rightarrow$ (), was to get a set of parentheses with no A inside. An alternative way to do that is to dump that rule but to add the rule A $\rightarrow$ ε. Now we always introduce an A when we expand S, but we can get rid of it later. If we do this, then there's no left factoring to be done. We still have to get rid of the left recursion on A, just as we did above, however.

(c) If we change A $\rightarrow$ A.S to S $\rightarrow$ S.A, then there's no left recursion to get rid of and we can leave the rules unchanged. Notice, though, that we'll get different parse trees this way, which may or may not be important. To see this, consider the string (a.a.a) and parse it using both the original grammar and the one we get if we change the last rule.

3. (a) True, since all regular languages are context-free.
(b) False, there exist languages that are context-free but not regular.
(c) False. All regular languages are also context-free and thus are generated by context-free grammars.
(d) True, since if L were regular, it would also be context free and thus would be accepted by some PDA.
(e) True, since there must also be a deterministic FSM and thus a deterministic PDA.
(f) False. Consider L = a^nb^n. L' = {w $\in$ {a, b}* : either some b comes before some a or there is an unequal number of a's and b's.}. Clearly this language is not regular since we can't count the a's and b's.
(g) True, since the deterministic context-free languages are closed under complement.
(h) False. Suppose L = $a^nc^*b^n$, which is clearly not regular. Let x = aa, y = c, and z = bb. $xy^nz \in$ L.
(i) False. L could be finite.
(j) False. L1 could be a^nb^n and L2 could be {$\varepsilon \cup a^nb^m$: $n \neq m$}. Neither is regular. But L1 $\cap$ L2 = {ε}, which is regular.
(k) False. Let L1 = a* and L2 = {$a^nb^mc^m$: $n \neq m$}. L2 is not context free. But L = L1L2 = $a^*b^mc^m$, which is context free.
(l) False. Let L = ww^R.
(m) True.
(n) True, since we have a procedure for eliminating such unit productions.
(o) False, since there exist context-free languages that are not regular.
(p) True.

4. (a) No grammar in Chomsky Normal Form can generate ε, yet $\varepsilon \in$ L.

(b) In the original grammar, we could generate zero copies of AAA (by letting S go to ε), one copy of AAA (by letting S go to AAA), two copies (by letting S go to SS and then each of them to AAA), three copies of AAA (by letting S go to SS, then one of the S's goes to SS, then all three go to AAA), and so forth. We want to make sure that we can still get one copy, two copies, three copies, etc. We just want to eliminate the zero copies option. Note that the only role of S is to determine how many copies of AAA are produced. Once we have generated A's we can never go back and apply the S rules again. So all we have to do is to eliminate the production $S \rightarrow \varepsilon$. The modified grammar to accept L - ε is thus:

G = ({S, A, B, C, a, b}, {a, b}, R, S), where R = {

$S \rightarrow SS \mid AAA$

$A \rightarrow aA \mid Aa \mid b$

If we convert this grammar to Chomsky Normal Form, we get:

G = ({S, A, B, C, a, b}, {a, b}, R, S), where R = {

$S \rightarrow SS$	$A \rightarrow AC$
$S \rightarrow AB$	$A \rightarrow b$
$B \rightarrow AA$	$C \rightarrow a$
$A \rightarrow CA$ }	

This grammar is still ambiguous.

(c) (from the grammar of part (b)): M = ({p, q}, {a, b}, {S, A, a, b,}, Δ, p, {q})

Δ = { ((p, ε, ε), (q, S))	((q, ε, A), (q, aA))
((q, ε, S), (q, SS)	((q, ε, A), (q, Aa))
((q, ε, S), (q, AAA)	((q, ε, A), (q, b))
	((q, a, a), (q, ε))
	((q, b, b), (q, ε)) }

5. L is not deterministic context free for essentially the same reason that ww^R is not.

6. The original grammar was:

$E \rightarrow E + T$
$E \rightarrow T$
$T \rightarrow T * F$
$T \rightarrow F$
$F \rightarrow (E)$
$F \rightarrow id$

Step 2. There are no ∈ rules. We show steps 3, 4, and 5 next to each other, so it's clear where the rules in steps 4 and 5 came from. In each case, the first rule that is derived from a step 3 rule is next to its source. If more than one rule is derived from any given step 3 rule, the second and others are shown immediately under the first. That's why there are some blank lines in the first two columns.

Step 3.

$E \Rightarrow^* T, F$
$T \Rightarrow^* F$

G" = $E \rightarrow E + T$

$T \rightarrow T * F$

$F \rightarrow (E)$

$F \rightarrow id$

Then we add:

$E \rightarrow T * F$
$E \rightarrow (E)$
$E \rightarrow id$
$T \rightarrow (E)$
$T \rightarrow id$

Step 4.

$E \rightarrow EPT$

$T \rightarrow TMF$

$F \rightarrow LER$

$F \rightarrow id$

$E \rightarrow TMF$
$E \rightarrow LER$
$E \rightarrow id$
$T \rightarrow LER$
$T \rightarrow id$
$P \rightarrow +$
$M \rightarrow *$
$L \rightarrow ($
$R \rightarrow)$

Step 5.

$E \rightarrow E\ E'$
$E' \rightarrow PT$
$T \rightarrow T\ T'$
$T' \rightarrow M\ F$
$F \rightarrow L\ F'$
$F' \rightarrow E\ R$
$F \rightarrow id$

$E \rightarrow T\ T'$ (since $T' \rightarrow M\ F$)
$E \rightarrow LF'$ (since $F' \rightarrow E\ R$)
$E \rightarrow id$
$T \rightarrow L\ F'$ (since $F' \rightarrow E\ R$)
$T \rightarrow id$
$P \rightarrow +$
$M \rightarrow *$
$L \rightarrow ($
$R \rightarrow)$