CS 334 – Fall 2004: Answers to Assignment 8

1. 7.1.1

A and *C* are clearly generating, since they have productions with terminal bodies. Then we can discover *S* is generating because of the production *S*->*CA*, whose body consists of only symbols that are generating. However, *B* is not generating. Eliminating *B*, leaves the grammar

```
S -> CA
A -> a
C -> b
```

Since *S*, *A*, and *C* are each reachable from *S*, all the remaining symbols are useful, and the above grammar is the answer to the question.

2. 7.1.2

a)

Only *S* is nullable, so we must choose, at each point where *S* occurs in a body, to eliminate it or not. Since there is no body that consists only of *S*'s, we do not have to invoke the rule about not eliminating an entire body. The resulting grammar:

```
S -> ASB | AB
A -> aAS | aA | a
B -> SbS | bS | Sb | b | A | bb
```

b)

The only unit production is *B* -> *A*. Thus, it suffices to replace this body *A* by the bodies of all the *A*-productions. The result:

```
S -> ASB | AB
A -> aAS | aA | a
B -> SbS | bS | Sb | b | aAS | aA | a | bb
```

c)

Observe that *A* and *B* each derive terminal strings, and therefore so does *S*. Thus, there are no useless symbols.

d)

Introduce variables and productions *C* -> *a* and *D* -> *b*, and use the new variables in all bodies that are not a single terminal:

```
S -> ASB | AB
A -> CAS | CA | a
B -> SDS | DS | SD | b | CAS | CA | a | DD
C -> a
D -> b
```

Finally, there are bodies of length 3; one, *CAS*, appears twice. Introduce new variables *E*, *F*, and *G* to split these bodies, yielding the CNF grammar:

```
S -> AE | AB
A -> CF | CA | a
B -> SG | DS | SD | b | CF | CA | a | DD
C -> a
D -> b
E -> SB
F -> AS
G -> DS
```

3. 7.1.10

It's not possible. The reason is that induction on the number of steps in a derivation shows that every sentential form has odd length. Thus, it is not possible to find such a grammar for a language as simple as {00}.

To see why, suppose we begin with start symbol *S* and try to pick a first production. If we pick a production with a single terminal as body, we derive a string of length 1 and are done. If we pick a body with three variables, then, since there is no way for a variable to derive epsilon, we are forced to derive a string of length 3 or more.

4. 7.2.1a

Let *n* be the pumping-lemma constant and consider string $z = a^n b^{n+1} c^{n+2}$. We may write $z = uvwxy$, where *v* and *x*, may be ``pumped,'' and $|vwx| <= n$. If *vwx* does not have *c*'s, then uv^3wx^3y has at least *n+2* *a*'s or *b*'s, and thus could not be in the language.

If *vwx* has a *c*, then it could not have an *a*, because its length is limited to *n*. Thus, *uwy* has *n* *a*'s, but no more than *2n+2* *b*'s and *c*'s in total. Thus, it is not possible that *uwy* has more *b*'s than *a*'s and also has more *c*'s than *b*'s. We conclude that *uwy* is not in the language, and now have a contradiction no matter how *z* is broken into *uvwxy*.

5. 7.3.1a

For each variable *A* of the original grammar *G*, let *A'* be a new variable that generates *init* of what *A* generates. Thus, if *S* is the start symbol of *G*, we make *S'* the new start symbol.

If *A* -> *BC* is a production of *G*, then in the new grammar we have *A* -> *BC*, *A'* -> *BC'*, and *A'* -> *B'*. If *A* -> *a* is a production of *G*, then the new grammar has *A* -> *a*, *A'* -> *a*, and *A'* -> Λ.

6. 7.3.3a

Consider the language $L = \{a^ib^jc^k \mid i <= k \text{ or } j <= k\}$. *L* is easily seen to be a CFL; you can design a PDA that guesses whether to compare the *a*'s or *b*'s with the *c*'s. However, $min(L) = \{a^ib^jc^k \mid k = \min(i,j)\}$. It is also easy to show, using the pumping lemma, that this language is not a CFL. Let *n* be the pumping-lemma constant, and consider $Z = a^nb^nc^n$.

7. 7.4.1a

If there is any string at all that can be ``pumped,'' then the language is infinite. Thus, let *n* be the pumping-lemma constant. If there are no strings as long as *n*, then surely the language is finite. However, how do we tell if there is some string of length *n* or more? If we had to consider all such strings, we'd never get done, and that would not give us a decision algorithm.

The trick is to realize that if there is any string of length *n* or more, then there will be one whose length is in the range *n* through *2n-1*, inclusive. For suppose not. Let *z* be a string that is as short as possible, subject to the constraint that $|z| >= n$. If $|z| < 2n$, we are done; we have found a string in the desired length range. If $|z| >= 2n$, use the pumping lemma to write $z = uvwxy$. We know uwy is also in the language, but because $|vwx| <= n$, we know $|z| > |uwy| >= n$. That contradicts our assumption that z was as short as possible among strings of length n or more in the language.

We conclude that $|z| < 2n$. Thus, our algorithm to test finiteness is to test membership of all strings of length between n and $2n-1$. If we find one, the language is infinite, and if not, then the language is finite.

8. 7.4.3a

Here is the table:

```
   {S,A,C}
   {B}      {B}
   {B}      {S,C}    {B}
   {S,C}    {S,A}    {S,C}    {S,A}
   {A,C}    {B}      {A,C}    {B}      {A,C}
------------------------------------------
     a        b        a        b        a
```

Since S appears in the upper-left corner, *ababa* is in the language.

9. 7.4.4

The proof is an induction on n that if $A =>^* w$, for any variable A, and $|w| = n$, then all parse trees with A at the root and yield w have $2n-1$ interior nodes.

Basis: $n = 1$. The parse tree must have a root with variable A and a leaf with one terminal. This tree has $2n-1 = 1$ interior node.

Induction: Assume the statement for strings of length less than n, and let $n > 1$. Then the parse tree begins with A at the root and two children labeled by variables B and C. Then we can write $w = xy$, where $B =>^* x$ and $C =>^* y$. Also, x and y are each shorter than length n, so the inductive hypothesis applies to them, and we know that the parse trees for these derivations have, respectively, $2|x|-1$ and $2|y|-1$ interior nodes.

Thus, the parse tree for $A =>^* w$ has one (for the root) plus the sum of these two quantities number of interior nodes, or $2(|x|+|y|-1)$ interior nodes. Since $|x|+|y| = |w| = n$, we are done; the parse tree for $A =>^* w$ has $2n-1$ interior nodes.

10. E → TR, R → +TR | Λ, T → FV, V → *FV | Λ, F→ (E) | a