SOLUTIONS HW#4

1. We can show that the right-linear grammar can be reduced at a strongly right-linear grammar. Then it is sufficient to prove that the strongly right-linear grammar generates the regular sets. The same thing happens also for the "left" grammars.

So it is sufficient to prove that given a strongly right-linear grammar G we can always construct an NFA: M = (Q, Σ, δ, s, F) such that L(M) = L(G). We define:

Q = N$\cup${f}, where f is an accept state, and N is the set of nonterminals
Σ is the set of terminals
$\delta(A, x) = \{B \mid A \rightarrow xB \in P\}$
$\delta(A, e) = \{f \mid A \rightarrow e \in P\}$
S = S
F = {f}

The proof can be done by induction on the length of the derivation.

2. (b) We have the CFG: S→aSa | aBa   B→bB | b

<u>Chomsky:</u>
S→AC | AD
B→EB | b
C→SA
D→BA
E→b
A→a

<u>Greibach:</u>
S→aSA | aBA
B→bB | b
A→a

(d) We have the CFG:
S→aSa | bSb | aSb | bSa| aCb | bCa | aDb | bDa | ab | ba
C→aC | e
D→bD |e

<u>Chomsky:</u>
S→AE_1 | BE_2 | AE_2 | BE_1 | AE_3 | BE_4 | AE_5 | BE_6 | AB |BA
E_1→SA
E_2→SB
E_3→CB

$E_4 \rightarrow CA$
$E_2 \rightarrow DB$
$E_2 \rightarrow DA$
$C \rightarrow AC \mid a$
$D \rightarrow BD \mid b$
$A \rightarrow a$
$B \rightarrow b$

<u>Greibach:</u>
$S \rightarrow aSA \mid bSB \mid aSB \mid bSA \mid aCB \mid bCA \mid aDB \mid bDA \mid aB \mid bA$
$C \rightarrow aC \mid a$
$D \rightarrow bD \mid b$
$A \rightarrow a$
$B \rightarrow b$

3. A CFG that generates all strings with exactly twice as many a's as b's is:
$S \rightarrow ACB \mid ABC \mid bCC \mid e$
$C \rightarrow AS \mid bCCC$
$B \rightarrow ACBB \mid ABAB \mid ABBC \mid bS$
$A \rightarrow a$

The proof that this CFG is correct can be done by induction. It is easy to see that S generates strings for which #a's = 2#b's, C generates strings for which #a's = 2#b's +1 and B generates strings for which #a's = 2#b's -2. If we define f(x) = #a(x) – 2#b(x), then the graph of the function f(y) for all the prefixes of x will have jumps equal to either -2 or +1. This implies that we can get from a string for which the value of f is m to another one for which the value of f is n>m by going through all the intermediate values m+1, m+2,…, n-1. So we will have a concatenation of n-m strings for which the value of f is equal to 1.

Let us prove now the reverse implication. If x is a string for which #a's = 2#b's then its length must be multiple of 3. We will prove that it can be generated by induction on the number of a's. Suppose we can generate all the sequences y for which $f(y)\in\{-2, 0, 1\}$ and contain less than or k a's. We want to prove now that we can generate all the sequences with k+1 a's for which $f(x))\in\{-2, 0, 1\}$. If f(x)=0 and the $x=bx_1$ then $f(x_1)=2$. So from what we stated above, since f(e)=0, we can find two substrings x_2 and x_3 that concatenate to give x_1 and $f(x_i)=1$ for i=1,2. So we can get x using bCC, because we can generate x_i from the induction hypothesis. Now if $x=ax_1$ then $f(x_1)=-1$. But we can write $x_1=x_2x_3$ where x_2 and x_3 are two sequences with at most k a's, one for which f is 1 and the other one for which the value of f is -2 also because of the fact that we could only have upward jumps equal to 1.

Now we also have the other two situations. If f(x) = 1. If $x=bx_1$ then $f(x_1)=3$. So we could write x_1 as a concatenation of 3 strings for which f is equal to 1. We are in the situation $C \rightarrow bCCC$. If $x=ax_1$ then $f(x_1)=0$ and x_1 has at most k a's so we can generate x_1 and we are in the situation $C \rightarrow aS$.

If $f(x)=-2$ and $x=bx_1$ then $f(x_1)=0$ so we are in the situation described above and we know that we can generate x_1. If $x=ax_1$ then $f(x_1)=-3$ then in order to get from e to x_1, so for the value of the function f from 0 to -3 we need two down jumps and une up, which corresponds to one of the following possibilities: CBB, BCB, BBC and all are covered in our CFG and with the induction hypothesis.

So we proved by induction that S generates all the sequences we need, but taking into account the strings generated by B and C.

$M = (Q, \Sigma, G, \delta, s, \perp, F)$
$Q = \{p,q, f\}$
$\Sigma = \{a, b\}$
$G = \{+, -, \perp\}$
$S = p$
$F = \{f\}$

1. $(p, a, \perp)$ $(p, -\perp)$
2. $(p, a, -)$ $(p, --)$
3. $(p, a, +)$ (p, e)
4. $(p, b, \perp)$ $(p, ++\perp)$
5. $(p, b, -)$ (q, e) Go to state q to check for the next stack symbol.
6. $(p, b, +)$ $(p, +++)$

7. $(q, e, \perp)$ $(p, +)$ Handle the extra + from the b that was just read.
8. $(q, e, -)$ (p, e)

9. $(p, e, \perp)$ $(f, \perp)$ Accept if the stack has no minuses.
10. $(p, e, +)$ $(f, +)$

Lemma: If the stack contents at a given step are the string a, then a is either $+^n \perp$ or $-^n \perp$, where $n \geq 0$.
Proof: Induction on the length of the derivation. Each transition preserves this rule.

Lemma: Let a be the contents of the stack after reading input x and suppose that the NPDA is in state p. If $a = +^n \perp$, then $2\#b(x) - \#a(x) = n$. If $a = -^n \perp$, then $2\#b(x) - \#a(x) = -n$.
Proof: Again use induction on the length of the derivation. Each transition preserves this rule.

Since the NPDA accepts exactly when the stack is of the form $+^n \perp$ for some $n \geq 0$, we see from the two lemmas that it accepts exactly the given language!

The strings in the examples are accepted, accepted, not accepted.

4. We can prove by showing that there exists a PDA that accepts $A \cap R$. Since A is CFL, there is a PDA that accepts A: $M_1 = (Q_1, \Sigma, G_1, \delta_1, s_1, \perp, F_1)$. Since R is a regular set, there is a DFA that accepts R: $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$.

We define the new PDA: $M_1 = (Q_3, \Sigma, G_3, \delta_3, s_3, \perp, F_3)$ where:

$Q = Q_1 \times Q_2$
Σ is the same as above
$G_3 = G_1$ since M_2 does not have a stack
$s_3 = (s_1, s_2)$
$F_3 = F_1 \times F_2$
$\delta_3((p, q), a, A) = (\delta_1(p, a), \delta_2(q, a, A))$

It can be proved by induction that $\delta_3((p, q), x, A) = (\delta_1(p, x), \delta_2(q, x, A))$. The induction is after $|x|$.

Now we need $L(M_3) = A \cap R$.

$x \in L(M_3) \Leftrightarrow$
$\delta_3(s_3, x, \perp) \in (F_3, G^*) \Leftrightarrow$
$(\delta_1(s_1, x), \delta_2(s_2, x, \perp)) \in (F_1 \times F_2, G^*) \Leftrightarrow$
$\delta_1(s_1, x) \in F_1, \delta_2(s_2, x, \perp) \in (F_2, G^*) \Leftrightarrow$
$x \in L(R)$ and $x \in L(A) \Leftrightarrow$
$x \in L(R) \cap L(A) = L(A \cap R)$

5. We prove the problem by contradiction. Suppose L_1 and L_2 are two CFLs. From the previous problem, the intersection of a CFL and a regular set is a CFL. So if we find some L_1, L_2 CFLs and R a regular set such that $(L_1||L_2) \cap R$ is not CFL, then $L_1||L_2$ is not a CFL.

Let us choose:

$L_1 = \{a^n b^n \mid n \geq 0\}$
$L_2 = \{a^m b^m \mid m \geq 0\}$
$R = a^*c^*b^*d^*$

L_1 and L_2 are CFLs and R is a regular set.
$L_1||L_2 = \{x \in (a^*b^*c^*d^*)^*$ such that $\#a(x) = \#b(x)$ and $\#c(x) = \#d(x)\}$

But $(L_1||L_2) \cap R = \{a^n c^m b^n d^m \mid m, n \geq 0\} = A$. This is not a CFL. This can be proved using pumping lemma. Say the demon picks k. You pick $z = a^k c^k b^k d^k$. Call each of the substrings a^k, c^k, b^k, d^k a *block*. $z \in A$ and $|z| \geq k$. Say the demon picks u, v, w, x, y such that

$z=uvwxy$, $vx \neq e$ and $|vwx|<k$. No matter what the demon does, you can win by picking $i=2$.

i. If one of v or x contains two different letters then uv^2wx^2y is not of the form a*c*b*d*
ii. If v and x are from the same block then uv^2wx^2y has one block longer than the other three, so it is not in A.
iii. If v and x are in different blocks, then the blocks must be adjacent, otherwise $|vwx|>k$. But in this case uv^2wx^2y does not have two pairs of blocks of equal length.

This covers all the possibilities. So $L_1||L_2$ is not always a CFL.