Solutions Exam 170816

Here we only give a brief explanation of the solution. Your solution should in general be more elaborated than these ones.

1. Our property is: $P(n)$: if $S \Rightarrow^n w$ then $\#_0(w) \geqslant \#_1(w)$, where $\#_0$ and $\#_1$ are functions counting the number of 0's and 1's in a word, respectively.

   We will use course-of-value/strong induction on the length of the derivation (number of steps) $S \Rightarrow^n w$.

   Base cases: $S \Rightarrow w$, hence the rule applied should have been $S \to 010$ or $S \to 10$. Hence $w = 010$ or $w = 10$. We have both that $\#_0(010) \geqslant \#_1(010)$ and that $\#_0(10) \geqslant \#_1(10)$, so the property holds in both cases.

   Step case: Our IH is: if $S \Rightarrow^k w$ in at most $n > 0$ steps ($1 \leqslant k \leqslant n$) then $\#_0(w) \geqslant \#_1(w)$.

   Let $S \Rightarrow^{n+1} w$ with $n > 0$.

   Since $n > 0$ then the first rule applied should have been $S \to 0S$ or $S \to S1S0$.

   In the case the first rule was $S \to 0S$ then $w = 0w_1$ with $S \Rightarrow^n w_1$. Then the IH applies to w_1 and we know that $\#_0(w_1) \geqslant \#_1(w_1)$. Hence $\#_0(w) = 1 + \#_0(w_1) \geqslant \#_0(w_1) \geqslant \#_1(w_1) = \#_1(w)$.

   In the case the first rule was $S \to S1S0$ then $w = w_1 1 w_2 0$ with $S \Rightarrow^i w_1$, $S \Rightarrow^j w_2$, and $1 \leqslant i, j \leqslant n$. Then the IH applies to both w_1 and w_2, hence we know that $\#_0(w_1) \geqslant \#_1(w_1)$ and $\#_0(w_2) \geqslant \#_1(w_2)$. Hence $\#_0(w) = 1 + \#_0(w_1) + \#_0(w_2) \geqslant 1 + \#_1(w_1) + \#_1(w_2) = \#_1(w)$.

2. (a)

	0	1
$\to^* q_0$	q_1	q_0
$^*q_1$	q_1	q_2
$^*q_2$	q_1	q_3
$^*q_3$	q_1	q_4
$^*q_4$	q	q_4
q	q	q

   (b) $1^*(0(\epsilon + 1 + 11))^*1^*$

3. (a) The set of equations is:

$$\begin{aligned} E_0 &= 1E_0 + 0(E_1 + E_2) & \qquad E_3 &= 0E_5 \\ E_1 &= 1E_1 + 0(E_3 + E_4) & \qquad E_4 &= 1E_5 \\ E_2 &= 0E_2 + 1(E_3 + E_4) & \qquad E_5 &= (0+1)E_5 + \epsilon \end{aligned}$$

   The solution to E_5 is $E_5 = (0+1)^*$ and hence we obtain $E_3 = 0(0+1)^*$ and $E_4 = 1(0+1)^*$. Now we get

$$E_1 = 1E_1 + 0(0+1)(0+1)^* \qquad E_2 = 0E_2 + 1(0+1)(0+1)^*$$

   so we have

$$E_1 = 1^*0(0+1)^+ \qquad E_2 = 0^*1(0+1)^+$$

   and finally

$$E_0 = 1E_0 + 0(1^*0 + 0^*1)(0+1)^+ \Rightarrow E_0 = 1^*0(1^*0 + 0^*1)(0+1)^+$$

(b)

	0	1
$\rightarrow q_0$	q_1q_2	q_0
q_1q_2	$q_2q_3q_4$	$q_1q_3q_4$
$q_2q_3q_4$	q_2q_5	$q_3q_4q_5$
$q_1q_3q_4$	$q_3q_4q_5$	q_1q_5
$^*q_2q_5$	q_2q_5	$q_3q_4q_5$
$^*q_1q_5$	$q_3q_4q_5$	q_1q_5
$^*q_3q_4q_5$	q_5	q_5
$^*q_5$	q_5	q_5

4. (a) q_2 and q_4 are not accessible so we eliminate them.
Then we need to run the algorithm that identifies equivalent states.

	q_0	q_1	q_3
q_5	X	X	
q_3	X	X	
q_1			

The resulting automaton is:

	0	1
$\rightarrow q_0q_1$	q_0q_1	q_3q_5
$^*q_3q_5$	q_3q_5	q_3q_5

(b) $0^*1(0+1)^*$

5. (a) The expressions generate the same language and I will show it with double inclusion:
   - $(1(0^*+010^*))^* \subseteq \epsilon + 1(0+1)^*$: $(1(0^*+010^*))^*$ generates either the empty string (ϵ) or a string that starts with 1. After that 1 it comes a sequence of 0's and 1's. Whatever that sequence is, it will also be generated by $(0+1)^*$; so any non-empty string in $(1(0^*+010^*))^*$ will be generated by $1(0+1)^*$.
   - $\epsilon + 1(0+1)^* \subseteq (1(0^*+010^*))^*$: ϵ is clearly generated by $(1(0^*+010^*))^*$.
   String generated by $1(0+1)^*$ start with 1 and then follows a sequence of 0's and 1's in any order. Observe that non-empty string generated by $(1(0^*+010^*))^*$ start with 1. Now, if a sequence of 0's comes after that 1, we can generate it with 0^*. If we instead need 1's, what we actually do is generate ϵ with 0^* and then the next iteration of the outermost * will give us the 1. Repeating this procedure we will be generating the 0's and 1's in the sequence generated from $(0+1)^*$.

   (b) (1.75pts each)
   i. Let $\mathcal{L}_1 = \{0,1\}^* - \epsilon$ (recall that the difference of two regular languages is also regular) and $\mathcal{L}_2 = \{0^n1^n \mid n \geqslant 0\}$.
   Here $\mathcal{L}_1 \cup \mathcal{L}_2 = \{0,1\}^*$ which is regular and $\mathcal{L}_1 \cap \mathcal{L}_2 = \{0^n1^n \mid n \geqslant 1\}$ which is not regular.
   ii. Let $\mathcal{L}_1 = \epsilon$ and $\mathcal{L}_2 = \{0^n1^n \mid n \geqslant 1\}$.
   Here $\mathcal{L}_1 \cap \mathcal{L}_2 = \emptyset$ which is regular and $\mathcal{L}_1 \cup \mathcal{L}_2 = \{0^n1^n \mid n \geqslant 0\}$ which is not regular.

   Observe that in both cases we have that $\mathcal{L}_1 \nsubseteq \mathcal{L}_2$ and $\mathcal{L}_2 \nsubseteq \mathcal{L}_1$ as required.

6. (a) In the grammar below, S is the starting symbol.

$$S \rightarrow d \mid DS$$
$$D \rightarrow ABc \mid Ac \mid Bc \mid c \mid AbC \mid Ab \mid bC \mid b$$
$$A \rightarrow a \mid aA \qquad B \rightarrow b \mid bB \qquad C \rightarrow c \mid cC$$

(b) D generates strings of the form $(a^*b^*c + a^*bc^*)$ (see below for its explanation).
The production $S \to DS$ produces as many of those strings as needed.
The production $S \to d$ will terminate the production of such strings or it could be used to simply generate the string d.

A generates a^+, B generates b^+, and C generates c^+.

Since we are not allowed to use ϵ-productions, D will generate a^*b^*c via ABc, Ac, Bc or c and a^*bc^* via AbC, Ab, bC, c.

(c) The leftmost derivation is:
$S \Rightarrow DS \Rightarrow ABcS \Rightarrow aABcS \Rightarrow aaBcS \Rightarrow aabBcS \Rightarrow aabbcS \Rightarrow aabbcd$

(d) A grammar is ambiguous if there are two different leftmost derivations, or two different rightmost derivations or two different parse trees for at least one word in the language.

(e) Yes, $abcd$ can be derived with the following two leftmost derivations:
$S \Rightarrow DS \Rightarrow ABcS \Rightarrow aBcS \Rightarrow abcS \Rightarrow abcd$
$S \Rightarrow DS \Rightarrow AbCS \Rightarrow abCS \Rightarrow abcS \Rightarrow abcd$

(f) A grammar in CNF has rules only of the form $A \to a$ or $A \to BC$, that is, a variable to a single symbol of the alphabet or a variables to two variables.

(g) No. The grammar just consisting of $S \to \epsilon$ has no equivalent grammar in CNF. On the other hand, any grammar generating a language that doesn't contain the empty word can be converted into an equivalent grammar in CNF.

(h)

$$
\begin{array}{ll}
S \to d \mid DS & \\
D \to PZ \mid AZ \mid BZ \mid c \mid QC \mid AY \mid YC \mid b & \\
A \to a \mid XA & X \to a \\
B \to b \mid YB & Y \to b \\
C \to c \mid ZC & Z \to c \\
P \to AB & Q \to AY
\end{array}
$$

7. (a) See slide 26 lecture 13.

(b) Let us assume our language $\mathcal{L} = \{ww \mid w \in \{0,1\}^*\}$ is context-free.
Hence the PL should apply.

Let n be the constant given by the PL.
Let $w = 0^n1^n0^n1^n$. We have that $w \in \mathcal{L}$ and that $|w| \geqslant n$.

Hence $w = xuyvz$ with $uv \neq \epsilon$ and $|uyv| \leqslant n$.

This means that uyv can contain at most 2 of the 4 "set of symbols in the word (meaning the first or second set of 0's, and the first or second set of 1's).

If uyv contains only one set of symbols, then xu^kyv^kz for $k > 1$ will be adding symbols in that particular set (the first or the second for that symbols) but not in the other set which correspond to the same symbol (the second or the first, respectively). Hence the resulting word will not be of the form $w'w'$.

Observe that if uyv contains two set of symbols, those sets should be sets of different symbols and not the two sets for a particular symbol! Then xu^kyv^kz for $k > 1$ will be adding symbols in those two sets of different symbols but not in the corresponding two other sets for those symbols. Hence again, the resulting word will not be of the form $w'w'$.

Then, $\mathcal{L}$ cannot be context-free.

8.

$\{S,A,B,C\}$					
$\{S,A,B,C\}$	$\{S,A,B,C\}$				
$\{S,A,B,C\}$	$\{S,A\}$	$\{S,A,B,C\}$			
$\{S,C\}$	$\{S,A\}$	$\{S,A\}$	$\{S,B\}$		
$\{S,C\}$	$\{B\}$	$\{S,A\}$	$\{C\}$	$\{S,B\}$	
$\{A\}$	$\{B\}$	$\{B\}$	$\{C\}$	$\{C\}$	$\{A\}$
a	b	b	c	c	a

S belongs to the upper-most set, which means that the word is generated by the grammar since S is the starting symbol of the grammar.

9. Let $X \in \{a, b\}$ and let $M = (\{q_0, q_1, q_2, q_f\}, \Sigma, \delta, q_0, \square, \{q_f\})$, with δ as follows:

   In q_0 we have read an empty sequence of a's and b's, in q_1 and odd sequence, and in q_2 an non-empty even sequence.

$\delta(q_0, X) = (q_1, X, \mathsf{R})$	If q_0 finds an a or a b then M moves to q_1 which will check that there is another a or b in the tape;
$\delta(q_1, X) = (q_2, X, \mathsf{R})$	If q_1 reads an a or a b then it moves to q_2 which will check that the rest of the tape has an even nr of a's and b's;
$\delta(q_2, X) = (q_1, X, \mathsf{R})$	If q_2 finds an a or a b then it moves to q_3 which will check that there is another a or b in the tape;
$\delta(q_2, \square) = (q_f, \square, \mathsf{R})$	If q_2 reads the empty symbol then the tape is non-empty and has an even nr of a's and b's and it should be accepted;

   Observe that if the TM read any other symbol from Σ other than a or b then it will get stuck and not accept the tape. Similarly, if it reads $\square$ in any other state but q_2.