Solutions Exam 180529

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: $P(n) : \forall w$, if $S \Rightarrow^n w$ then $n \leqslant \#_b(w) \leqslant \#_a(w) \leqslant 2n$.

   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 $S \to aba$ or $S \to ab$.

   Here, $1 \leqslant 1 = \#_b(aba) \leqslant \#_a(aba) = 2 \leqslant 2$ and $1 \leqslant 1 = \#_b(ab) \leqslant \#_a(ab) = 1 \leqslant 2$, which proves the base cases.

   Step case: Our IH is: $\forall w$, if $S \Rightarrow^k w$ with $1 \leqslant k \leqslant n$, then $n \leqslant \#_b(w) \leqslant \#_a(w) \leqslant 2n$.

   Let $S \Rightarrow^{n+1} w$ with $n > 0$. We need to prove $P(n+1)$, that is, $\forall w$, if $S \Rightarrow^{n+1} w$ then $n+1 \leqslant \#_b(w) \leqslant \#_a(w) \leqslant 2(n+1)$.

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

   In the case the first rule was $S \to aSb$ then we have that $S \Rightarrow aSb \Rightarrow^n w$ so $w = aw_1b$ with $S \Rightarrow^n w_1$. Then the IH applies to n and we know that $n \leqslant \#_b(w_1) \leqslant \#_a(w_1) \leqslant 2n$. Now
   $n+1 \leqslant \#_b(w_1) + 1 \leqslant \#_a(w_1) + 1 \leqslant 2n+1$ which gives us that
   $n+1 \leqslant \#_b(w) \leqslant \#_a(w) \leqslant 2n+1 \leqslant 2(n+1)$ as desired.

   In the case the first rule was $S \to aSbSa$ then we have that $S \Rightarrow aSbSa \Rightarrow^n w$ so $w = aw_1bw_2a$ with $S \Rightarrow^i w_1$, $S \Rightarrow^j w_2$, $1 \leqslant i, j \leqslant n$ and $i + j = n$. Then the IH applies to both i and j so we know that $i \leqslant \#_b(w_1) \leqslant \#_a(w_1) \leqslant 2i$ and $j \leqslant \#_b(w_2) \leqslant \#_a(w_2) \leqslant 2j$. Then
   $i + j = n \leqslant \#_b(w_1) + \#_b(w_2) \leqslant \#_a(w_1) + \#_a(w_2) \leqslant 2(i+j) = 2n$. From this we get
   $n+1 \leqslant \#_b(w_1) + \#_b(w_2) + 1 \leqslant \#_a(w_1) + \#_a(w_2) + 1 \leqslant 2n+1$ and
   $n+1 \leqslant \#_b(w_1) + \#_b(w_2) + 1 \leqslant \#_a(w_1) + \#_a(w_2) + 2 \leqslant 2n+2$. Hence
   $n+1 \leqslant \#_b(w) \leqslant \#_a(w) \leqslant 2(n+1)$ as desired.

2. We define a NFA:

	0	1
$\to q_0$	$\{q_1, q_3\}$	$\{q_4\}$
q_1	$\emptyset$	$\{q_0, q_2\}$
q_2	$\{q_3\}$	$\{q_0, q_4\}$
q_3	$\{q_3\}$	$\{q_4\}$
$^*q_4$	$\emptyset$	$\emptyset$

3. (a) $0^*1 + 0^*10(0+1)^* + 0^*10^*1 + 0^*10^*1(0+1)^+ = 0^*1(\epsilon + 0(0+1)^* + 0^*1 + 0^*1(0+1)^+) = 0^*1(\epsilon + 0(0+1)^* + 0^*1(\epsilon + (0+1)^+))$

   (b)

	0	1
$\to q_0$	q_0	$q_1q_2q_3$
$^*q_1q_2q_3$	q_2q_5	q_3q_4
$^*q_2q_5$	q_2q_5	$q_3q_4q_5$
$^*q_3q_5$	q_5	q_5
$^*q_3q_4q_5$	q_5	q_5
$^*q_5$	q_5	q_5

4. I will solve equations:

$$\begin{array}{ll}
E_0 = 0E_0 + 1E_1 + 2E_2 & E_0 = 0E_0 + 1E_1 + 20^*2E_4 \\
E_1 = 0E_3 + 1E_4 = 0(0+1+2)^* + 1E_4 & E_1 = 0E_3 + 1E_4 = 0(0+1+2)^* + 1E_4 \\
E_2 = 0E_2 + 2E_4 = 0^*2E_4 & \\
E_3 = (0+1+2)E_3 + \epsilon = (0+1+2)^* & \\
E_4 = 2E_1 + 1E_3 = 2E_1 + 1(0+1+2)^* & E_4 = 20(0+1+2)^* + 21E_4 + 1(0+1+2)^* \\
 & \quad = (21)^*(20+1)(0+1+2)^*
\end{array}$$

Hence $E_1 = 0(0+1+2)^* + 1(21)^*(20+1)(0+1+2)^* = (0+1(21)^*(20+1))(0+1+2)^*$ and $E_0 = 0E_0 + 1(0+1(21)^*(20+1))(0+1+2)^* + 20^*2(21)^*(20+1)(0+1+2)^*$. So

$$E_0 = 0^*(10 + (11 + 20^*2)(21)^*(20+1))(0+1+2)^*$$

5. (a) q_4 is not reachable so we eliminate it before the do the table.

	q_0	q_1	q_2	q_3	q_5
q_6	X	X	X	X	X
q_5	X	X		X	
q_3	X		X		
q_2	X	X			
q_1	X				

(b) The equivalent classes are $\{q_0\}, \{q_1, q_3\}, \{q_2, q_5\}, \{q_6\}$. Recall that q_4 has already been eliminated for not being reachable. q_6 is a dead state but it shall not be eliminated!

(c) The resulting automaton is:

	a	b
$\rightarrow^*\ q_0$	q_1q_3	q_1q_3
q_1q_3	q_2q_5	q_1q_3
$^*q_2q_5$	q_6	q_2q_5
q_6	q_6	q_6

(d) $\epsilon + (a+b)b^*ab^*$

6. (a) $\mathcal{L}_1 = 01 + 0^*$ and $\mathcal{L}_2 = \{0^n1^n \mid n \geqslant 0\}$.
Here $\mathcal{L}_1 \cap \mathcal{L}_2 = \{01\}$ and $\mathcal{L}_1 \cup \mathcal{L}_2 = \mathcal{L}_2 \cup \{0^n \mid n \geqslant 0\}$. It is easy to show that $\mathcal{L}_1 \cup \mathcal{L}_2$ is not regular. The proof is similar to that showing $\mathcal{L}_2$ is not regular: by choosing $w = 0^m1^m$ with m the constant of the pumping lemma, we can show that for any $k > 1$, xy^kz will neither be in $\mathcal{L}_2$ nor in $\{0^n \mid n \geqslant 0\}$.

(b) $\mathcal{L}_1 = \Sigma^* - \epsilon$ (recall regular languages are closed under difference) and $\mathcal{L}_2 = \{0^n1^n \mid n \geqslant 0\}$.
$\mathcal{L}_1 \cap \mathcal{L}_2 = \{0^n1^n \mid n \geqslant 1\}$ and $\mathcal{L}_1 \cup \mathcal{L}_2 = \Sigma^*$.

7. The expressions generate the same language and I will show it with double inclusion:

- $((a^* + aba^*)b)^* \subseteq \epsilon(a+b)^*b$: $((a^* + aba^*)b)^*$ generates either the empty string (ϵ) or a string that ends with b. Before that b comes a sequence of a's and b's. Whatever that sequence is, it can be generated by $(a+b)^*$; so any non-empty string in $((a^* + aba^*)b)^*$ will be generated by $(a+b)^*b$.
- $\epsilon + (a+b)^*b \subseteq ((a^* + aba^*)b)^*$: ϵ is clearly generated by $((a^* + aba^*)b)^*$. String generated by $(a+b)^*b$ end with b and before that they contain a sequence of a's and b's in any order. Observe that non-empty string generated by $((a^* + aba^*)b)^*$ end with b. Observe also that $(a^*b)^* \subseteq ((a^* + aba^*)b)^*$ and that with $(a^*b)^*$ we can generate any sequence of a's and b's which ends with b: for each b we need in the sequence we repeat the outmost closure and generate ϵ with the a^* part of a^*b.

8. (a)

$$S \to AB \qquad A \to a \mid aA \qquad B \to aBc \mid bB \mid ac \mid b$$

(b) A generates the sequence of one or more a's at the beginning of the word and B generates the rest of the word.

Every a in w should generate a c in the end, this is done with the productions generating aBc or ac (for the case we are done). We can also add any number of b's with the productions generating bB or b (for the case we are done).

(c) See slides 15-16 in lecture 12.

(d) Observe that the separation of the number of a's in A and in B is given by the number of c's: any a in B should correspond to a c while an a in A will not. Then there no ambiguity in whether an a should come from A or from B.

There is only one way to generate a sequence of a's: either by generating the last a in the sequence, or by adding one a to a sequence of a's.

To construct the rest of the word we follow the order of the sequence of a's and b in w. If we need to generate an a then we should use the production aBc if we are not done, or ac if we are done. If we need to generate a b then we should use the production bB if we are not done, or b if we are done. So the choice of productions to use in here is unique.

(e) (1pt) Leftmost derivation: $S \Rightarrow AB \Rightarrow aB \Rightarrow aaBc \Rightarrow aabBc \Rightarrow aabacc$.

(2pts) Recursive inference:

i. a belongs to the language of A because $A \to a$;
ii. ac belongs to the language of B because $B \to ac$;
iii. bac belongs to the language of B because $B \to bB$ and ii);
iv. $abacc$ belongs to the language of B because $B \to aBc$ and iii);
v. $aabacc$ belongs to the language of s because $S \to AB$, i) and iv);

(f)

$$\begin{array}{lll} S \to AB & A \to a \mid XA & X \to a \\ B \to XT \mid YB \mid XZ \mid b & T \to BZ & Y \to b \quad Z \to c \end{array}$$

9. (a) See slide 10 lecture 14.

(b) Let us assume our language $\mathcal{L}$ is context-free. Hence the PL should apply.

Let n be the constant given by the PL.
Let $w = a^n b^{n+1} c^{n+2}$. We have that $w \in \mathcal{L}$ and that $|w| \geqslant n$.

Hence $w = xuyvz$ with $uv \neq \epsilon$ and $|uyv| \leqslant n$. Then uv will contain only one or two of the letters, but never the three of them.

If uv contains only a's (resp b's), then any $k > 1$ will add at least an a (resp b) so that the number of a's (resp b's) in $xu^k yv^k z$ will no longer be strictly smaller than the number of b's (res c's) in the word.

If uv contains only c's then with $k = 0$ we are removing at least a c and hence the number of c's in $xu^k yv^k z$ will no longer be strictly greater then the number of b's in the word.

If uv contains both a's and b's then, for $k > 1$, we will add at least a b so that the number of b's in $xu^k yv^k z$ will no longer be strictly smaller than the number of c's in the word.

If uv contains both b's and c's then, for $k = 0$, we will remove at least a b and hence the number of b's in $xu^k yv^k z$ will no longer be strictly greater then the number of a's in the word.

Since these are all possible cases for uv and no of them work, then $\mathcal{L}$ cannot be context-free.

10.

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

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.

11. Let $M = (\{q_0, ..., q_6, q_f\}, \{a, b, c\}, \delta, q_0, \square, \{q_f\})$, with δ is as follows:

$\delta(q_0, a) = (q_0, a, R)$	any number of a's can be present at the beginning;
$\delta(q_0, c) = (q_1, c, R)$	we read a c, now it can come another c or a b;
$\delta(q_1, c) = (q_2, c, R)$	we read another c, so after the b's we need to read two c's!
$\delta(q_1, b) = (q_3, b, R)$	we read a b, now more b's can come but then only a single c;
$\delta(q_2, b) = (q_4, b, R)$	we read a b, now more b's can come and then two c's;
$\delta(q_3, b) = (q_3, b, R)$	any number of extra b's is fine;
$\delta(q_3, c) = (q_6, c, R)$	we read the only c we were expecting, now only a blank symbol can come;
$\delta(q_4, b) = (q_4, b, R)$	any number of extra b's is fine;
$\delta(q_4, c) = (q_5, c, R)$	we read the first of the two c's we were expecting;
$\delta(q_5, c) = (q_6, c, R)$	we read the second c we were expecting, now only a blank symbol can come;
$\delta(q_6, \square) = (q_f, \square, R)$	we have a right tape so we accept.

The Turing decider moves only to the right. It will halt when the sequence of symbols is not correct, or it will end up in the final state when the sequence is correct and needs to be accepted.

q_0 reads any number of a's.
If $w = c$ then the TM moves $q_1 \to q_3 \to q_6$. Here q_3 checks that at least a b comes, and even read extra b's.
If $w = cc$ then the TM moves $q_1 \to q_2 \to q_4 \to q_5 \to q_6$. Here q_4 checks that at least a b comes, and even read extra b's.
q_6 makes sure nothing else comes after the last c in w.