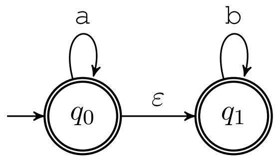
Problem 1 (25 points)

(a) (10 points) Convert the following NFA (over $\{\mathtt{a},\mathtt{b}\}$) into a DFA.



(b) (5 points) Give a regular expression for the language of the above NFA.

(c) (10 points) Give a DFA for the following language:

$$L = \{w \in \{\mathtt{a},\mathtt{b}\}^* \mid w \neq \varepsilon, \text{ and } w \text{ begins and ends with different symbols}\}$$

Solution:

Solutions and common mistakes for Q1 will be added later, once the TA responsible for grading this question has finished typesetting them.

Problem 2 (20 points) Consider the language

$$L = \{\mathtt{a}^i\mathtt{b}^j\mathtt{a}^k \mid j \leqslant i + k\}.$$

(a) (10 points) Give a pushdown automaton for L. Briefly explain how your pushdown automaton works (insufficient explanation will get no points).

(b) (10 points) Consider the language

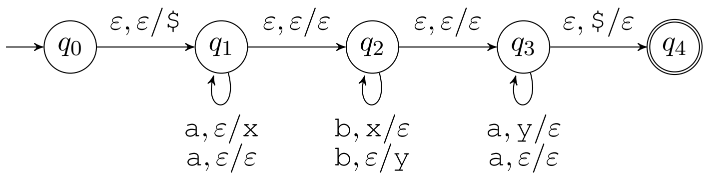
$$L' = \{w \in L \mid w \text{ is a palindrome}\}.$$

Show that L' is not context-free. Recall that a palindrome reads the same forward and backward.

Solution:

(a) Our PDA below accepts any string $w = \mathtt{a}^i\mathtt{b}^j\mathtt{a}^k$ such that $j \leqslant i+k$, because we can rewrite $w = \mathtt{a}^i\mathtt{b}^p\mathtt{b}^q\mathtt{a}^k$ where $p + q = j$, $0 \leqslant p \leqslant i$ and $0 \leqslant q \leqslant k$ (for example, take $p = \min\{i, j\}$ and $q = j - p$). The PDA reads i a's at q_1 while pushing p x's there, reads $j = p + q$ b's at q_2 by first popping p x's and then pushing q y's there, and reads k a's at q_3 while popping q y's there.

Conversely, any string accepted by the PDA belongs to L. Indeed, any string accepted by the PDA has the form $\mathtt{a}^i\mathtt{b}^j\mathtt{a}^k$, because q_1, q_2, q_3 reads a's, b's, a's in that order. Exactly p x's are pushed at q_1 ($p \leqslant i$), p' x's are popped at q_2, q' y's are pushed at q_2 ($p' + q' = j$), and q y's are popped at q_3 ($q \leqslant k$). Since the stack is empty upon reaching the accepting state, $p = p'$ and $q = q'$, we have $j = p' + q' = p + q \leqslant i + k$.



Many students gave a PDA that erronously accepts any string whose number of b is at most twice the number of a, regardless of the ordering of a's and b's (such as abababab).

(b) L' is the same as $\{\mathtt{a}^i\mathtt{b}^j\mathtt{a}^i \mid j \leqslant 2i\}$. For every nonnegative integer (pumping length) m, consider $s = \mathtt{a}^m\mathtt{b}^{2m}\mathtt{a}^m \in L'$. If L' were context-free, the pumping lemma for context-free languages splits s into $uvwxy$ such that $|vwx| \leqslant m$, $|vx| \geqslant 1$, and for any $i \geqslant 0$ we have $uv^iwx^iy \in L'$. Consider three cases:

(i) v or x contains an a from the beginning $\mathtt{a}^m$ block: pumping down (take $i = 0$) gives us uwy of the form $\mathtt{a}^p\mathtt{b}^q\mathtt{a}^m$ where $p < m$, so this string is not in L'.

(ii) v or x contains an a from the ending $\mathtt{a}^m$ block: this is similar to the previous case.

(iii) both v and x contain only b's: pumping up (take $i = 2$) gives us uv^2wx^2y of the form $\mathtt{a}^i\mathtt{b}^q\mathtt{a}^i$ for some $q > 2i$, so this string is not in L'.

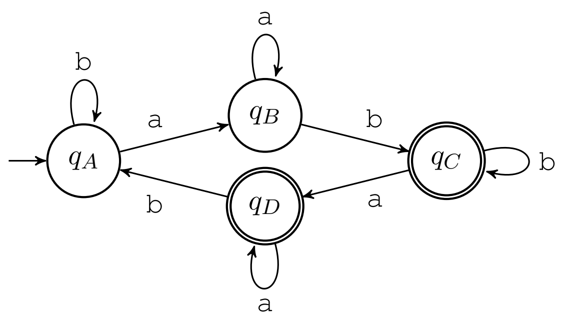
v and x cannot contain a from both the beginning $\mathtt{a}^m$ block and the ending $\mathtt{a}^m$ block, since $|vwx| \leqslant m$. In all cases, we get a string not in L', so L' cannot be context-free.

Many students claimed that in case (i) if we pump up (take $i = 2$), we also get a string of the form $\mathtt{a}^p\mathtt{b}^q\mathtt{a}^i$. This is not true, because if $s = \mathtt{aabbbbaa}$ is split so that $x = \mathtt{ab}$, then $uv^2wx^2y = \mathtt{aababbbbaa}$.

Problem 3 (18 points) The language

$$L = \{w \in \{\texttt{a}, \texttt{b}\}^* \mid w \text{ has an odd number of occurrences of the substring } \texttt{ab}\}$$

has a DFA



(a) (6 points) Prove that the above DFA is minimal: For every pair of distinct states, write down a string (in the upper triangular part of the following table) to distinguish them.

	q_A	q_B	q_C	q_D
q_A	$\times$			
q_B	$\times$	$\times$		
q_C	$\times$	$\times$	$\times$	
q_D	$\times$	$\times$	$\times$	$\times$

(b) (2 points) Which strings stop at q_A? Which strings stop at q_B?

(c) (10 points) Give a context-free grammar for L. Briefly explain how your grammar works (insufficient explanation will get no points).

Solution:

(a)

	q_A	q_B	q_C	q_D
q_A	$\times$	b	ε	ε
q_B	$\times$	$\times$	ε	ε
q_C	$\times$	$\times$	$\times$	b
q_D	$\times$	$\times$	$\times$	$\times$

Other solutions are possible.

(b) Solution 1:

q_A: Empty string, or strings with an even number of occurences of `ab` and ending in `b`.

q_B: Strings with an even number of `ab` and ending in `a`.

Solution 2:

Any regular expression that correctly specifies strings ending in q_A and q_B.

Common mistake: Fail to mention that the empty string also stops at q_A.

(c)

Solution 1:

$$A \to \mathtt{a}B \mid \mathtt{b}A$$
$$B \to \mathtt{a}B \mid \mathtt{b}C$$
$$C \to \mathtt{b}C \mid \mathtt{a}D \mid \varepsilon$$
$$D \to \mathtt{a}D \mid \mathtt{b}A \mid \varepsilon$$

Solution 2:

$$S \to C \mid D$$
$$C \to C\mathtt{b} \mid B\mathtt{b}$$
$$D \to D\mathtt{a} \mid C\mathtt{a}$$
$$B \to B\mathtt{a} \mid A\mathtt{a}$$
$$A \to A\mathtt{b} \mid D\mathtt{b} \mid \varepsilon$$

Both CFGs simulate the given DFA. In Solution 1, variable B generates all strings that, if the DFA starts at q_B, stops at an accepting state. Other variables are analogous.

Solution 2 simulates accepting paths in the DFA backwards. For instance, the path $q_A \xrightarrow{\mathtt{a}} q_B \xrightarrow{\mathtt{b}} q_C \xrightarrow{\mathtt{a}} q_D$ corresponds to the derivation $S \Rightarrow D \Rightarrow C\mathtt{a} \Rightarrow B\mathtt{ba} \Rightarrow A\mathtt{aba} \Rightarrow \mathtt{aba}$.

Any other solution that attempts to convert the regular expression of the DFA to a CFG would get partial scores, as it results in unnecessary complication.

Problem 4 (37 points) Consider the context-free grammar G:

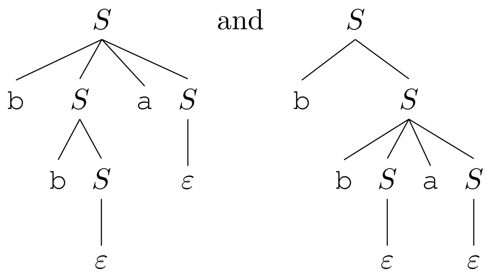
$$S \to \mathtt{b}S\mathtt{a}S \mid \mathtt{b}S \mid \varepsilon$$

(a) (7 points) Show that G is ambiguous. *Hint: Some string of length three over* $\{\mathtt{a}, \mathtt{b}\}$ *will help you.*

(b) (8 points) Convert G to Chomsky Normal Form.

Solution:

(a) The string bba has two different parse trees:



Note that showing ambiguity requires demonstrating two different *parse trees* (or different *leftmost derivations*). It is not sufficient to just demonstrate two different *derivations* (which may have the same parse tree).

(b) We notice that the start variable S appears on the right of some i rules. First, we add a new start variable S'.

$$S' \to S$$
$$S \to \mathtt{b}S\mathtt{a}S \mid \mathtt{b}S \mid \varepsilon$$

Next, we remove ε-production. We need to add a new rule whenever S appears on the right. If S appears multiple times in same rule, we need to add rules for all possible combinations of removal.

$$S' \to S \mid \varepsilon$$
$$S \to \mathtt{b}S\mathtt{a}S \mid \mathtt{ba}S \mid \mathtt{b}S\mathtt{a} \mid \mathtt{ba}$$
$$S \to \mathtt{b}S \mid \mathtt{b}$$

Then, we can remove unit production and replace all terminal appeared on right by a new nonterminal.

$$S' \to BSAS \mid BAS \mid BSA \mid BA \mid BS \mid \mathtt{b} \mid \varepsilon$$
$$S \to BSAS \mid BAS \mid BSA \mid BA \mid BS \mid \mathtt{b}$$
$$A \to \mathtt{a}$$
$$B \to \mathtt{b}$$

Finally, we break down the rules by introducting new variables. You may split them differently as long as it is CNF. Here is one possible solution.

$$\begin{aligned}
S' &\to CD \mid BD \mid CA \mid BA \mid BS \mid \mathtt{b} \mid \varepsilon \\
S &\to CD \mid BD \mid CA \mid BA \mid BS \mid \mathtt{b} \\
C &\to BS \\
D &\to AS \\
A &\to \mathtt{a} \\
B &\to \mathtt{b}
\end{aligned}$$

Common mistake: In the last step, many students replace $\quad S \to BSAS \mid BS \quad$ by

$$S \to CD \mid C, \qquad C \to BS \qquad \text{and} \qquad D \to AS.$$

This introduces a new unit production, hence the grammar is not in CNF yet.

Grading (8 points): 1 for new start variable S', 4 for ε-production removal, 1 for terminal replacement, 2 for grammar in CNF

Problem 4 (continued)

(c) (10 points) Using the grammar from part (b), apply CYK algorithm on input `bbaa`. Show the table of partial derivations. Draw a parse tree derived by the algorithm.

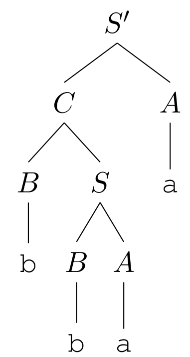
(d) (12 points) Consider the language L of the context-free grammar G from the previous page. Show that L is irregular. *Hint: Guess what L is.*

Solution:

(c) On input `bbaa`, the run of the CYK algorithm looks like this:

$S \mid S'$			
$C \mid S \mid S'$	–		
$C \mid S \mid S'$	$S \mid S'$	–	
$B \mid S \mid S'$	$B \mid S \mid S'$	A	A
`b`	`b`	`a`	`a`

The table yields the following tree.



Common mistake: Given a input $x_1x_2...x_n$, in CYK table, the (i, j)-entry should contain ALL nonterminals that can generate the substring $x_ix_{i+1}...x_j$. There can be multiple nonterminals in single cell, even though some of them are not useful in the final parse tree.

Remark: If your grammer in part (b) is not correct, you can still get at most 9 marks for this question, as long as there is a corrcet CYK table based on your grammer and a correct parse tree based on the CYK table.

Grading (10 points): 7 for the CYK table, 3 for a parse tree

(d) $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid$ for all prefix x of w, the number of `b`'s in x is at least that of `a`'s$\}$.

The language L can be shown to be irregular using pairwise distinguishable strings. We claim that the set $\{\mathtt{b}^n \mid n \geqslant 0\} = \{\varepsilon, \mathtt{b}, \mathtt{bb}, \mathtt{bbb}, \dots\}$ is pairwise distinguishable by L. Indeed, for any two strings $x \neq y$ in this set, we write $x = \mathtt{b}^i$ and $y = \mathtt{b}^j$, and assume $i > j$ without loss of generality. Take $z = \mathtt{a}^i$. Then $xz \in L$ since number of `a`'s is 0 for all prefixes of x, and it is at most i for any prefix of xz (but not of x) which contains i `b`'s. And $yz \notin L$, since yz, a prefix of yz itself, has more `a`'s than `b`'s.

Alternatively, this can also be proven using the pumping lemma. Assume L is regular and let n be the pumping length. Let $s = \mathtt{b}^n\mathtt{a}^n \in L$. No matter how we break up z into uvw where $|uv| \leqslant n$ and $|v| > 0$, v is in the part $\mathtt{b}^n$. Write $v = \mathtt{b}^k$ for some $k > 0$. If we take $i = 0$, the string $uv^iw = uv^0w = \mathtt{b}^{n-k}\mathtt{a}^n$ contains strictly less b's than a's and thus is not in L. So L is irregular by the pumping lemma.

Many students claimed L to be the set of strings with number of a's at least that of b's and does not start with a, which is wrong. In such cases, in the proof where you want to show a string is in or not in L, some points would be deducted if you say that it follows directly from definition. On the other hand, full marks could be given even if you did not state what L is, but was able to show that the strings are in or not in L by deriving from the grammar G.