Discussion

It always pleases me when students come up with elegant solutions and insights that had not occurred to me when I created the exam. I experienced this pleasure several times while grading your work. Here are the most notable examples.

- Ohene Kwasi Ohene-Adu solved #4 by considering the NFAs for $(L_1 L_2)^*$ and $(L_1 \cup L_2)^*$ built using standard constructions from the NFAs for $L_1$ and $L_2$. Although it’s quite hard to argue rigorously, this approach does make it easy to visualize what’s happening.

- David Blinn came up with a very cute proof for #6.4 that does not involve complements (for comparison, see my official solution below). He used the language $L = 0(01)^*00(01)^*$ and observed that $\text{SELECT}(L) = \{x \in \{0,1\}^* : N_0(x) - N_1(x) \geq 2\}$, after which it was a simple job for the pumping lemma.

- Eric Robinson came up with an ingenious “square” PDA drawn in a manner that makes it very clear that it works correctly. I can’t do justice to his solution with a few words, so I suggest you get him to show you his construction!

There were not very many negatives to note. Sometimes your work fell a little short of the level of rigor expected in this course. It appears that several of you were stumped by #6.3 and/or #6.4 in the sense that you tried to prove that the statements are true. If you did so you were, of course, bound to fail, for these statements are false. These two problems were the main sources of lost points.

Overall I would rate the class’s work on the midterm as “very good.” With a little more care your work on the final would be “excellent.”

Please note that Chien-Chung Huang graded #1, #2, #7 and #8 and that I graded the rest.

The Solutions

1. Write a regular expression for the language generated by the following grammar:

   $S \rightarrow AT$
   $T \rightarrow ABT | TBA | AA$
   $A \rightarrow 0$
   $B \rightarrow 1$

   A single line answer will do; you don’t have to justify or show any steps. Your regular expression should be as simple as possible.

   Solution: The grammar generates $0(01)^*00(10)^*$.

2. Draw a DFA for the language

   $\{x \in \{0,1\}^* : x \text{ contains an equal number of occurrences of the substrings } 01 \text{ and } 10 \}$.

   For example, 101 and 0000 are in the language, but 1010 is not.

   Solution: The idea is to handle strings beginning with a 1 and strings beginning with a 0 separately. The following DFA does the job:
3. Recall that $x^R$ denotes the reverse of the string $x$. For a language $L$, let $L^R = \{x^R : x \in L\}$. Prove that if $L$ is regular, so is $L^R$.

**Solution:** The idea is to start with a DFA for the regular language $L$ and “reverse all the arrows” in this DFA, make its start state an accept state of the resulting machine, and make all its accept states start states (or rather, since only one start state is allowed, to introduce $\varepsilon$-transitions from a new start state to all the former accept states). This procedure clearly gives us an NFA, because, for instance, if there are five different states of the initial DFA that all lead into a state $q$ on reading input symbol $a$, then after the conversion, $q$ will lead to these five different states on input $a$. Now we give a formal proof.

Suppose $L$ is a regular language. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing $L$. Let us construct an NFA $M' = (Q \cup \{q_{\text{new}}\}, \Sigma, \delta', q_{\text{new}}, \{q_0\})$ where $\delta'$ is given by

$$
\begin{align*}
\delta'(q_{\text{new}}, \varepsilon) &= F, \\
\delta'(q_{\text{new}}, a) &= \emptyset, \quad \forall a \in \Sigma, \\
\delta(q, \varepsilon) &= \emptyset, \quad \forall q \in Q, \\
\delta(q, a) &= \{r \in Q : \delta(r, a) = q\}, \quad \forall q \in Q, a \in \Sigma.
\end{align*}
$$

We claim that $L(M') = L^R$, which would prove that $L^R$ is regular. To prove our claim, we need to argue that (1) $L(M') \subseteq L^R$ and that (2) $L^R \subseteq L(M')$.

To prove (1), consider an $x \in L(M')$. By definition, this means that we can write $x = a_1a_2\ldots a_n$ with each $a_i \in \Sigma_\varepsilon$ and find a sequence $r_0, r_1, \ldots, r_n$ of states of $M'$ such that

- $r_0 = q_{\text{new}}$, the start state of $M'$,
- $r_i \in \delta'(r_{i-1}, a_i)$, for $1 \leq i \leq n$, and
- $r_n \in \{q_0\}$, the set of final states of $M'$.

By construction, $a_1$ must be $\varepsilon$, because otherwise the set $\delta'(r_0, a_1) = \delta'(q_{\text{new}}, a_1)$ would be empty. Also, the states $r_{i-1}$ for $2 \leq i \leq n$ must all be different from $q_{\text{new}}$, and so, every $a_i$ for $2 \leq i \leq n$ must be different from $\varepsilon$ (i.e., in the set $\Sigma$). Therefore, by construction, $\delta'(r_{i-1}, a_i)$ for $2 \leq i \leq n$ is the set of all $r$ such that $\delta(r, a) = r_{i-1}$. Since, by the second bullet above, $r_1$ is in this set, it follows that $\delta(r_1, a_1) = r_{i-1}$. Finally, the state $r_1$ must lie in $F$, because it must lie in $\delta'(r_0, a_1) = \delta'(q_{\text{new}}, \varepsilon) = F$. Thus, we have

- $r_n = q_0$, the start state of $M$,
- $r_{i-1} = \delta(r_i, a_i)$, for $n \geq i \geq 2$, and
Prove that there exist languages

4. Let \( L_1, L_2 \) be two languages over the same alphabet \( \Sigma \). Prove that \((L_1^* L_2^*)^* = (L_1 \cup L_2)^*\). Remember that to prove \( A = B \) for sets \( A \) and \( B \) you must separately prove the two statements \( A \subseteq B \) and \( B \subseteq A \).

Solution: Suppose \( x \in (L_1^* L_2^*)^* \). By definition of Kleene star, \( x \) is a concatenation of zero or more strings, each in \( L_1^* L_2^* \), i.e. \( x = x_1 x_2 \ldots x_n \) with each \( x_i \in L_1^* L_2^* \). Again, by definition, each \( x_i = y_{i1} y_{i2} \ldots y_{i_s} z_{i1} z_{i2} \ldots z_{it_i} \) with each \( y_{ij} \in L_1 \) and each \( z_{ij} \in L_2 \). Putting it all together:
\[
x = y_{11} \ldots y_{1s_1} z_{11} \ldots z_{1t_1} \ldots y_{2s_2} z_{21} \ldots z_{2t_2} \ldots \ldots z_{nt_n}.
\]
Since each \( y_{ij} \) and each \( z_{ij} \) is in \( L_1 \cup L_2 \), it follows that \( x \in (L_1 \cup L_2)^* \). We have shown that \((L_1^* L_2^*)^* \subseteq (L_1 \cup L_2)^*\).

Now, suppose \( x \in (L_1 \cup L_2)^* \). Then \( x = x_1 x_2 \ldots x_n \) where each \( x_i \in L_1 \cup L_2 \). If an \( x_i \in L_1 \), we can write \( x_i = x_i \varepsilon \) which puts it in \( L_1^* L_2^* \). If an \( x_i \in L_2 \), we can similarly write \( x_i = \varepsilon x_i \) which puts it in \( L_1^* L_2^* \). Therefore, each \( x_i \in L_1^* L_2^* \), whence \( x \in (L_1^* L_2^*)^* \). We have shown that \((L_1 \cup L_2)^* \subseteq (L_1^* L_2^*)^*\).

This completes the proof.

5. Prove that there exist languages \( A, B, C \subseteq \{0, 1\}^* \) that satisfy all of the following properties:

(a) \( A = B \cap C \).
(b) \( B \) and \( C \) are both non-regular.
(c) \( A \) is infinite and regular.

To get any credit, you must prove all three properties for whatever \( A, B, C \) you have decided to use.

Solution: Define the sets \( X = \{0^n^2 : n \geq 0\} \), \( Y = 0^* - X \) and \( A = 1^* \). Note that, by construction, any two of \( X, Y \) and \( A \) are disjoint; so a Venn diagram of these three sets would look like this:

![Venn diagram]

Define \( B = X \cup A \), \( C = Y \cup A \). By the disjointness observed above (see diagram), condition (a) clearly holds. Since \( A = 1^* \) is regular, condition (c) also holds. The only nontrivial thing is to establish condition (b). Let us use bars to denote complements of languages with
draw a PDA for the language

6. A permutation of a string \(x\) is any string that can be obtained by rearranging the characters of \(x\). Thus, for example, the string \(abc\) has exactly six permutations:

\[
abc, aeb, bca, cba, bac, cab.
\]

Clearly, if \(y\) is a permutation of \(x\), then \(|y| = |x|\). For a language \(L\) over alphabet \(\Sigma\), define

\[
\text{PERMUTE}(L) = \{x \in \Sigma^* : x \text{ is a permutation of some string in } L\},
\]

\[
\text{SELECT}(L) = \{x \in \Sigma^* : \text{every permutation of } x \text{ is in } L\}.
\]

Classify each of the following statements as \text{true} or \text{false}, and give proofs justifying your classifications.

6.1. If \(L_1 = 1^*0\), then \(\text{PERMUTE}(L_1)\) is regular.

\textbf{Solution: True.} In this case \(\text{PERMUTE}(L_1) = \{x \in \{0,1\}^* : x \text{ contains exactly one } 0\} = 1^*01^*\), which is clearly regular.

6.2. If \(L_1 = 1^*0\), then \(\text{SELECT}(L_1)\) is regular.

\textbf{Solution: True.} Any string in \(\text{SELECT}(L_1)\) must clearly be in \(L_1\) (since a string is always a permutation of itself); however, a string in \(L_1\) must end in a 0, so permuting it will destroy this property unless the string is just a plain 0. Thus, \(\text{SELECT}(L_1) = \{0\}\), which is clearly regular.

6.3. Regular languages are closed under the operation \(\text{PERMUTE}\).

\textbf{Solution: False.} For a counterexample, consider \(L = (01)^*\). Any string in \(L\) contains an equal number of 0’s and 1’s, so, when we take all permutations of all strings in \(L\), we get precisely the language \(\{x \in \{0,1\}^* : x \text{ contains as many } 0\text{’s as } 1\text{’s}\}\). We have proved in class, using the pumping lemma, that this language is not regular. Thus \(\text{PERMUTE}(L)\) need not be regular even if \(L\) is.

6.4. Regular languages are closed under the operation \(\text{SELECT}\).

\textbf{Solution: False.} For a counterexample, consider \(L’ = (01)^*\), with the complement being taken with respect to alphabet \(\{0,1\}\). Clearly \(L’\) is regular. If a string \(x\) has an unequal number of 0’s and 1’s, then no matter how we permute it we will never land in the set \((01)^*\), i.e., every permutation of \(x\) is in \(L’\), i.e., \(x \in \text{SELECT}(L’)\). On the other hand, if a string \(x\) has as many 0’s as 1’s, then it can be permuted into the form \((01)^*\), so \(x \notin \text{SELECT}(L’)\). In short, we have shown that \(\text{SELECT}(L’) = \{x \in \{0,1\}^* : x \text{ contains an unequal number of } 0\text{’s and } 1\text{’s}\}\). Thus, \(\text{SELECT}(L’)\) is the complement of a non-regular language, whence it is itself non-regular.

7. Draw a PDA for the language \(\{0^i1^j : i < j < 2i\}\). For clarity, keep your stack alphabet disjoint from \(\{0,1\}\). Provide a brief justification (no need for a formal proof) that your PDA works correctly.
Solution: Let $L$ be the given language. First, let us try to build a PDA for a slightly easier language: $L' = \{0^i1^j : i \leq j \leq 2i\}$. The idea is to push either one or two $a$'s (choose nondeterministically) onto our stack as we read the $0$'s in the input string, and then pop one symbol at a time as we read the $1$'s. If we empty the stack at the same time as when we finish reading the input, we may accept.

But for the language $L$, we must reject strings of the form $0^i1^i$ and $0^i1^{2i}$. To do so, observe that the shortest string in $L$ is $00111$ and that any string in $L$ is therefore of the form $000^i1^j000$ for some $i \geq 0, j \geq 0$, and $(i' + 2) < (j' + 3) < 2(i' + 2)$. Since $i'$ and $j'$ are integers, the latter condition is equivalent to $i' \leq j' \leq 2i'$. Thus, strings in $L$ are simply strings in $L'$ with two $0$'s prepended and three $1$'s appended. With this in mind, we build our PDA:

8. Design a context-free grammar for the complement of the language $\{a^n b^n : n \geq 0\}$ over the alphabet $\{a, b\}$. Give brief explanations for the “meanings” of your variables (i.e. explain what strings are generated by each of your variable).

Solution: A string in the complement of $\{a^n b^n : n \geq 0\}$ is either of the form $\{a^i b^j : i > j \geq 0\}$ or of the form $\{a^i b^j : j > i \geq 0\}$ or not of the form $a^*b^*$, which means that it contains the substring $ba$. In the following grammar, the variable $U$ generates strings of the third type, while $T$ generates strings of the form $a^n b^n$ to which we either prepend a string of one or more $a$'s (generated by $A$) or append a string of one or more $b$'s (generated by $B$).

$$
S \rightarrow AT \mid TB \mid U \\
A \rightarrow Aa \mid a \\
B \rightarrow Bb \mid b \\
T \rightarrow aTb \mid \varepsilon \\
U \rightarrow VbaV \\
V \rightarrow Va \mid Vb \mid \varepsilon
$$