Solutions to Mid-Term Exam
Written by Amit Chakrabarti

Prof. Amit Chakrabarti
Computer Science Department
Dartmouth College

Who Graded What

Hong Lu: #1, #2, #6

Amit Chakrabarti: #3, #4, #5

Vibhor Bhatt: #7, #8

The Solutions

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

$$\begin{aligned} S &\longrightarrow AT \\ T &\longrightarrow ABT \mid TBA \mid AA \\ A &\longrightarrow 0 \\ B &\longrightarrow 1 \end{aligned}$$

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.

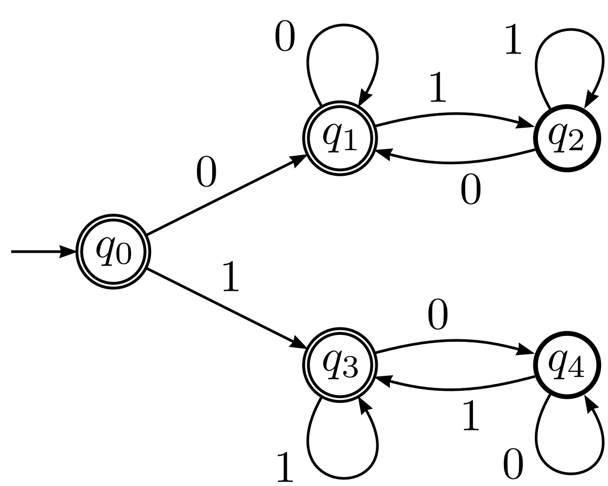
Solution: The grammar generates $0(01)^*00(10)^*$.

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 $x^{\mathcal{R}}$ denotes the reverse of the string x. For a language L, let $L^{\mathcal{R}} = \{x^{\mathcal{R}} : x \in L\}$. Give a complete formal proof that if L is regular, so is $L^{\mathcal{R}}$.

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

ε-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 δ' is given by

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

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

To prove (1), consider an $x \in \mathcal{L}(M')$. By definition, this means that we can write $x = a_1 a_2 \ldots a_n$ 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 $1 \le i \le n$, and
- $r_n \in \{q_0\}$, the set of final states of M'.

By construction, a_1 must be ε, 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 $2 \le i \le n$ must all be different from q_{new}, and so, every a_i for $2 \le i \le n$ must be different from ε (i.e., in the set Σ). Therefore, by construction, $\delta'(r_{i-1}, a_i)$ for $2 \le i \le n$ is the set of all r such that $\delta(r, a) = r_{i-1}$. Since, by the second bullet above, r_i is in this set, it follows that $\delta(r_i, a_i) = r_{i-1}$. 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 $n \ge i \ge 2$, and
- $r_1 \in F$, the set of final states of M.

Thus we have a sequence $r_n, r_{n-1}, \ldots, r_1$ of states of M which satisfies the above three bullets; this ensures that $a_n a_{n-1} \ldots a_2 \in \mathcal{L}(M) = L$, i.e., that $x^{\mathcal{R}} \in L$, i.e., that $x \in L^{\mathcal{R}}$.

The proof of (2) is very similar, so we only sketch it here. We start with an $x \in L^{\mathcal{R}}$. This means $x^{\mathcal{R}} \in L = \mathcal{L}(M)$. Therefore $x^{\mathcal{R}}$ takes M through a sequence $r_0, r_1, \ldots, r_n$ of steps with $r_0 = q_0$ and $r_n \in F$. Arguing just as above, we can show that x can take M' through the sequence of states $q_{\text{new}}, r_n, r_{n-1}, \ldots, r_0$ and since q_{new} is the start state of M' and r_0 is an accept state of M', we see that M' accepts x. So $x \in \mathcal{L}(M')$.

4. Consider two languages $A, B \subseteq \Sigma^*$. Prove that $(A^*B^*)^* = (A \cup B)^*$. Remember that to prove $X = Y$ for sets X and Y you must separately prove $X \subseteq Y$ and $Y \subseteq X$.

First Solution: Suppose $x \in (A^*B^*)^*$. By definition of Kleene star, x is a concatenation of zero or more strings, each in A^*B^*, i.e. $x = x_1 x_2 \ldots x_n$ with each $x_i \in A^*B^*$. Again, by definition, each $x_i = y_{i1} y_{i2} \ldots y_{is_i} z_{i1} z_{i2} \ldots z_{it_i}$ with each $y_{ij} \in A$ and each $z_{ij} \in B$. Putting it all together:

$$x = y_{11} \ldots y_{1s_1} z_{11} \ldots z_{1t_1} y_{21} \ldots y_{2s_2} z_{21} \ldots z_{2t_2} \ldots\ldots\ldots z_{nt_n}\,.$$

Since each y_{ij} and each z_{ij} is in $A \cup B$, it follows that $x \in (A \cup B)^*$. We have shown that $(A^*B^*)^* \subseteq (A \cup B)^*$.

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

This completes the proof.

Second Solution: Since $A^* \subseteq (A \cup B)^*$ and $B^* \subseteq (A \cup B)^*$, we have

$$A^*B^* \subseteq (A \cup B)^*(A \cup B)^* = (A \cup B)^* .$$

Applying the Kleene star to both sides, we get $(A^*B^*)^* \subseteq ((A \cup B)^*)^* = (A \cup B)^*$.

Since $A \subseteq A^* \subseteq A^*B^*$ and $B \subseteq B^* \subseteq A^*B^*$, we have $A \cup B \subseteq A^*B^*$. Applying the Kleene star to both sides, we get $(A \cup B)^* \subseteq (A^*B^*)^*$.

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

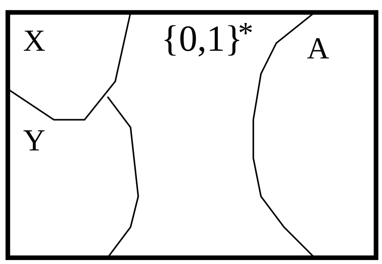
(a) $A = B \cap C$.

(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 $A = 11^*$. 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:



Define $B = X \cup A$, $C = Y \cup A$. By the disjointness observed above (see diagram), condition (a) clearly holds. Since $A = 11^*$ 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 respect to $\{0,1\}^*$. Then, by the disjointness conditions (see diagram), $X = B - A = B \cap \overline{A}$. Similarly, $Y = C \cap \overline{A}$. Now we shall establish condition (b) using a proof by contradiction.

First, suppose B is regular. Then, by closure of regular languages under complement and intersection, $X = B \cap \overline{A}$ would also be regular. But we've prove in class that X is not regular, so we have a contradiction. Thus, B must be non-regular. Next, suppose C is regular. Again, by closure properties, $Y = C \cap \overline{A}$ would be regular. The complement of Y *with respect to* 0^* is X, so X must also be regular and again we have a contradiction. Thus, C must be non-regular.

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:

$$abc,\ acb,\ bac,\ bca,\ cab,\ cba\,.$$

Clearly, if y is a permutation of x, then $|y| = |x|$. For a language L over alphabet Σ, define

$$\begin{aligned} \textsc{Permute}(L) &= \{x \in \Sigma^* : x \text{ is a permutation of some string in } L\}, \\ \textsc{Select}(L) &= \{x \in \Sigma^* : \text{every permutation of } x \text{ is in } L\}. \end{aligned}$$

Classify each of the following statements as TRUE or FALSE, and give proofs justifying your classifications.

6.1. If $L_1 = 1^*0$, then $\textsc{Permute}(L_1)$ is regular.
Solution: TRUE. In this case $\textsc{Permute}(L_1) = \{x \in \{0,1\}^* : x \text{ contains exactly one } 0\} = 1^*01^*$, which is clearly regular.

6.2. If $L_2 = 0^*1^*$, then $\textsc{Select}(L_2)$ is regular.
Solution: TRUE. We claim that $\textsc{Select}(L_2) = 0^* \cup 1^*$, which is clearly regular. To prove this, we first note that any string in $0^* \cup 1^*$ has only one permutation (namely, itself) and that belongs to L_2; therefore $0^* \cup 1^* \subseteq \textsc{Select}(L_2)$. We then note that any string $x \in L_2 - (0^* \cup 1^*)$ contains at least one 0 and at least one 1, so one of its permutations must contain the substring "10." This permutation clearly does not belong to L_2, so $x \notin \textsc{Select}(L_2)$. Therefore $\textsc{Select}(L_2) \subseteq 0^* \cup 1^*$.

6.3. Regular languages are closed under the operation PERMUTE.
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 $\textsc{Permute}(L)$ need not be regular even if L is.

6.4. Regular languages are closed under the operation SELECT.
Solution: FALSE. For a counterexample, consider $L' = \overline{(01)^*}$, 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., $x \in \textsc{Select}(L')$. 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 $x \notin \textsc{Select}(L')$. In short, we have shown that $\textsc{Select}(L') = \{x \in \{0,1\}^* : x \text{ contains an unequal number of } 0\text{'s and } 1\text{'s}\}$. Thus, $\textsc{Select}(L')$ is the complement of a non-regular language, whence it is itself non-regular.

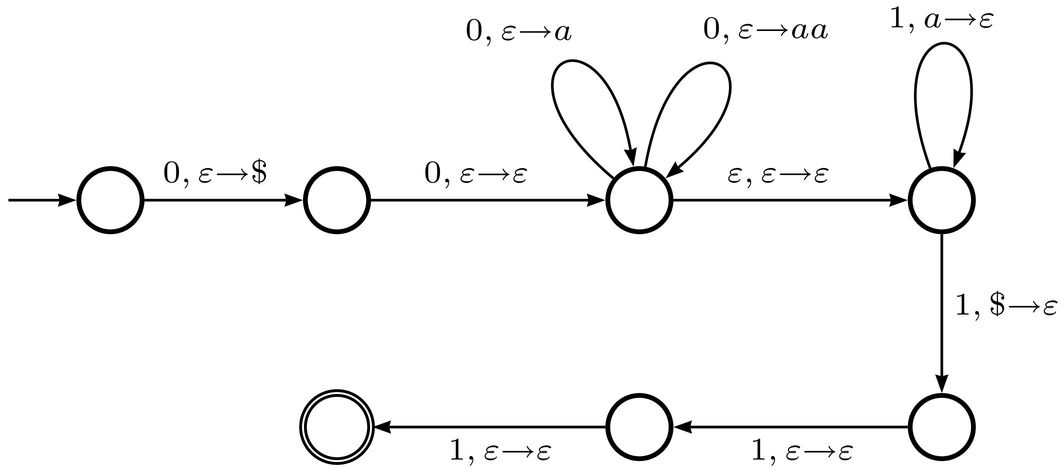
7. Draw a PDA for the language $\{0^i1^j : i < j < 2i\}$. For clarity, keep your stack alphabet disjoint from $\{0,1\}$. Provide a brief justification (no need for a formal proof) that your PDA works correctly.

Solution: Let L be the given language. First, let us try to build a PDA for a slightly easier language: $L' = \{0^i1^j : i \le j \le 2i\}$. 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

$$000^{i'}1^{j'}111$$

for some $i' \ge 0, j' \ge 0$, and $(i'+2) < (j'+3) < 2(i'+2)$. Since i' and j' are integers, the latter condition is equivalent to $i' \le j' \le 2i'$. 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).

$$\begin{array}{rcl} S & \longrightarrow & AT \mid TB \mid U \\ A & \longrightarrow & Aa \mid a \\ B & \longrightarrow & Bb \mid b \\ T & \longrightarrow & aTb \mid \varepsilon \\ U & \longrightarrow & VbaV \\ V & \longrightarrow & Va \mid Vb \mid \varepsilon \end{array}$$

Another Cute Solution: The following cute CFG, suggested by Jason Reeves '07, also happens to work (proof left as an exercise to the reader):

$$\begin{array}{rcl} A & \longrightarrow & B \mid aAb \\ B & \longrightarrow & a \mid b \mid aCa \mid bCb \mid bCa \\ C & \longrightarrow & \varepsilon \mid B \mid aCb \end{array}$$