CS181 Winter 2016 - Problem Set #3 Solutions

1. **(20 points)** Consider a new binary operation ∇ defined as follows: if A and B are two languages, then $A\nabla B = \{xy | x \in A, y \in B, \text{ and } |x| = |y|\}$. Give a proof sketch for the following statement: If A and B are regular languages, then $A\nabla B$ is a context-free language.

   Solution:
   Intuition.
   This operation almost represents concatenation except that the two strings taken from the languages must be equal in size. Recall that we knew how to do concatenation using NFAs. So, let's use a construction similar to the one given for concatenation while using the stack to our advantage to ensure that both strings have the same size. when we read the first string x, for every symbol that we read, we push a special symbol $*$ onto the stack. Then, when we read the second string y, for every symbol that we read, we pop the special symbol $*$ from the stack. Finally, we can check that the two strings have equal length if the stack is empty.
   Construction
   Since A and B are regular languages, we know that there must exist DFAs representing these languages. Let $M_A = (Q_A, \Sigma, \delta_A, F_A, q_A)$ be the DFA that recognizes language A and $M_B = (Q_B, \Sigma, \delta_B, F_B, q_B)$ be the DFA that recognizes language B. Our new PDA P that will recognize $A\nabla B$ will have state space $Q_A \cup Q_B \cup \{q_0\} \cup \{q_f\}$. Moreover, our stack alphabet will be $\Gamma = \{\$, *\}$. The $*$ symbol will be used to make sure that the strings have the same size. P will have a unique accept state, so that $F = \{q_f\}$. Its initial state will be q_0. We will have an ϵ transition which pushes the dollar symbol on the stack from q_0 to q_A: $\delta(q_0, \epsilon, \epsilon) = \{(q_A, \$)\}$. For states $q \in Q_A$, we push the $*$ symbol on the stack with each symbol that is consumed: $\delta(q, c, \epsilon) = \{(\delta_A(q, c), *)\}$. Similarly, for states $q \in Q_B$, we pop the $*$ symbol from the stack with each symbol that is consumed: $\delta(q, c, *) = \{(\delta_B(q, c), \epsilon)\}$. We have ϵ transitions from all old accepting states F_A to q_B: $\delta(q, \epsilon, \epsilon) = \{(q_B, \epsilon)\}$, where $q \in F_A$. Finally, for each of the old accepting states F_B, we pop the dollar symbol off the stack and transition to the unique accepting state in P: $\delta(q, \epsilon, \$) = \{(q_f, \epsilon)\}$, where $q \in F_B$. It is clear from our construction that this PDA recognizes $A\nabla B$. Thus, this proves that $A\nabla B$ is a context-free language.

2. **Grammars (30 points).** A *regular-grammar* $G = (V, \Sigma, R, S)$ is a grammar, such that for **every** variable $A \in V$, any rule in R is in one of the following forms:

   1) $A \to aB$
   2) $A \to a$
   3) $A \to \varepsilon$

   where a is some terminal alphabet and B is some variable, i.e., $a \in \Sigma$ and $B \in V$.
   (note that B might be the same variable as A).

(a) Show that any regular language L has a *regular*-grammar G_L that generates it.

(b) Suppose that G is some arbitrary regular grammar. Give a proof sketch that G produces a regular language.

(Hint: Show how to convert an NFA N into a regular-grammer G (and vice-versa), by mapping variables from G into states of N.)

Solution.

(a) (a)
Intuition.
Let's map the state space of the DFA representing L to the variable set of the grammar and convert every transition to a rule. The starting variable of the grammar would be the one corresponding to the start state of the DFA. Let's look at any transition. It goes from one state A to another state B on input symbol c. We can rewrite that as a rule in the following manner : variable A generates cB. Finally, for every final state, we add the rule that this variable goes to ϵ.
Construction.
Let $M = (Q, \Sigma, q_0, F, \delta)$ be some DFA. We show how to convert M into a regular grammar. Define $G = (V, \Sigma, R, S)$ such that the set of variables V has a variable for each one of the states of M (that is $V = Q$). Let the starting symbol be the variable related to the initial state, $S = q_0$. For every outgoing transition $\delta(q, \alpha) = p$ in M, where $p, q \in Q$ and $\alpha \in \Sigma$, add to R the rule $q \to \alpha p$ (note the change of convention, as we mark variables as small letters, and terminals as greek letters). If $q \in Q$ is a final state, add the rule $q \to \epsilon$.
We now prove that $L(G) = L(M)$.
Direction 1: $L(M) \subseteq L(G)$. Assume $x \in L(M)$ then M accepts $x = x_1 x_2 \ldots x_n$. Assume that computation is $q_0 = r_0 \to r_1 \to r_2 \to \cdots \to r_n$, where $\delta(r_{i-1}, x_i) = r_i$ for $i > 0$, and r_n is a final state, $r_n \in F$.
Therefore the following derivation in G generates x:

$$q_0 \Rightarrow x_1 r_i \Rightarrow x_1 x_2 r_2 \Rightarrow \cdots \Rightarrow x_1 x_2 \ldots x_n r_n \Rightarrow x\epsilon$$

It is easy to see that derivation i is according the rule $r_{i-1} \to x_i r_i$, which was added to G since $\delta(r_{i-1}, x_i) = r_i$, according to the definition. the last derivation is due to the rule $r_n \to \epsilon$, since $r_n \in F$
Direction 2: $L(G) \subseteq L(M)$. Assume x is generated by G, that is $q_0 \Rightarrow^* x$. The only rules in G are of the type $A \to aB$ or $A \to \epsilon$, so it must be that the the last derivation in $q_0 \Rightarrow^* x$ is of the second type and the rest are of the first type. Moreover, it must be that we have exactly $n = |x|$ derivations of the first type. Since every derivation rule is associated with transition in M, we can simulate the computation of x in M, and see that it ends in a state which is final, since the last derivation is of the second type.

(b) (b)

The same trick can be applied in reverse to convert a regular grammar into a DFA! But before that, let's first convert our grammar to eliminate the second rule completely! For every rule of the form $A \to a$, we rewrite it into two rules by adding a new variable A' in the following manner: $A \to aA'$ and $A' \to \epsilon$. Now, let's look at this new grammar which represents the same language as our previous grammar.

Let's map the set of variables to the state space of the DFA and convert the rules to transitions. The start state of the DFA would be the state corresponding to the starting variable of the grammar. Let's look at any rule. Suppose it is $A \to aB$. We add the transition that goes from the state corresponding to variable A to the state corresponding to variable B on input symbol a. Suppose the rule is of the form $A \to \epsilon$. We make the state corresponding to A as a final state of the DFA and not add any transition corresponding to this rule. The formal construction and proof are similar to the previous case with the changes described above.

3. **(30 points).** Let

$$L = \{xy \mid \ x, y \in \{0,1\}^* \ \ |x| = |y| \text{ and x} \neq \text{y}\}.$$

L consists strings where the first part of the string is different from the second part, with no special symbol in the middle and an additional constraint that two parts be of equal length. Show that there exists a PDA that accepts L. Explain your solution.

Solution:

Intuition.

The idea here is that, in order for the first half of string x and the second half of the string y to be different, there must be a position i in the strings x and y where $x_i \neq y_i$. We can break the string x down into three separate pieces and check for inequality of the middle part which consists of only the non equal symbol. Notice that in this, the third part of the first string can "overflow" into the first part of the second string. That is, the two strings can be transformed to not have equal length by enforcing that the first and third parts of each string have equal length! Then, we can just design a CFG that generates a concatenation of two strings that have at least one symbol not being equal and each of these strings has two equal length parts surrounding this unequal symbol. Then convert this CFG to a PDA.

Construction

Knowing that $|x| = |y| =$ length l, we can break the string xy down into separate pieces as follows:

$$(0 \cup 1)^{i-1} x_i (0 \cup 1)^{l-i} \cdot (0 \cup 1)^{i-1} y_i (0 \cup 1)^{l-i}.$$

Let $n = i - 1$ and $m = l - i$, we can rewrite the string into:

$$(0 \cup 1)^n x_i (0 \cup 1)^m (0 \cup 1)^n y_i (0 \cup 1)^m = (0 \cup 1)^n x_i (0 \cup 1)^n \cdot (0 \cup 1)^m y_i (0 \cup 1)^m.$$

Notice how we can now view the string as simply a concatenation of two $(0 \cup 1)^n x_i (0 \cup 1)^n$ strings. We just have to enforce the rule $x_i \neq y_i$.
We can create a CFG to implement our idea:

$S \to AB \mid BA$
$A \to 0A1 \mid 1A0 \mid 0A0 \mid 1A1 \mid 0$
$B \to 0B1 \mid 1B0 \mid 0B0 \mid 1B1 \mid 1$

Note that the first rule covers both $x_i = 0, y_i = 1$ and $x_i = 1, y_i = 0$

You can now transform it into a PDA by invoking the transformation of CFG-to-PDA that was covered in the lecture.