CS181-Homework3

February 2018

1 Exercise

proof idea: To prove that if A and B are regular then $A \nabla B$ is a context free grammar. We want two machines that run one after another. To do this, we will need to have one machine that push the symbol into the stack, and the other machine pop the symbol from the stack for each $\epsilon - transition$. The basic idea here is we want to keep the string length in each of the machine are equal. We will create $\epsilon - transition$ from each of the accepting state in Machine A that correspond to each of the starting state in Machine B. We also need to add one accepting state, so that each of the accepting state in Machine B should transition to to a new accepting state on an input ϵ when the top of the stack is a \$ sign.

Let M_A recognize L1, such that

$$M_A = \{Q_A, \Sigma_A, \delta_A, q_{A0}, F_A\}$$

Let M_B recognize L1, such that

$$M_B = \{Q_B, \Sigma_B, \delta_B, q_{B0}, F_B\}$$

Let's construct a PDA that will have the properties as we discussed,

$$M = \{Q, \Sigma, \delta, q_0, F\}$$

And for both M_A and M_B, we have states $q_0, q_1, q_2 ... q_n$ and $q_0, q_1, q_2 ... q_m$ respectively. Thus, our M will have states $q_{0A}, q_{1A} ..... q_{nA}, q_{0B}, q_{1B} .. q_{mB}$ as we mentioned above, we will add $q_{start} and q_{end}$. The set of states in our M:

$$Q = Q_1 \cup Q_2 \cup \{q_{start}, q_{end}\}$$

The set of our alphabet

$$\Sigma = \Sigma_A \cup \Sigma_B \cup \{\epsilon\}$$

our stack alphabet will be

$$\Gamma = \Sigma_A$$

$$q_0 = q_{start}$$

$$F = q_{end}$$

Our transition function will decide when to push the symbol into the stack and pop from the stack.

δ' {q,i,s} = (q_{0A}, Push \$) $q = q_{start}$, $i = \epsilon, s = \epsilon$
δ' {q,i,s} =(δ_A (q,i), Push i) q $\in Q_A$,i = Σ_A, $s = \epsilon$
δ' {q,i,s} =(q_{0B},Do nothing), q $\in F_A$,i = Σ_A, $s = \epsilon$
δ' {q,i,s} = (δ_B (q,i), Pop i),q $\in Q_B$,i = Σ_B, $s \neq$ \$
δ' {q,i,s} = (q_{end},Do nothing), $q_{end} \in F_B$, $i = \epsilon$, $s =$ \$

In this transition function, if we see $s = \epsilon$, then we do not need to worry about what's on the top of the stack. For this operation, we will keep push to the stack and pop from the stack until we reach q_{end} and \$ is on the top of the stack then we accept the input otherwise we reject. This also means the the number of the push and the number of pop mush equal to each other as we discussed in our proof idea.
Thus, our machine will recognizes a string by M iff $w = xy$ where x accepts by M_A and y accepts M_B, in which our new M must go from the starting state to the accepting state in M_A. If it accepts in our M_A, then $\epsilon - transition$ to M_B, it would need to do the same thing in which go from the starting state to the accept state in our M_B. The use of the stack is to ensure that we have the equal length such that $|A| = |B|$, and if we have the $|A| \neq |B|$, then that means the input will result in the computation failing.

1.1 Exercise

a. To show that L_1 over alphabet δ = { 0,1,\$ } is a context-free grammar.We need to design a context free grammar that captures these properties. We say that our start symbol is S and terminal symbols are { 0,1,\$ } and we also need to make sure that our context free grammar satisfies the L_1 constraint such that $|x| \neq |y|$

$$S \rightarrow \text{ourgrammar}$$
$$ourgrammar \rightarrow \text{Num ourgrammar Num}$$
$$ourgrammar \rightarrow \# \text{ Num starNum}$$
$$ourgrammar \rightarrow \text{starNum Num } \#$$
$$starNum \rightarrow \text{Num startNum} \mid \epsilon$$
$$Num \rightarrow 0 \mid 1$$

In these rules, if we look from bottom up by substituting rules , we could see that these rules cover imbalance property of L_1 such that $|x| \neq |y|$. In these rule lists, the third line says that we must have at least one digit x on the right side. Similarly, the fourth line says that we must have at least one digit y on the left side. The starNum non-terminal symbol in this case is to encompass { 0,1,$ } and it also has one another property is that it has the ϵ in which we will do nothing on the substitution. Thus, this will ensure the inequality property of L_1.
Now we can take care of the case where they have the same number of characters, but they don't have the same indices. Lets take a look on the common string $NNNN.....0SNNN.....1S$ Notice that in this case, N can stands for any number of numbers; however, in this case they should have the same,but if they are not the same, then what means we accept the language based on our rule lists above.
We will add additional rules to our grammar rule list:

$$XZero \rightarrow \text{Num XZero Num}$$
$$XZero \rightarrow 0 \text{ starNum } \#$$

These two rules will allow us to have the first part of our generic string such $NNNN.....0SNNN.....$ and in order to allows 1 we will add another rule:

$$S \rightarrow \text{XZero 1 starNum}$$
$$XOne \rightarrow \text{Num XOne Num}$$
$$XOne \rightarrow 1 \text{ starNum } \#$$

Based on the rules we have, we constructed a context free grammar in which meet our imbalance constraint.
b. In the previous example, we have discussed two cases. In case 1, we say that we have the same length of of x and y. In the second case, we look at a generic string in which we have the same numbers of x and y but with different indices, because otherwise it would capture by the first case. So then we look at the $NNNN.....0SNNN.....1S$, we can omit the # in this string, then we will have a string $NNNN.....0SNNN.....1S$, this would be invalid because we assume that two Ss will give us the same length. For the previous problem, it is easily to get $NNNN.....0NNN....$; however, we can have two simple grammar here:

$$S \rightarrow \text{ZeroNum OneNum} \mid \text{OneNum ZeroNum}$$
$$ZeroNum \rightarrow \text{Num ZeroNum Num} \mid 0$$
$$OneNum \rightarrow \text{Num OneNum Num} \mid 1$$

Based on the rules we have, we constructed a context free grammar in which meet our imbalance constraint of L_2. We prove that L_2 is a context free grammar.