CS331: Homework #4

Due on February 28, 2013 at 11:59pm

Professor Zhang 9:00am

Josh Davis

Problem 1

Find a CFG to describe L.

The CFG, G, that describes L is below:

$$G = (V, \Sigma, R, S)$$

such that

$$G = (\{A, B\}, \{a, b, c\}, R, A)$$

where

$$\begin{aligned} &R: \\ &\quad A \to aAc|B|\epsilon \\ &\quad B \to bB|\epsilon \end{aligned}$$

The push down automata that represents G is as follows:

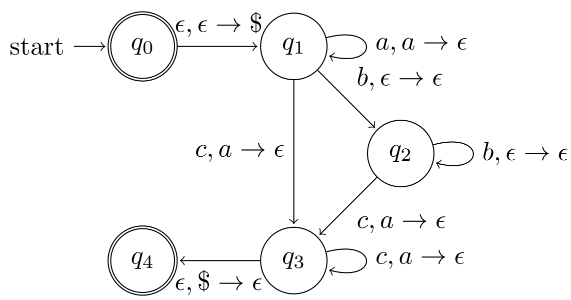


Figure 1: PDA, A

Problem 2

Part A
Prove that if L is context-free and L' is a regular languagte, then $L \cap L'$ is context-free too.

Proof. If L is a context-free language, and L' is a regular language, then let M and N be the finite automata and push down automata that accept both languages respectively.

We can construct a new push down automata, O such that M and N both receive the input and the new machine accepts a word only if both M and N accept it. This is possible because a finite automata doesn't use a stack which means one stack is sufficient for the complete push down automata. This concludes the proof. $\square$

Part B
Let $\Sigma = \{a, b, c\}$ and

$$L = \{w \in \Sigma^* : w \text{ contains equal number of a's, b's, and c's}\}$$

Prove that L is not a context-free language.

Proof. Suppose L is context-free. Let $L' = \{a^*b^*c^*\}$, a regular language.

Then according to what we proved in the first part, $L \cap L'$ is context-free. This is a contradiction because the language $\{a^n b^n c^n\}$ is not context-free thus violating our previous proof. We have arrived at a contradiction therefore our proof is complete.

$\square$

Problem 3

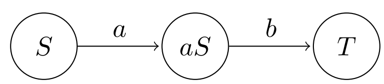
Prove that right linear grammars recognize exactly the class of regular languages.

Proof. This proof will consist of two parts. First proving that any regular language can be described by a RLG, and that any language described by an RLG is regular.

Part One Proving that any regular language can be described by an RLG.

Let $G = \langle V, \Sigma, R, S \rangle$, a right linear grammar. We can construct a NFA, $A = (Q, \Sigma, \delta, q, F)$ as follows:

1. $Q = \{v : \text{ for every } v \in V\}$
2. The alphabet is the same, $\Sigma = \Sigma$
3. The start state is the start variable, $q = S$
4. Since a grammar is complete once all variables are removed, we can add a new final state for this, $F = V_{final}$
5. For every rule, there is a transition from the variable state, S to the next variable state, T. Since it is a right linear grammar, for every rule there exists a rule such that $S \to wT$, where $w \in \Sigma^*$. Therefore we can represent the transition from one variable state to the next as a string of states such that there is a state for every terminal character in the rule. For example, the rule $S \to abT$ would be represented as follows:



This shows that our right linear grammar, G can be made into an automata, A.

Part Two Proving that any language described by an RLG is regular.

Let $A = (Q, \Sigma, \delta, q, F)$, a finite automata. We can construct a right linear grammar similar to how we constructed an NFA above. We will construct a RLG, $G = \langle V, \Sigma, R, S \rangle$ as follows:

1. $V = \{q : \text{ for every } q \in Q\}$
2. The alphabet is the same, $\Sigma = \Sigma$
3. The start variable becomes the start state, $S = q$
4. For every transition, $\delta(q, a) = q_1$ where $q_1 \in Q$, we can create a rule for it such that R would be defined as follows:

$$R :$$
$$q \to aq_1$$

This shows that any finite automata can be made into a right linear grammar.

Since both sides have been proven, the proof is complete and RLGs recognize exactly the class of regular languages.

$\square$

Problem 4

Use pumping lemma to show that the following language is not context-free:

$$L = \{a^i b^j c^k : i, j, k \geq 0 \text{ and } i > j \text{ and } j > k\}$$

Proof. We let $w = a^{p+2}b^{p+1}c^p$ where p is the pumping length. We then split up w so that $w = uvxyz$. In doing so, we can break it up into cases like so:

1. vxy contains just a's. This will result in a contradiction because pumping down yields $a^0a^{p-2}a^0 = \epsilon a^p \epsilon$ which means $i \leq j$ and thus $w \notin L$.
2. vxy contains just b's. This will result in a contradiction because pumping down yields $b^0b^{p-2}b^0 = \epsilon b^p \epsilon$ which means $j \leq k$ and thus $w \notin L$.
3. vxy contains just c's. This will result in a contradiction because pumping up will eventually give us $k > j$ thus $w \notin L$.
4. vxy is the split between a's and b's. This can be divided into four more cases:
   (a) $vxy = aab$, pumping down will lower the number of b's to be the same as the number of c's. Thus $w \notin L$.
   (b) $vxy = abb$, pumping down will lower the number of a's to be the same as the number of b's. Thus $w \notin L$.
   (c) $vxy = \epsilon ab$, pumping up will increase the number of b's past the number of a's. Thus $w \notin L$.
   (d) $vxy = ab\epsilon$, pumping down will lower the number of a's to be the same as the number of b's. Thus $w \notin L$.
5. vxy is the split between b's and c's. Regardless of where the subsplit is, it can be pumped until eventually $i < j$ or $i < k$, thus $w \notin L$.

Now that I have (hopefully) exhausted all possibilities, the proof is complete and the language, L, is not context-free. ☐

Problem 5

Let $G = \langle \{S\}, \{a, b\}, R, S \rangle$ be a context-free grammar with the rules, R defined as follows:

$$S \to aS | aSbS | \epsilon$$

Part One Prove that G is ambiguous.

Proof. We can prove that G is ambiguous if there are two left most derivations for one string.

Take $s = aab$, it can be derived twice as follows:

$$S = aS = aaSbS = aabS = aab$$
$$S = aSbS = aaSbS = aabS = aab$$

Thus the context free language, G is ambiguous. □

Part Two Give unambigous grammar that generates the same language as G.

The new grammar can be defined as follows:

$$S \to aT|T$$
$$T \to aSbU|S$$
$$U \to a|b|\epsilon$$

Using the previous example of $s = aab$, we can try to derive it multiple times like so:

$$S = aT = aSbU = aTbU = aabU = aab$$
$$S = T = aSbU = ... = \text{ no way to get aab}$$

And there it can't be derived multiple ways. So unless I'm missing something, I think the homework is finished. YAY!