

COMP 481: Automata, Formal Languages, and Computability

Spring 2009

Solutions to Homework Assignment #5

1. (a) $S \rightarrow T1T1T1T$
 $T \rightarrow 0T|1T|\epsilon$

(b) $S \rightarrow 00S|01S|10S|11S|0|1$

(c) The grammar has no production rules; i.e., $R = \emptyset$.
2. (a) $L(G)$ is the language of all strings w over the alphabet $\Sigma = \{0, \#\}$, where the string either has exactly two $\#$ symbols (the string in this case is in the language $0^* \# 0^* \# 0^*$) or exactly one $\#$ symbol, and the number of 0's after the $\#$ is twice the number of 0's before it (the string in this case is in the language $\{0^m \# 0^{2m} : m \geq 0\}$).
- (b) $L(G) \cap 0^* \# 0^* = L'$, where $L' = \{0^m \# 0^{2m} : m \geq 0\}$. Since L' is not regular (it's easy to prove that, using the Pumping Lemma for regular languages), then $L(G)$ is not regular.
3. $S \rightarrow S + T | S - T | T$
 $T \rightarrow T \times F | T \div F | F$
 $F \rightarrow (S) | x$
4. PDAs are shown in Figure 1.

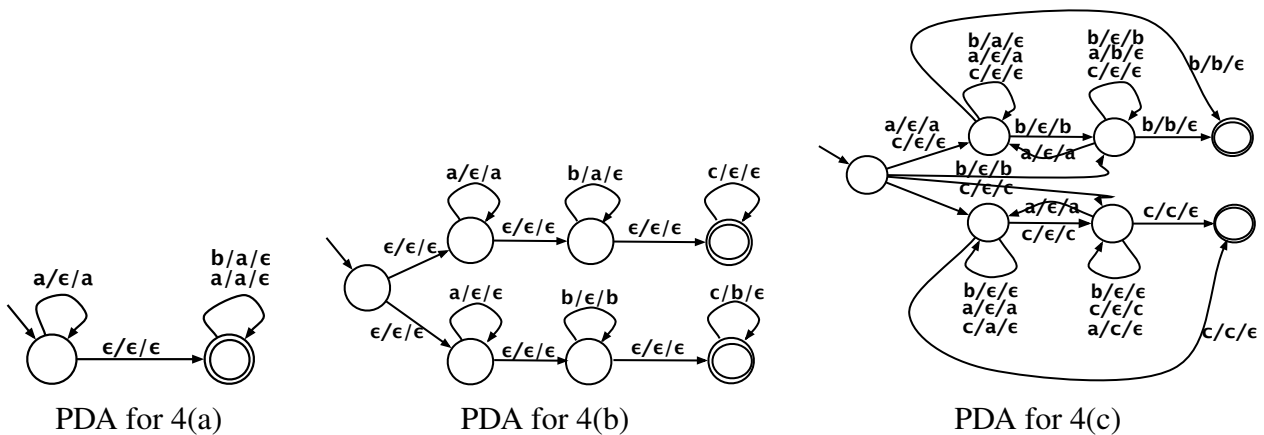


Figure 1: PDAs for problem 4.

5. An NFA for language L would have states of the form (p, α) , where p is a state in M and α is a string of k or fewer stack symbols, representing the current contents of M 's stack. Such a pair provides a complete description of M 's current status, and for any such pair and any possible input, it is possible using M 's definition to specify the pairs that might result.

Furthermore, there are only finitely many such pairs. The initial state of the NFA is (s, ϵ) , and state (p, α) is accepting whenever p is an accepting state of M .

6. DPDAs are shown in Figure 2.

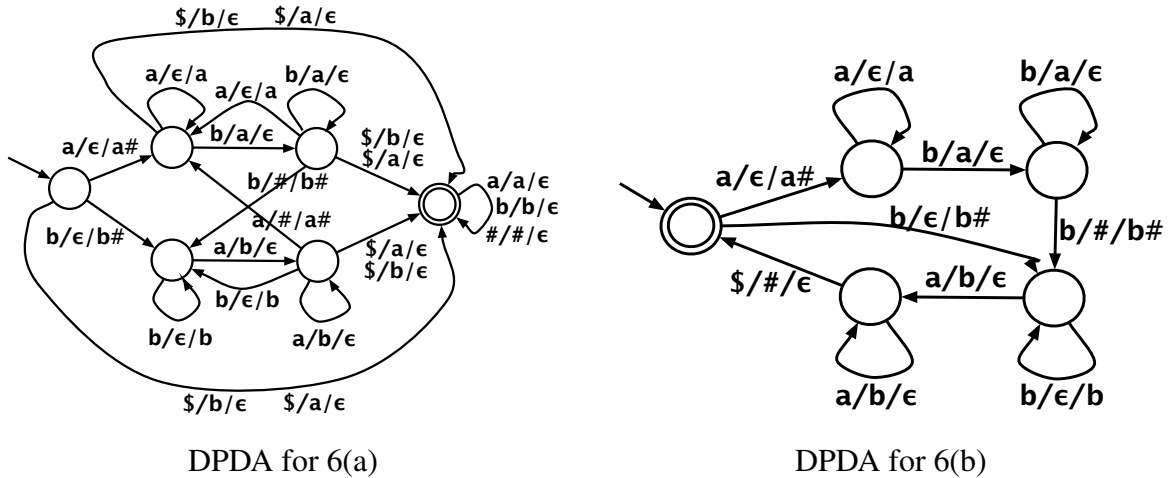


Figure 2: DPDAs for problem 6.

7. We provide only sketches of the proofs. You are responsible for the complete formal proof.

- (a) Not context-free. Apply the Pumping Lemma to string $a^{k^2}b^k$, where k is the Lemma's constant. The only case you have to be careful about is when v and y contain both a 's and b 's—you need to carefully show an $i \geq 0$ that would make the string $uw^ixy^iz \notin L$.
- (b) Context-free. A CFG for the language is:
 $S \rightarrow Tb$
 $T \rightarrow aTa|aTb|bTa|bTb|a.$
- (c) Context-free. The set of all strings of balanced parentheses is a DCFL, and since DCFLs are closed under complement, it follows that the language is a CFL.
- (d) Not context-free. Apply the PL to string $a^k b^k a^k b^k$, where k is the lemma's constant.
- (e) Not context-free. Apply the PL to string $a^k b^k c^k$, where k is the lemma's constant.
- (f) Context-free. Notice that a string w in the language has to satisfy at least one of two conditions:
 - i. w begins with a 1. In other words, $w \in L_1$, where $L_1 = 1(0 \cup 1)^*$.
 - ii. w contains a segment $0^i 1^j$ where $i \neq j$. In other words, $w \in L_2$, where $L_2 = ((0 \cup 1)^* 1)^* 0^i 1^j (0(0 \cup 1)^*)^*$, where $i \neq j$. If there are symbols before the $0^i 1^j$ segment, we force them to end with a 1, and if there are symbols after the $0^i 1^j$ segment, we force them to begin with a 0.

We have that the language is $L_1 \cup L_2$, where L_2 is the concatenation of three languages $L_2 = L_{21}L_{22}L_{23}$, where $L_{21} = ((0 \cup 1)^*1)^*$, $L_{22} = \{0^i1^j : i, j \geq 0, i \neq j\}$, and $L_{23} = (0(0 \cup 1)^*)^*$. Since L_{21} , L_{22} and L_{23} are all CFLs, then so is L_2 . Further, since L_1 is CFL, so is the language in question.

8. PDA is shown in Figure 3.

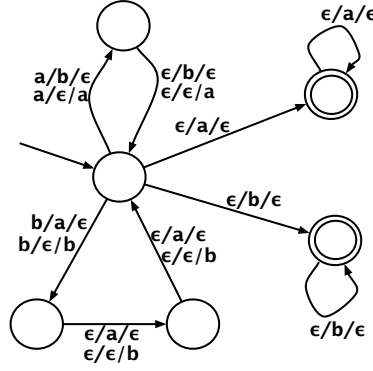


Figure 3: PDA for problem 8.

9. Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA accepting L , with $Q = \{q_1, \dots, q_n\}$. We will construct a PDA to accept that language L' . The idea is to apply a “multi-track” simulation:

- Nondeterministically divide the input w into two parts $w = xz$.
- Simulate M on x and also push x onto the stack. Assume that it ends at state q_i , for some $1 \leq i \leq n$.
- Guess string y and simulate M on y starting at state q_i ; simultaneously simulate M on z starting from state q_j for all $j = 1, \dots, n$. During these simulations, we use string x in the stack to check that $|y| = |z| = |x|$.
- Accept the input if the simulation of y ends at state q_j and the simulation of z starting at q_j ends at a final state.

To implement the above idea, we need the following NFA: we define $M' = (Q, \Sigma, \Delta, q_1, F)$, where $\Delta(q_i, a) = \{\delta(q_i, b) : b \in \Sigma\}$, for $q_i \in Q$ and $a \in \Sigma$. Now, we can define our new PDA as $M' = (Q', \Sigma, \Sigma, \delta', q_1, F')$, where Q' , F' and δ' are defined as follows:

- $Q' = Q \cup Q^{n+1}$ (so that each state is of the form $q_i \in Q$ or $[q_{i_0}, \dots, q_{i_n}]$ with $q_{i_j} \in Q$ for $j = 0, 1, \dots, n$).
- $F' = \{[q_m, q_{i_1}, \dots, q_{i_n}] \in Q^{n+1} : q_{i_m} \in F\}$.
- At each state $q_j \in Q$, $\delta'(q_j, a, \varepsilon) = \{(\delta(q_j, a), a)\}$, for all $a \in \Sigma$.
- At each state $q_j \in Q$, $\delta'(q_j, \varepsilon, \varepsilon) = \{([q_j, q_1, \dots, q_n], \varepsilon)\}$.

(e) At state $[q_j, q_{i_1}, \dots, q_{i_n}] \in Q^{n+1}$, $\delta'([q_j, q_{i_1}, \dots, q_{i_n}], a, b) = \{([q_k, \delta(q_{i_1}, a), \dots, \delta(q_{i_n}, a)], \varepsilon) : q_k \in \Delta(q_j, a)\}$.

10. The minimum value of p is 5. Notice that the derivation of a string $w \in B$ satisfies exactly one of the following two conditions:

- (a) The derivation is of the form $S \Rightarrow TT \Rightarrow^* w$, and variable U is not used.
- (b) The derivation is of the form $S \Rightarrow U \Rightarrow^* w$, and variable T is not used.

In case (1), unless $w = \#\#$, the variable T has to appear at least twice in the derivation, in which case the conditions of the Pumping Lemma would be satisfied. Therefore, the conditions of the Pumping Lemma hold in this case if and only if $|w| \geq 3$ (to ensure that $w \neq \#\#$).

In case (2), unless $w = 0\#00$, the variable U has to appear at least twice in the derivation, in which case the conditions of the Pumping Lemma would be satisfied. Therefore, the conditions of the Pumping Lemma in this case hold if and only if $|w| \geq 5$ (to ensure that $w \neq 0\#00$).

Hence, one of the two variables T and U appears at least twice on path from the root to a leaf in a parse tree of string $w \in B$ if and only if $|w| > \max\{3, 5\}$. In other words, $p = 5$.

Notice that the answer $p = b^{|V|} + 1 = 4^3 + 1 = 65$ is wrong in the sense that this value of the constant would work, but it is not the minimum value.

11. We have shown that for any string $w \in L$, where $|w| = n$, exactly $2n - 1$ steps are required for any derivation of w , when the CFG is in CNF. Hence, if k is the number of derivation steps of a string w by a CFG in CNF, then $|w| = \frac{k+1}{2}$.

Therefore, if G in CNF generates a string w with a derivation having at least 2^b steps, then the string is of length $\geq \lceil \frac{2^b+1}{2} \rceil = 2^{b-1} + 1$. Notice that when the CFG is in CNF, a string w whose length is $|w| = 2^{b-1} + 1$ has a parse tree whose height is $b + 1$ (can be easily proven by induction on b). Therefore, the parse tree must have a path from the root to a leaf in which one variable appears at least twice. From this, the conditions of the Pumping Lemma follow in a similar manner to the general proof of the Pumping Lemma.

12. An example of such a language is

$$L = \{a^i b^j c^k d^l : i, j, k, l \geq 0, \text{ if } i = 1 \text{ then } j = k = l\}.$$

Assume L is a CFL, and let p be the constant. If you choose a string in which $i = 1$, then pumping, when the vxy is in the a 's segment of the string, would not work, since changing the number of a 's would result in a string in which $i \neq 1$, but then the string would be in the language vacuously.

If you choose a string in which $i \neq 1$, then pumping, for example when the vxy is in either of the b 's, c 's, or d 's segment would not work. If there are no such segments, then we already have $j = k = l = 0$, and the string would be in the language no matter what the value of i is.

Therefore, whatever string $w \in L$ is chosen, there will be at least one decomposition $w = uvxyz$, where $uv^ixy^iz \in L$ for every $i \geq 0$.