Concordia University

Dept. of Computer Science and Software Engineering

COMP 335 – Introduction to Theoretical Computer Science

Fall 2018

Solutions for Assignment 3

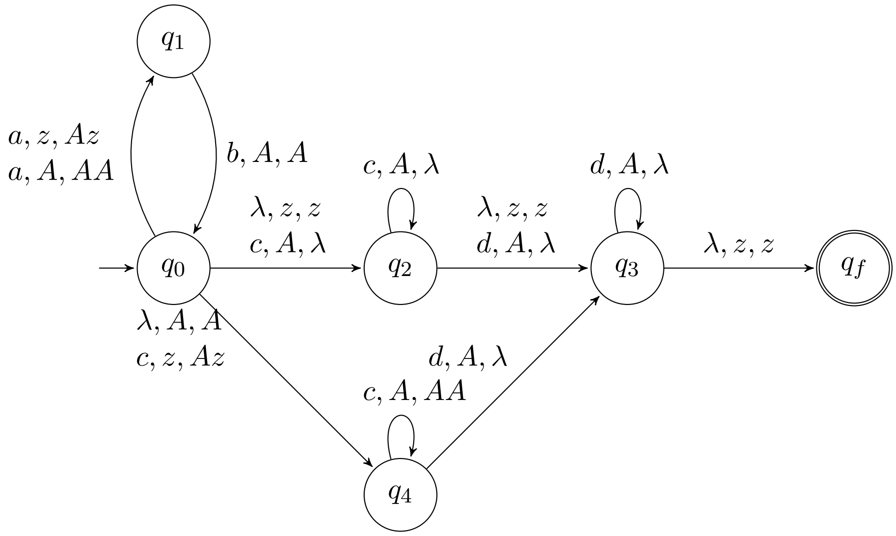
1. (a) The given language is the union of L_1 and L_2 where
$L_1 = \{(ab)^i c^j d^k \ : j = i - k\} = \{(ab)^i c^j d^k \ : i = j + k\}$ and
$L_2 = \{(ab)^i c^j d^k \ : j = k - i\} = \{(ab)^i c^j d^k \ : k = i + j\}$
It is now easy to give a grammar for the given language.
$S \to S_1 \mid S_2$
$S_1 \to AS_1d \mid X$
$X \to AXc \mid \lambda$
$A \to ab$
$S_2 \to AS_2d \mid Y$
$Y \to cYd \mid \lambda$
Here S_1 generates strings in L_1 and S_2 generates strings in L_2.

(b) The given language is the union of L_a and L_b where
$L_a = \{w \in \{a,b\}^* \ : n_a(w) > n_b(w)\}$ and
$L_b = \{w \in \{a,b\}^* \ : n_b(w) > n_a(w)\}$
We give the grammar for L_a here. The grammar for L_b is similar and left as an exercise.
$S \to EaS \mid EaE$
$E \to aEb \mid bEa \mid EE \mid \lambda$
We prove below that this grammar does indeed generate the language L_a (the proof is not required as part of the assignment.) First observe that E generates all strings with equal numbers of a's and b's. It is then easy to see that S derives *only* strings with more a's than b's. Does it generate *all* such strings? We will show by induction that it does. Assume inductively that S derives all strings of length $\leq n$ that have more a's than b's. Now consider a string w of length $n+1$ that has more a's than b's. Let u be the *longest* prefix of w such that u has the same number of a's and b's (note that u could be λ). Then it follows that the next symbol after u must be an a *(Why?)*. That is, $w = uav$ where $u, v \in \{a,b\}^*$. Observe that $|v| < |w| = n+1$, and $n_a(v) \geq n_b(v)$ *(Why?)*. Now, if $n_a(v) = n_b(v)$, we see that $E \overset{\star}{\Rightarrow} u$ and $E \overset{\star}{\Rightarrow} v$, and therefore $S \Rightarrow EaE \overset{\star}{\Rightarrow} uav$. If however $n_a(v) > n_b(v)$, then by the inductive hypothesis, $S \overset{\star}{\Rightarrow} v$, and therefore $S \Rightarrow EaS \overset{\star}{\Rightarrow} uav$. We have shown that in both cases, $S \overset{\star}{\Rightarrow} w$. This completes the proof by induction that S derives all strings with more a's than b's.

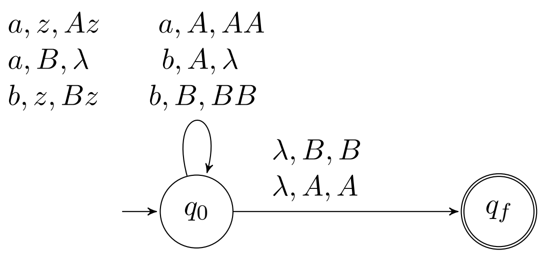
(c) For any string $w \in L$, the condition $|w| = 3n_a(w)$ implies that $2n_a(w) = n_b(w) + n_c(w)$. We will use the variable X to generate either b or c. So we construct a grammar that generates strings with twice as many X's as a's. The following grammar does the job.
$S \to aSXX \mid XXSa \mid XaSX \mid SS \mid \lambda$
$X \to b \mid c$
As in the previous question, it is easy to see that S only generates strings in the language. We still need to be sure that it generates *all* strings in the language. Intuitively, if a string w in the language can be written as $w = avx$ with $x \in \{bb, bc, cb, cc\}$, then v must also be in the given language, and so we conclude (using an inductive argument) that there is a derivation for w starting with production $S \to aSXX$. However, if $w = avx$ with $x \notin \{bb, bc, cb, cc\}$, there must be a non-empty prefix u of w that is in the language, and since $w = uv$ for some v, it follows that

v is also in the language. So using an argument by induction, there is a derivation for w starting with $S \to SS$. A similar argument holds for strings not starting with a.

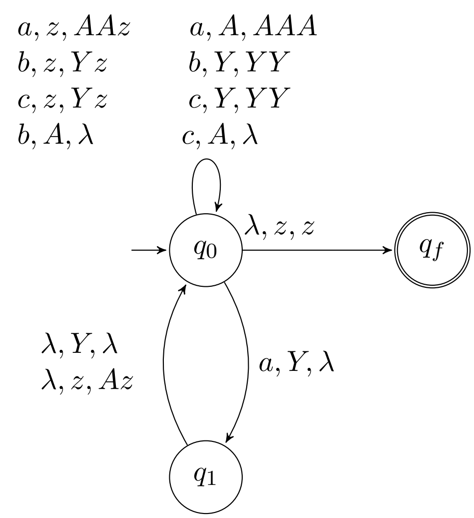
2. (a) The transition diagram for the PDA is given below.

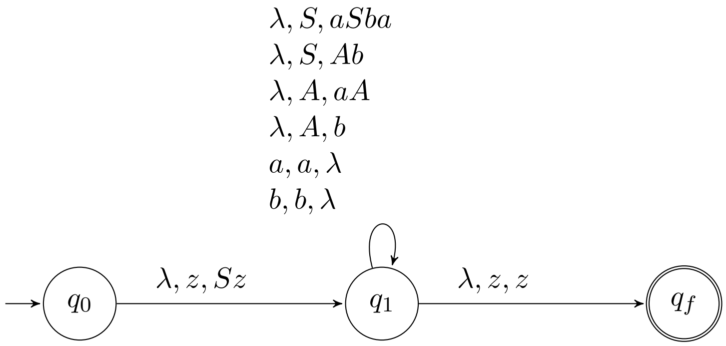


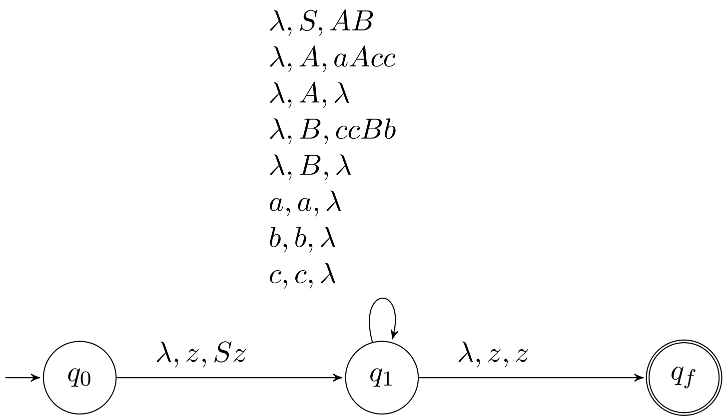
(b) The main idea is that for immediately after processing a prefix v of the input string: if $n_a(v) > n_b(v)$, the stack contains $n_a(v) - n_b(v)$ A's; if $n_b(v) > n_a(v)$, the stack contains $n_b(v) - n_a(v)$ B's; otherwise it contains only a single z.



(c) The idea is similar to the previous question. The stack never contains both A's and Y's. Suppose we have just read the prefix v of the input string, and let $\alpha = n_a(v)$ and $\beta = n_b(v) + n_c(v)$. If $\beta > 2\alpha$, then the stack contains $\beta - 2\alpha$ Y's. If $\beta = 2\alpha$, the stack contains only z. If $\beta < 2\alpha$, the stack contains $2\alpha - \beta$ A's.



3. (a) 

(b) 

4. (a) The string $aabbb$ has two derivation trees (corresponding to the derivations $S \Rightarrow Sb \Rightarrow aaSbb \Rightarrow aabbb$, and $S \Rightarrow aaSb \Rightarrow aaSbb \Rightarrow aabbb$).
   (b) $L(G) = \{a^{2n}b^m \mid m > n\}$
   (c) $S \rightarrow aaSb \mid B$
   $B \rightarrow bB \mid b$

5. S is the only variable, and therefore also the start variable. The set of terminals $T = \{+, (,),^*, a, b, \emptyset, \lambda, \cdot\}$. The productions are:
$S \rightarrow S + S \mid S \cdot S \mid (S) \mid S^* \mid a \mid b \mid \emptyset \mid \lambda$

Note that λ here is a symbol in the alphabet; S does not derive the empty string, but instead it derives the *symbol* λ. It would also be acceptable to leave out the $\cdot$ from the set of terminals, and use the production $S \rightarrow SS$ instead of $S \rightarrow S \cdot S$.

6. (a) $S \rightarrow aSb \mid bSa \mid a \mid b$
   There are neither λ productions nor unit productions. First we remove productions of the type $S \rightarrow w$ where $|w| > 1$ and w contains a terminal to get:
   $S \rightarrow ASB \mid BSA \mid a \mid b$
   $A \rightarrow a$
   $B \rightarrow b$
   Next we remove productions with more than 2 variables on the right hand side. The final CNF grammar is:
   $S \rightarrow AX \mid BY \mid a \mid b$
   $X \rightarrow SB$
   $Y \rightarrow SA$
   $A \rightarrow a$

$B \to b$

(b) $S \to aPbQRaT \mid aS \mid Rb$
$P \to PQR \mid aP \mid PbT \mid \lambda$
$Q \to R \mid bQ \mid \lambda$
$R \to aSb \mid S \mid \lambda$
$T \to ab$

<u>Step 1:</u> The nullable variables are P, Q, and R. We remove the nullable variables to obtain:
$S \to aPbQRaT \mid aS \mid Rb \mid abQRaT \mid aPbRaT \mid aPbQaT \mid abRaT \mid abQaT \mid aPbaT \mid abaT \mid b$
$P \to PQR \mid aP \mid PbT \mid QR \mid PR \mid PQ \mid P \mid Q \mid R \mid a \mid bT$
$Q \to R \mid bQ \mid b$
$R \to aSb \mid S$
$T \to ab$

<u>Step 2:</u> Remove unit productions. We see that $P \Rightarrow Q \Rightarrow R \Rightarrow S$, therefore we get:
$S \to aPbQRaT \mid aS \mid Rb \mid abQRaT \mid aPbRaT \mid aPbQaT \mid abRaT \mid abQaT \mid aPbaT \mid abaT \mid b$
$P \to PQR \mid aP \mid PbT \mid QR \mid PR \mid PQ \mid bQ \mid b \mid aSb \mid a \mid bT$
$P \to aPbQRaT \mid aS \mid Rb \mid abQRaT \mid aPbRaT \mid aPbQaT \mid abRaT \mid abQaT \mid aPbaT \mid abaT \mid b$
$Q \to aSb \mid bQ \mid b$
$Q \to aPbQRaT \mid aS \mid Rb \mid abQRaT \mid aPbRaT \mid aPbQaT \mid abRaT \mid abQaT \mid aPbaT \mid abaT \mid b$
$R \to aSb$
$R \to aPbQRaT \mid aS \mid Rb \mid abQRaT \mid aPbRaT \mid aPbQaT \mid abRaT \mid abQaT \mid aPbaT \mid abaT \mid b$
$T \to ab$

<u>Step 3:</u> We remove productions $S \to w$ where $|w| > 1$ and w contains a terminal to get
$S \to APBQRAT \mid AS \mid RB \mid ABQRAT \mid APBRAT \mid APAQAT \mid ABRAT \mid ABQAT \mid APBAT \mid ABAT \mid b$
$P \to PQR \mid AP \mid PBT \mid QR \mid PR \mid PQ \mid BQ \mid b \mid ASB \mid a \mid BT$
$P \to APBQRAT \mid AS \mid RB \mid ABQRAT \mid APBRAT \mid APAQAT \mid ABRAT \mid ABQAT \mid APBAT \mid ABAT \mid b$
$Q \to ASB \mid BQ \mid b$
$Q \to APBQRAT \mid AS \mid RB \mid ABQRAT \mid APBRAT \mid APAQAT \mid ABRAT \mid ABQAT \mid APBAT \mid ABAT \mid b$
$R \to ASB$
$R \to APBQRAT \mid AS \mid RB \mid ABQRAT \mid APBRAT \mid APAQAT \mid ABRAT \mid ABQAT \mid APBAT \mid ABAT \mid b$
$T \to AB$
$A \to a$
$B \to b$

<u>Step 4:</u> Remove productions with more than 2 variables on the right hand side. The final CNF grammar is:

$S \to AX_1 \mid AS \mid RB \mid AX_2 \mid AX_6 \mid AX_8 \mid AX_7 \mid AX_{11} \mid AX_{12} \mid AX_{13} \mid b$
$X_1 \to PX_2$
$X_2 \to BX_3$
$X_3 \to QX_4$
$X_4 \to RX_5$
$X_5 \to AT$
$X_6 \to PX_7$
$X_7 \to BX_4$
$X_8 \to PX_9$
$X_9 \to AX_{10}$

$X_{10} \to QX_5$
$X_{11} \to BX_{10}$
$X_{12} \to PX_{13}$
$X_{13} \to BX_5$
$P \to PX_{14} \mid AP \mid PX_{15} \mid QR \mid PR \mid PQ \mid BQ \mid b \mid AX_{16} \mid a \mid BT$
$X_{14} \to QR$
$X_{15} \to BT$
$X_{16} \to SB$
$P \to AX_1 \mid AS \mid RB \mid AX_2 \mid AX_6 \mid AX_8 \mid AX_7 \mid AX_{11} \mid AX_{12} \mid AX_{13} \mid b$
$Q \to AX_{16} \mid BQ \mid b$
$Q \to AX_1 \mid AS \mid RB \mid AX_2 \mid AX_6 \mid AX_8 \mid AX_7 \mid AX_{11} \mid AX_{12} \mid AX_{13} \mid b$
$R \to AX_{16}$
$R \to AX_1 \mid AS \mid RB \mid AX_2 \mid AX_6 \mid AX_8 \mid AX_7 \mid AX_{11} \mid AX_{12} \mid AX_{13} \mid b$
$T \to AB$
$A \to a$
$B \to b$

7. This can be proved by strong induction on n, the length of the longest path (from the root to a leaf). For the base case, we take $n = 1$. Since G is in CNF, the derivation tree corresponds to a derivation $S \to a$ for some terminal a. Then the length of the derived string is $1 \leq 2^{1-1}$ as needed. Now for the inductive step. Suppose for any $i \leq n$, it is true that a derivation tree in which the longest path is of length i, has yield of length at most 2^{i-1}. Now consider a derivation tree T in which the longest path is of length $n+1$. Since G is in CNF, the first step in the derivation must use a production of the form $S \to AB$, so the children of the root node in T are two variables A and B. Call the two two sub-trees rooted at A and B the trees T_1 and T_2. Clearly the longest paths in T_1 and T_2 are of length at most n. Let the yield of T_1 be w_1 and the yield of B be w_2. By the induction hypothesis, $|w_1| \leq 2^{n-1}$ and $|w_2| \leq 2^{n-1}$. Since $w = w_1 w_2$, we conclude that $|w| = |w_1| + |w_2| \leq 2^{n-1} + 2^{n-1} = 2^n$ as needed.

8. Assume without loss of generality that G is in CNF, except possibly for the production $S \to \lambda$. We now construct the grammar G'. For every variable A in the grammar G, we introduce a new variable A' and include both A and A' as variables in G'. The job of the variable A' is to generate prefixes of sentential forms generated by A. The start variable of the grammar G' is the variable S'.

   Since G is in CNF (except possibly for $S \to \lambda$), there are three types of productions:

   (a) $A \to BC$. Corresponding to this production, we include the following productions in G':
   $A \to BC$
   $A' \to BC'$
   $A' \to B'$

   (b) $A \to a$. Corresponding to this production, we include the following productions in G':
   $A \to a$
   $A' \to a$
   $A' \to \lambda$

   (c) $S \to \lambda$. If this production is in G, we also include it in G'.