CS 3133: Homework 5

Adam Camilli (aocamilli@wpi.edu)

October 9, 2017

1. **7.17.b** (pg. 249) Use the Pumping Lemma to prove that $L = \{a^i b^j c^i d^j \mid i, j \geq 0\}$ is not context-free.

   (a) Assume indirectly that L is context-free. Therefore $\exists\ z \in L$ such that

      i. $|z| > k$

      ii. $z = uv^n wx^n y \in L, n \geq 0$

      iii. $|vwx| \leq k$

      iv. $|vw| > 0$

   (b) Taking $z = a^k b^k c^k d^k$, we get by the property (iii) that v cannot contain both a and c, and that x cannot contain both b and d. By property (iv) we also know that one element $\{v, x\}$ must be of non-zero length.

   (c) Therefore, for any $n \neq 1$, we obtain a contradiction: For $uv^0 wx^0 y$ we must either have less a's than c's or vice versa, or we must have less b's than d's or vice versa. For any $n > 1$, we simply replace "less" with "more" in the previous statement.

QED

2. Let M be the Turing machine defined by

δ	B	a	b	c
q_0	(q_0, B, R)	(q_0, a, R)	(q_0, b, R)	(q_1, c, L)
q_1	(q_2, B, R)	(q_1, b, L)	(q_1, a, L)	-
q_2	-	-	-	-

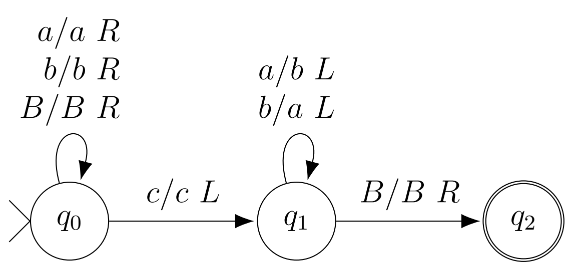
(a) Trace the computation for the input string $abcab$.

$$
\begin{array}{rr}
\boldsymbol{q_0}BabcabB & \rightarrow \\
\vdash B\boldsymbol{q_0}abcabB & \rightarrow \\
\vdash Ba\boldsymbol{q_0}bcabB & \rightarrow \\
\vdash Bab\boldsymbol{q_0}cabB & \rightarrow \\
\vdash Bab\boldsymbol{q_1}cabB & \leftarrow \\
\vdash Ba\boldsymbol{q_1}acabB & \leftarrow \\
\vdash B\boldsymbol{q_1}bacabB & \leftarrow \\
\vdash \boldsymbol{q_2}BbacabB & \text{HALT}
\end{array}
$$

(b) Trace the first six transitions of the computation for the input string $abab$.

$$
\begin{array}{rr}
\boldsymbol{q_0}BababB & \rightarrow \\
\vdash B\boldsymbol{q_0}ababB & \rightarrow \\
\vdash Ba\boldsymbol{q_0}babB & \rightarrow \\
\vdash Bab\boldsymbol{q_0}abB & \rightarrow \\
\vdash Baba\boldsymbol{q_0}bB & \rightarrow \\
\vdash Babab\boldsymbol{q_0}B & \rightarrow \\
\vdash BababB\boldsymbol{q_0} & \rightarrow \\
\ldots &
\end{array}
$$

(c) Give the state diagram of M and describe the result of a computation in M.



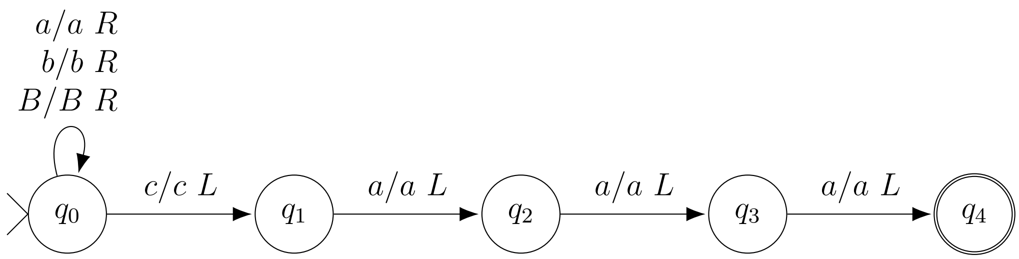
This Turing machine, upon reading c, replaces every a with b and vice versa before the first c and halts when finished (i.e. upon reading a blank). If there is not at least one c, the machine will never halt and thus it will be an infinite computation.

3. Construct a Turing machine with input alphabet $\{a, b, c\}$ that accepts strings in which the first c is immediately preceded by the substring aaa. A string must contain a c to be accepted by the machine.

   Let this Turing machine be defined by transition table

δ	B	a	b	c
q_0	(q_0, B, R)	(q_0, a, R)	(q_0, b, R)	(q_1, c, L)
q_1	-	(q_2, a, L)	-	-
q_2	-	(q_3, a, L)	-	-
q_3	-	(q_4, a, L)	-	-
q_4	-	-	-	-

   and state diagram

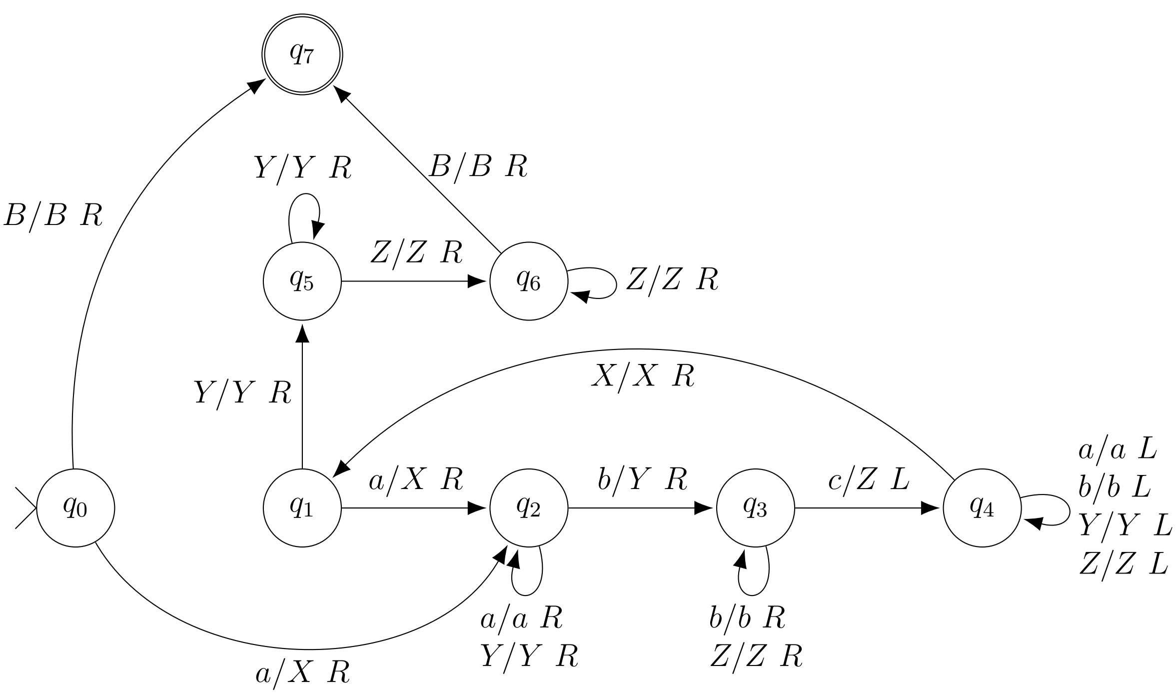


4. Construct a Turing machine with input alphabet $\{a, b, c\}$ that accepts the language $L = \{a^i b^i c^i \mid i \geq 0\}$ by halting only.

   This language is the typical example of a non-context free language, which require matching of three symbols a, b, and c which is beyond the capabilities of regular expressions and even one-stack PDAs to emulate. Turing machines, however, are more than up to the task: By associating each symbol in the language's alphabet with tape alphabet variables X, Y, and Z, one can easily keep track of the numbers of each symbol such that they each appear i times. An algorithm for this process is as follows:

   i. Halt on λ and accept.
   ii. On first a, overwrite with X, on all other a's and Y's move right. If b go to step (iii), else halt and reject.
   iii. On first b, overwrite with Y, on all other b's and Z's move right. If c go to step (iv), else halt and reject.
   iv. On first c, overwrite with Z, then move left until X is read.
   v. On a, repeat steps (ii-iv). On Y, move right on Y's and Z's until B is read, then halt.

   This algorithm can be represented by the following state diagram:



5. Construct a standard Turing machine that accepts the set of palindromes over $\{a, b\}$.

An even-length palindrome will be of the form ww^R, and an odd one of the form $w(a \cup b)w^R$. In both cases, a Turing machine is going to need to match each symbol of w with its corresponding symbol in w^R. One algorithm for this is as follows:

i. If λ, halt and accept.

ii. If a, overwrite with B and remember that starting symbol was a. Move right through entire input tape until B is read. Upon reading B, read left. If a, overwrite with B and move left through entire input tape until B is read, then read right. Else halt and reject.

iii. If b, overwrite with B and remember that starting symbol was b. Move right through entire input tape until B is read. Upon reading B, read left. If b, overwrite with B and move left through entire input tape until B is read, then read right. Else halt and reject.

iv. Repeat steps (ii) and (iii) until machine halts. If input tape consists of only blanks, or a single a or single b, accept. Else reject.

This algorithm can be represented with the following state diagram: