COMP 3803 - Assignment 4 Solutions

Solutions written in LaTeX

Last updated on April 6, 2015

1. **Q:** Prove, using the pumping lemma for CFLs, that the following are not context-free:

   A:

   - $L = \{a^n b^{2n} c^m d^{2n} : m, n \geq 0\}$:
     Suppose that L is context-free. Let $p \geq 1$ be the pumping length as given by the pumping lemma for CFLs. Let $s = a^p b^{2p} d^{2p}$. Since $s \in L$ and $|s| = 5p > p$, then by the pumping lemma we can write $s = uvxyz$, where:
     - $|vy| \geq 1$;
     - $|vxy| \leq p$;
     - and $uv^i xy^i z \in L$ for every $i \geq 0$.

     Then:
     - Suppose that vxy contains no d. Then, uv^2xy^2z contains more than $2p$ b's, or more than p a's, so that $uv^2xy^2z \notin L$.
     - Suppose that vxy contains no a. Then, uv^2xy^zx contains more than $2p$ b's, or more than $2p$ d's, so that $uv^2xy^2z \notin L$.
     - The only other case is that vxy contains both an a and a d, but then vxy must consist of at least $2p$ b's, i.e. $|vxy| \geq 2p + 2 > p$, contradicting the fact that $|vxy| \leq p$.

     In every case, we derive a contradiction, so L cannot be context-free. $\square$
   - $L = \{a^k b^m c^n : k = \max\{m, n\}\}$:
     Suppose that L is context-free. Let $p \geq 1$ be the pumping length as given by the pumping lemma for CFLs. Let $s = a^p b^p c^p$. Since $s \in L$ and $|s| = 3p > p$, then by the pumping lemma we can write $s = uvxyz$, where:
     - $|vy| \geq 1$;
     - $|vxy| \leq p$;
     - and $uv^i xy^i z \in L$ for every $i \geq 0$.

     Then:
     - Suppose that vxy contains only b's and c's. Then, uv^2xy^2z has only p a's, but more than p b's or more than p c's, so that $uv^2xy^2z \notin L$.

- Suppose that vxy contains only a's and b's. If vy contains some a's, then uxz has fewer than p a's while still containing p c's, so that $uxz \notin L$. If vy contains no a's – and thus contains only b's – then uv^2xy^2z contains only p a's but more than p b's, so that $uv^2xy^2z \notin L$.

In every case, we derive a contradiction, so L cannot be context-free. □

2. **Q:** Construct (deterministic or non-deterministic) pushdown automata that accept the following languages:

   A:

   Note: in every case, any unspecified transition is implied to be a stationary loop that rejects the input string.

   - $L = \{c^k a^n b^m : k \geq 0, n \geq m \geq 0\}$:
     We have states $\{q_c, q_a, q_b\}$, where q_k is responsible for processing the symbol k. Thus, q_c is the starting state, and we have the following transitions

$$
\begin{aligned}
&q_c c\$ \to q_c R\$ \\
&q_c a\$ \to q_a N\$ \\
&q_c \square\$ \to q_c N\varepsilon \\
\\
&q_a a\$ \to q_a R\$S \\
&q_a aS \to q_a RSS \\
&q_a bS \to q_b NS \\
&q_a \square\$ \to q_a N\varepsilon \\
&q_a \square S \to q_a N\varepsilon \\
\\
&q_b bS \to q_b R\varepsilon \\
&q_b \square\$ \to q_b N\varepsilon \\
&q_b \square S \to q_b N\varepsilon
\end{aligned}
$$

   - $L = \{a^n b^{2m} c^m : m, n \geq 0\}$:
     We have states $\{q_a, q_b, q_c, q_c'\}$, where q_k is responsible for processing the symbol

k. Thus, q_a is the starting state, and we have the following transitions:

$$\begin{aligned}
&q_a a\$ \rightarrow q_a R\$\\
&q_a b\$ \rightarrow q_b N\$\\
&q_a \square\$ \rightarrow q_a N\varepsilon\\
\\
&q_b b\$ \rightarrow q_b RS\\
&q_b bS \rightarrow q_b RSS\\
&q_b cS \rightarrow q_c NS\\
\\
&q_c cS \rightarrow q_c' N\varepsilon\\
&q_c' cS \rightarrow q_c R\varepsilon\\
&q_c \square\$ \rightarrow q_c N\varepsilon
\end{aligned}$$

- $L = \{c^k ww^R : w \in \{a,b\}^*\}$:
  We have the states $\{q_c, q_1, q_2\}$, where q_c is the start state responsible for processing c's; q_1 is the next state responsible for processing a string; and q_2 is the state responsible for processing its reverse:

$$\begin{aligned}
&q_c c\$ \rightarrow q_c R\$\\
&q_c a\$ \rightarrow q_1 N\$\\
&q_c b\$ \rightarrow q_1 N\$\\
&q_c \square\$ \rightarrow q_c N\varepsilon\\
\\
&q_1 a\$ \rightarrow q_1 RA\\
&q_1 aA \rightarrow q_1 RAA\\
&q_1 bA \rightarrow q_1 RAB\\
&q_1 aB \rightarrow q_1 RBA\\
&q_1 bB \rightarrow q_1 RBB\\
&q_1 aA \rightarrow q_2 NA\\
&q_1 bB \rightarrow q_2 NB\\
\\
&q_2 aA \rightarrow q_2 R\varepsilon\\
&q_2 bB \rightarrow q_2 R\varepsilon\\
&q_2 \square\$ \rightarrow q_2 N\varepsilon
\end{aligned}$$

3. **Q:** Construct a one-tape Turing machine that accepts the language $L = \{a^{2n}b^n : n \geq 0\}$, where $\Sigma = \{a, b\}$. You have to give the informal description, describing the states

you are going to use, their meaning, and what kind of transitions it will have. This will give you partial credit. But you have to also work out all the details of the actual transitions. You can assume that the head of the machine is at the start of the string.

A:

The machine will execute computation in the following steps:

(1) Accept the string if it is empty.
(2) Locate the leftmost character, and delete it if it is an a.
(3) Locate its neighbouring a and delete it.
(4) Locate the rightmost character, and delete it if it is a b.
(5) Return to step (1).

If any of the above steps fails, we reject the input. It should be clear that the accepted strings are of the desired form.

More specifically, we have the following states:

- q_a: starting state; expecting an a in the leftmost position, and delete it.
- q_a': the neighbouring character is an a, and delete it.
- q_R: locating the rightmost character.
- q_b: expecting a b in the rightmost position, and delete it.
- q_L: locating the leftmost character.
- q_{accept}: accept state.
- q_{reject}: reject state.

The following transitions describe a Turing machine that computes according to the algorithm described above (and as usual, every unmarked transition leads to q_{reject}):

$$q_a a \to q_a' \Box R$$
$$q_a \Box \to q_{accept}$$

$$q_a' a \to q_R \Box R$$

$$q_R a \to q_R a R$$
$$q_R b \to q_R b R$$
$$q_R \Box \to q_b \Box L$$

$$q_b b \to q_L \Box L$$

$$q_L a \to q_L a L$$
$$q_L b \to q_L b L$$
$$q_L \Box \to q_a \Box R$$

4. **Q:** Construct a one-tape Turing machine that accepts the language $L = \{w : w \text{ has more 1's than 0's}\}$, where $\Sigma = \{0, 1\}$. You have to give the informal description, describing the states you are going to use, their meaning, and what kind of transitions it will have. This will give you partial credit. But you have to also work out all the details of the actual transitions. You can assume that the head of the machine is at the start of the string.

A:

The machine will execute the computation in the following steps:

(1) Locate the leftmost unmarked 1 and mark it. If there is none, we reject.

(2) Return to the beginning of the string.

(3) Locate the leftmost unmarked 0 and mark it. If there is none, we accept.

(4) Return to the beginning of the string.

(5) Return to step (1).

It should be clear that the accepted strings are of the desired form.

More specifically, we have the following states:

- q_1: staring state; locating the leftmost unmarked 1 and marking it.
- q_{back}: returning to the beginning of the string to search for a 0.
- q_0: locating the leftmost unmarked 0 and marking it.
- q'_{back}: returning to the beginning of the string to search for a 1.
- q_{accept}: accept state.
- q_{reject}: reject state.

The following transitions describe a Turing machine that computes according to the algorithm described above:

$$\begin{aligned}
&q_1 0 \to q_1 0 R\\
&q_1 1 \to q_{back} + L\\
&q_1 + \to q_1 + R\\
&q_1 \square \to q_{reject}\\
\\
&q_{back} 0 \to q_{back} 0 L\\
&q_{back} 1 \to q_{back} 1 L\\
&q_{back} + \to q_{back} + L\\
&q_{back} \square \to q_0 \square R\\
\\
&q_0 0 \to q'_{back} + L\\
&q_0 1 \to q_0 1 R\\
&q_0 + \to q_0 + R\\
&q_0 \square \to q_{accept}\\
\\
&q'_{back} 0 \to q'_{back} 0 L\\
&q'_{back} 1 \to q'_{back} 1 L\\
&q'_{back} + \to q'_{back} + L\\
&q'_{back} \square \to q_1 \square R
\end{aligned}$$

5. **Q:** Construct a two-tape Turing machine that accepts the language $L = \{a^{2n}b^n : n \geq 0\}$, where $\Sigma = \{a, b\}$. You have to give the informal description, describing the states you are going to use, their meaning, and what kind of transitions it will have. But you do not have to also work out all the details of the actual transitions. You can assume that the head of the machine is at the start of the string. Observe that you must use both tapes for reading and writing, and not just ignore the second tape and resort to the one-tape solution of Question 3.

   A:

   The idea is to use the second tape to predict what we should be seeing on the input tape, as we read the string from left to right. Broadly speaking, the algorithm proceeds as follows:

   (1) Read two a's in the first tape and write one b in the second tape. If two a's are not seen, reject.

   (2) Repeat step (1) until a b is seen in the first tape.

   (3) Read a b in the first tape and delete a b from the second tape. If an a is seen in the first tape, reject.

(4) If we have reached the end of the first tape, and we have deleted all b's from the second tape, then accept.

More specifically, the states will behave as follows:

- q_{start}: starting state; if the first tape reads $\square$, we accept. If it reads an a, then switch to q_a, and otherwise reject. Note that this state only exists for our machine to accept the empty string.
- q_a: if the first tape reads an a, move the first tape to the right and switch to q'_a. If the first tape reads a b, switch to q_b.
- q'_a: if the first tape reads an a, write a b on the second tape and move both tapes to the right, and switch to q_a. Otherwise, reject.
- q_b: if the first tape reads a b, move the first tape to the right and move the second tape to the left. If both tapes read $\square$, then accept. Otherwise, reject.
- q_{accept}: accept state.
- q_{reject}: reject state.