# CS 4510: Automata and Complexity

**Spring 2015**

Extra Home work 4 // Due: Friday, March 27, 2015

Sagar Laud, slaud3, 902792910

1. (10 points)

   We will prove that this is regular by defining a DFA for it. Rather than drawing one, I will just define it formally. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA which accepts our given language such that:

   - $Q = \{q_0, q_1, q_2, q_3, q_4\}$
   - $\Sigma = \{0, 1\}$
   - $\delta$ is defined as follows:
     - $\delta(q_0, 0) = q_1$
     - $\delta(q_0, 1) = q_2$
     - $\delta(q_1, 0) = q_1$
     - $\delta(q_1, 1) = q_3$
     - $\delta(q_2, 0) = q_4$
     - $\delta(q_2, 1) = q_2$
     - $\delta(q_3, 0) = q_1$
     - $\delta(q_3, 1) = q_3$
     - $\delta(q_4, 0) = q_4$
     - $\delta(q_4, 1) = q_2$
   - $q_0$ is the start state
   - $F = \{q_1, q_2\}$

2. (10 points) The grammar for this is rather simple. We can do it using only one variable and two terminals. The grammar is as follows: $S \rightarrow 0S1S|1S0S|\epsilon$

3. (15 points)

   This is a rather tough problem. We have to look at the proof for the pumping lemma in order to solve it. The pumping lemma proof relies on the parse tree for a given string in a context free language and shows that if a variable is repeated once in the derivation, then we can loop on the derivation and pump the string. This proof relies on one more specific, we repeat the variable in the derivation twice, in that it occurs a total of 3 times in the derivation. Essentially, our parse tree will have a height 2V+1 where V is the number of variables in the derivation. So, atleast one variable must occur 3 times total. We can use this fact to prove this by showing a conversion from the traditional pumping lemma into the stronger case of the pumping lemma. For every case in the pumping lemma, there exists a corresponding case for the strong pumping lemma. Let us prove this. Begin with the traditional pumping lemma, where either v or y must not be $\epsilon$. So, let us assume that v = $\epsilon$ and y $\neq \epsilon$. Let us define p as the pumping lemma constant. Then, for a given string s in our language, there exists a partition of s = $uv^ixy^iz \in L$ for all $i \geq 0$, $|vxy| \leq p$, $|vy| \geq 0$. In our particular case, v = $\epsilon$ so our string s = $uxy^iz$. Now, let us define some new variables. Let us define $a = ux$, $b = c = d = y$, $e = z$. So now, let us show our conversion. When v = $\epsilon$, s = $uxy^{2i+1}z = ab^icd^ie \in L$ for all $i \geq 0$. A similar situation results from y = $\epsilon$ and v does not. Either way, every condition in the traditional pumping lemma can be converted to one of the stronger form. Since we have shown this equivalence, we have proven that the stronger case holds as well for the same class of languages thereby completing the proof.

4. (15 points)

   This is not too difficult. Begin by converting the CFG to it's corresponding PDA, let us call it M. Then, let us construct a DFA which accepts the complement of D, calling it D'. Then, we will construct another PDA M' which accepts the intersection of M and D' (we saw how to construct this PDA in class). This PDA now accepts all strings which are in M but not in D. Finally, take this PDA, M', and convert it back into a CFG. Now, if this CFG generates the empty language, then L(G) is a subset of L(D). So, if this CFG generates the empty language, then return true because L(G) is a subset of L(D), otherwise return false.

5. (15 points)

- a) The input is represented on the tape the same way as it is with a traditional DTM except that the character which the head is currently over will have a circle in the top right corner.
- b) The key here is to keep the read head moving before the write head. So, we begin with both heads in their starting positions. We read a symbol via FH and move FH to the right. Then, we write the symbol we just read with BH and move to the right as well. Essentially, we are reading and writing the characters back onto the tape as we go. Once we read the symbol which the head is currently over, the symbol with the circle in the top right hand corner, instead of duplicating the symbol, we write a b with BH and move to the right. Then, the next symbol we read must be rewritten with a circle over it. So, we read the next space with FH and move to the right, and write that symbol with a circle over it with BH and move to the right. From here, we do what we did till we read the rest of the string, essentially duplicate the symbols on the tape till we hit the end of the tape. Then we enter state q.
- c) This is very similar to the last problem except that we need to think ahead a little bit more. We need to add some aspect of lookahead into the equation. We still want FH to go first and be followed by BH. Just like in the previous solution, we duplicate the tape until we read the desired marked symbol. But, we have to move left this time so we have to store the symbol in the state machine. We begin in almost the same way as before. The logical difference this time arises from the concept of lookahead. Every time we read a symbol with FH, we do not write it directly to tape but replace what is stored in our state machine with this symbol. After doing this, we write the symbol stored in our state machine with BH. Naturally the very first step we replace the empty value stored in memory with the first symbol on the tape. We continue this, again essentially copying over the tape until we detect the marked symbol. In this case, we do not replace our symbol in memory. Instead, we write the symbol in memory with a circle at the top right corner indicating that the head is now over this current symbol. We then read via FH and make no change whatsoever to items in memory (this is just to move past the space that we are writing to in our new block). We then move BH to the right, write a b, and move one more step to the right after writing. Now, FH is past the point where we needed to make any modifications as is BH. So, we continue as we did in part b, we just copy over the rest of the input. FH will read a symbol, and BH will write that symbol to the tape. This will continue till the end of the input is reached and the final symbol of the input has been written. Then, we enter state q.