Assignment 7
Turing Machines

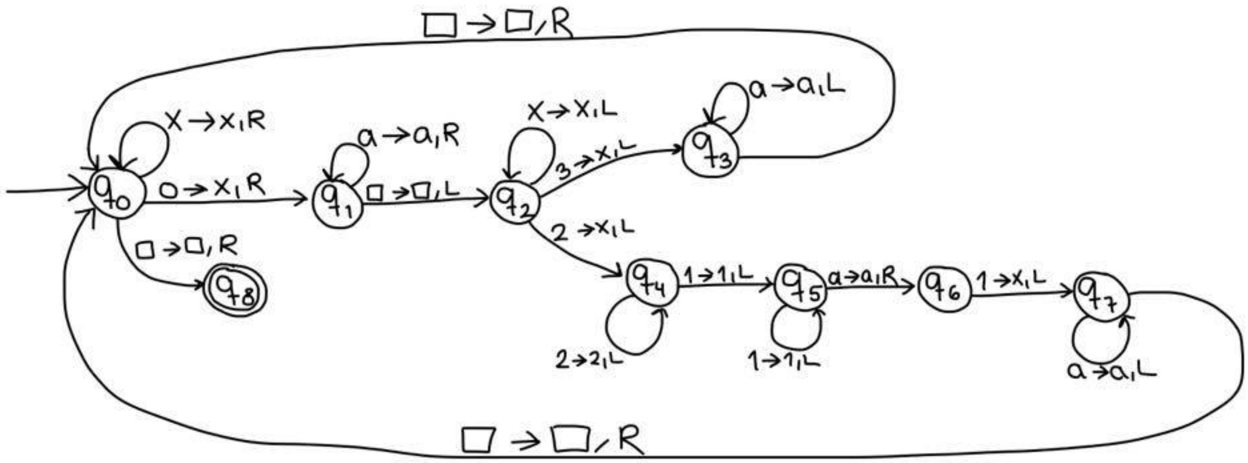
Josefin Ulfenborg
940806-5960
yunalescca@gmail.com

2017-05-24

Task 1

A short description of the language: In the word we must have as many 1's as 2's, and as many 0's as 1's/2's and 3's together.

a) Here is the TM I have constructed for this language. Note that $a \in \{0, 1, 2, 3, X\}$



b) High-level description of this TM:

1. If from the initial state, q_0, we read $\square$, then the word is empty so we move to q_8 and accept the word. Or, if we have been through the whole word and marked the symbols with X, we also move to the final state.

   Else, the word must start with a 0, so when we have read the first 0, we mark it with X and move to q_1

2. When in q_1 we move over the rest over the word until we encounter a $\square$, and then move one step back onto the last symbol in the word (state q_1)

3. From q_2 we will either encounter a 3, a 2 or an X. If 3 or 2 we mark it with an X and then move back one step in the word in to state q_3 respective q_4.

   Otherwise, if this is not the first run, we move back over the X's until we encounter another 3 or 2. This prevents wrong input such as just 1's.

4. In q_3, we move back until the beginning of the word again, and when we are in the beginning we move onto the first symbol again (back in q_0), and start the process over again.

5. Otherwise, if we read a 2, we are in q_4. From here we may have more 2's and so we need to move back past all 2's and so when we finally encounter a 1', we move to the left. This also prevents wrong input, such as 1's without 2's.

6. In q_5 we may either be on another 1 or a 0 (or X), and so if we are on a 1, we move past back all the 1's as well. Additionally, as soon as we are on a 0 (or X), we move back onto the first 1. This, together with the above, makes sure that we have an equal amount of 1's and 2's.

7. In q_6 we mark the first 1 with an X as well, and then move back to the first symbol before the first 1.

8. Finally in q_7 we move back over the word as we did in q_3, and when we encounter a $\square$, we move back onto the first symbol in the word.

Like this, I will always mark the first 0 in the line of 0's. Then, if we have both 1's, 2's and 3's, I will first mark the 3's. When marking the 3's we start with the last 3 in the line of 3's, and the same tactic goes for the 2's. However, when marking the 1's we start with the first 1 in the line of 1's. So for every 0 I mark either a 3 or a 2 (and 1, this way I must have an equal amount of 1's and 2's).

c) Yes, this is a Turing decider. It will receive input of finite length and if it does not find a match then it stops. For instance, if the word would just be *12*, then from the starting state, it is not defined what to do if it reads 1 as first input, and hence it will halt. This will be the case anywhere in the TM, that if it finds input for which it has not been defined (words that are not in the language).