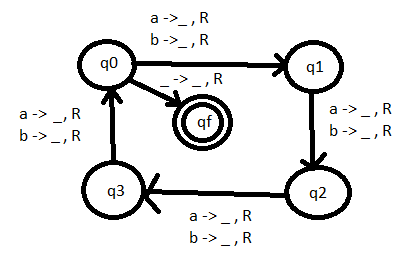
CSci 435: Formal Languages and Automata

Instructor: Dr. M. E. Kim Name: Sam Dressler

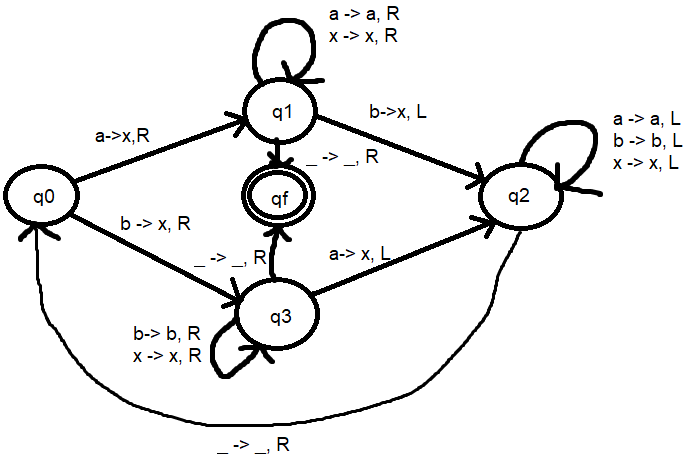
**Home Assignment 7: 120 points + 25 points (optional)**

Q1. [20] For a given language below, construct a TM with a *single final state* that accepts it.

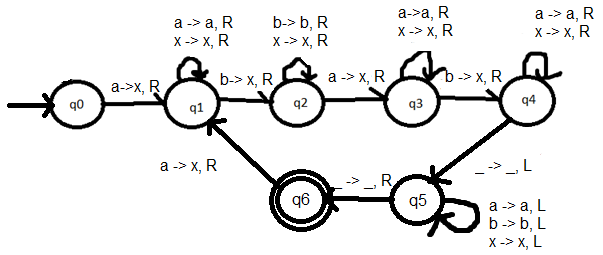
1. [6] L = {w ||*w*|is a multiple of 4} where Σ = {*a*, *b*}.



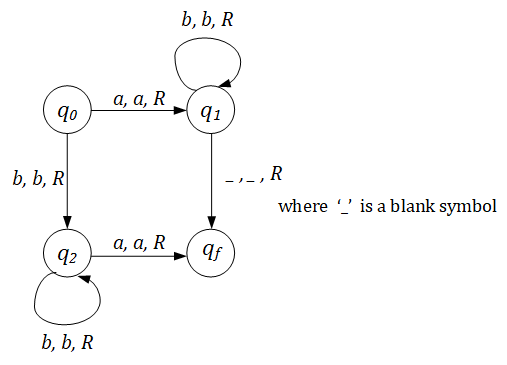
1. [7] L = {w | *na*(*w*) ≠ *nb*(*w*)} where Σ = {*a*, *b*}.



1. [7] L = {w | *anbn anbn* | *n* ≠ 0} where Σ = {*a, b*}.



Q2. [10] What language is accepted by the Turning machine whose transition graph is in the figure?

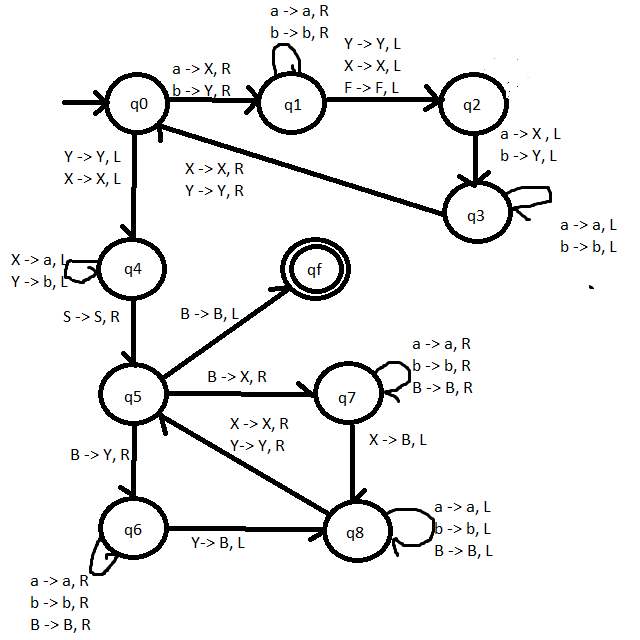


L = {w | ab\* + bb\*a}

Q3. [10] Construct a TM that accepts L = {ww | w ∈ {*a, b*}+ }.

Hint: This is a standard deterministic TM.

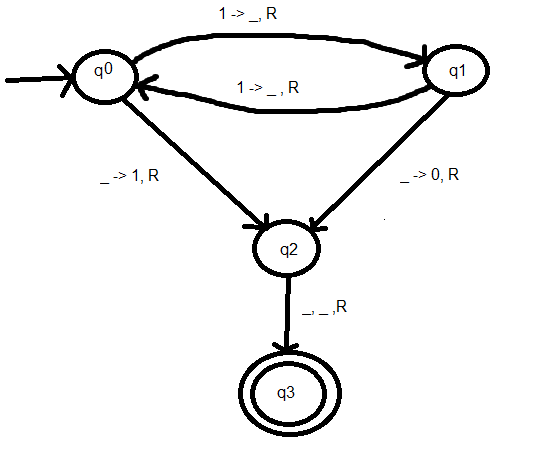
So, TM must **find** the middle of the string first; then, compare two halves.



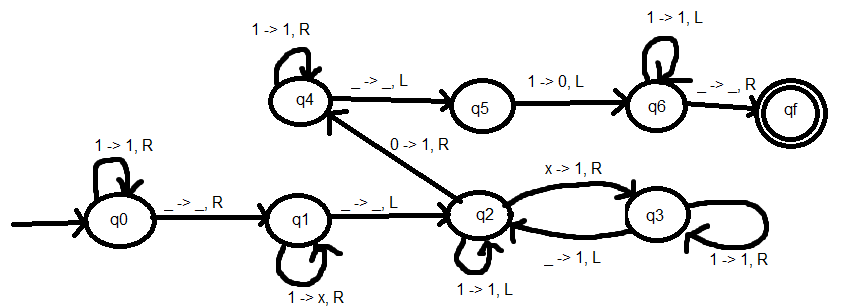
Q4. [20] Construct a TM that computes the following function

1. [10] .

The input *w* is in the unary representation.

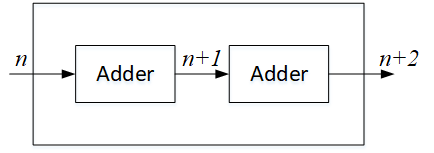


1. [10] *f* (*x, y*) = *x* + 2*y.*

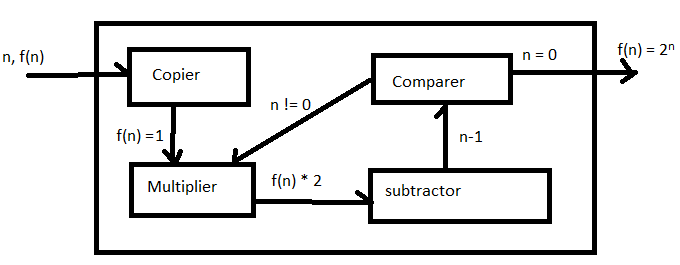


Q5. [20] Using adders, subtracters, comparers, copiers or multipliers, draw block diagram for TM that compute the functions:

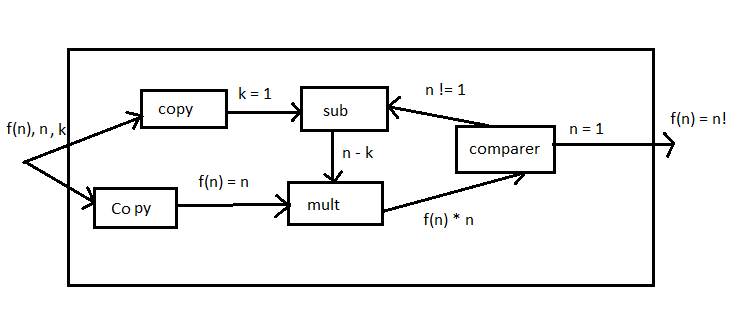
e.g.) *f*(*n*) = *n* + 2



1. [10]*f*(*n*) = *2n.*



1. [10] *f(n) = n!*

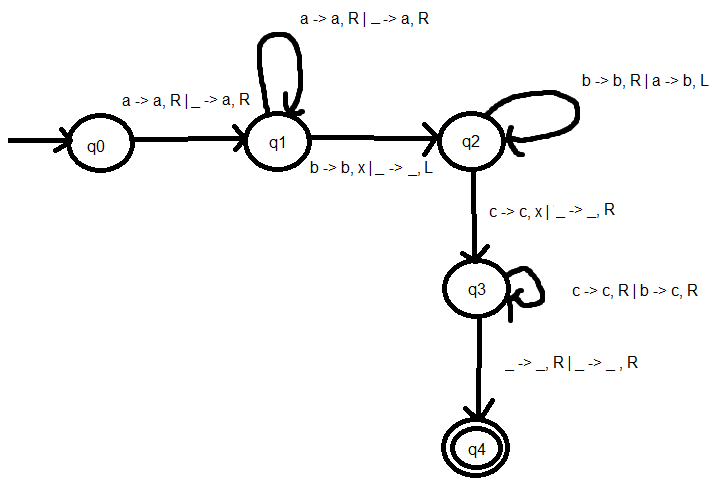


Q6. [15] For a two-tape Turing Machine,

1. [5] Give a *formal definition* of a transition function δ in two-tape TM.

δ (q0, (a, e)) -> ((q1, (x,y), (L, R)) – the transition function considers input symbols on both tapes.

1. [10] Construct a two-tape TM that accepts L = { *anbn cn* | *n* ≥ 1}



Q7.[15, optional] Construct a **Nondeterministic** TM (NTM) that accepts L ={ *wwRw* | *w* ∈ {*a, b*}+ }.

1. Draw its transition graph, (B) explain how your transitions work out and (C) how the nondeterministic simplifies the case.

Note that the middle of the string in wwR can be guessed in NTM.

Q8. [10] Give the encoding, using the suggested method in the slide of Chap.9-#25-#27, for

δ(*q1, a1*) = (*q1, a1*, R); δ(*q1, a2*) = (*q3, a1*, L); δ(*q3, a1*) = (*q2, a2*, L)

Encoding : a : 1, b : 11, R : 1, L : 11

**δ(*q1, a1*) = (*q1, a1*, R) ->** 01010101010

**δ(*q1, a2*) = (*q3, a1*, L) ->** 010110111010110

**δ(q3, a1) = (q2, a2, L) - >** 0111010110110110

Q9. [5] If *a* is encoded as 1, *b* as 11, R as 1, L as 11, decode the string 011010111011010.

String : 011010111011010 -> **δ(*q2, a*) = (*q3, b*, R);**

Q10. [10, optional] Describe an algorithm that examines a string in {0, 1}+ to determine whether or not it represents an encoded Turing Machine.

The algorithm will check to see if the first number in the string is 0. It will then check if 0 is followed by another zero. If it is, then the string will not be considered encoded. If the next number is a 1 then it keep checking how many ones are after. If the next number is a 0 then it will go back to the step where it checks if there are more than 1 zero between sections of 1’s. This process will continue until the end of the string is reached but if the number of sections is not equal to 5 then the string will not be considered encoded.

Q11. [10] Describe how Linear Bounded Automata could be constructed to accept

L = { *an* | *n* is a prime number}.

To construct a linear bounded automaton for an | n is a prime number:

1. Create a 4 strip LBA.
2. Strip 1 will hold the input string
3. Strip 2 will parse the input string for the number of a’s and return n into the third strip
4. Since 0 and 1 are not prime, reject a string if n is 0 or 1.
5. Else, take the n from step three and divide it by numbers 2 to n-1. If any number divides n then it will not be accepted.