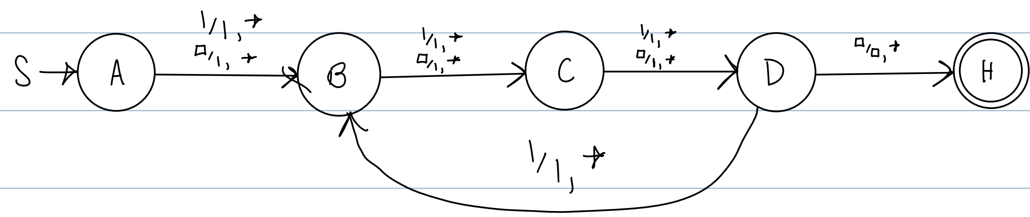
1. Implement a Turing Machine that rounds its unary input up to the next multiple of 3 by appending one or two 1s as needed. Assume that an empty tape represents the number zero (the symbol 0 is not used). For example, an input of 1, 11, or 111 would result in 111, an input of 1111 would result in 111111.



2. Finish the grammar we started in class to compute n div 2 for unary numbers. The rules we have so far are below. For example, to derive 5 div 2 we would begin with an input string of S11111S and eventually get the string 11 back. Similarly, to derive 6 div 2 we would begin with an input string of S111111S and get the string 111 as a result.

$S1 \to T$
$T1 \to 1S$

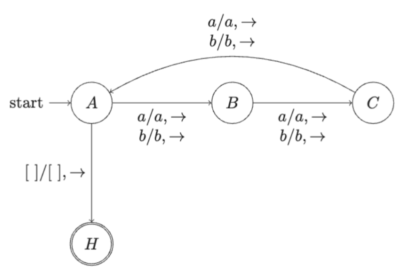
$S1 \to T$

$T1 \to 1S$

$TS \to \lambda$

$ST \to \lambda$

3. Convert the below Turing Machine to an unrestricted grammar:



1. $Aa \to aB$
2. $Ab \to bB$
3. $Ba \to aC$
4. $Bb \to bC$
5. $Ca \to aA$
6. $Cb \to bA$
7. $A\square \to \square H$
8. $\square H \to H$
9. $H\square \to H$
10. $H \to \lambda$

$\delta(A, a) = (B, a, \to)$

$\delta(A, b) = (B, b, \to)$

$\delta(B, a) = (C, c, \to)$

$\delta(B, b) = (C, c, \to)$

$\delta(C, a) = (A, a, \to)$

$\delta(C, b) = (A, b, \to)$

$\delta(A, \square) = (H, \square, \to)$

If $\leftarrow$, use:

$\_\, q_0 b \to q\, \_\, b$

If $\to$ use

$(B a) = (C, a, \to)$