# Lab 5: Turing Machines

CSCI 340: Computational Models

100 points

In this lab, you'll use JFLAP to design Turing Machines. The transitions in JFLAP again have three actions: the character being read, the character being written and the direction to move the tape head in (there is an "extra" option here compared to what we've been studying — the tape head can move left, right or stay — so L, R, or S).

Note that the blank character filling the tape on either end of the input (yes, the machines in JFLAP are 2-way infinite tapes) is □. You get this character on a transition by leaving the field blank. However, you can't seem to type this character in the input window, so it can't be part of the initial string you're testing.

Also note that the output is considered to be everything from what the tape head is currently pointing to until the end of the string. So please rewind the tape head to the beginning of the output at the end of each machine by looping until you are in the right position (this will often mean finding □ at the left).

Download the handout from AutoLab and do not change the filenames. Be sure to use JFLAP version 7! When you are ready to submit for testing, create a .zip file of the directory (it must still be called handout) and submit that .zip file to AutoLab. In all cases other than #4, the grading will be all or nothing for each problem, so be sure that you are thoroughly testing your machines!

1. [20pt] Construct a TM that concatenates two words divided by a B. The input alphabet is $\Sigma = \{ 1 \ 0 \ B \}$. So input of `1011B011` should produce `1011011`.

2. [20pt] Construct a TM that accepts all strings with more a's than b's (input alphabet is $\Sigma = \{ a \ b \}$). Leave the tape head positioned at the □ at the end of the output (no output expected).

3. [25pt] Construct a TM that accepts all strings of the form $a^n b^n a^n b^n$ where $\Sigma = \{ a \ b \}$. Leave the tape head positioned at the □ at the end of the output (no output expected).

4. [45pt] Construct a TM that adds two binary numbers together and leaves the answer on the TAPE in binary notation. The input will be of the form:

   $\#(0 + 1)^*\$(0 + 1)^*$

   You can consider this to be #*x-part*$*y-part*. You can change the *y-part* to represent the answer or you can introduce a new working area after the *y-part* to hold the answer. Remember to leave your tape head in position so that the answer shows up as the output of the machine. Be careful of the overflow condition where the answer has more bits than the largest number. (The insert subprogram from class will be helpful here).

   Note: For 20/35 points, your TM must handle input strings of the same length (your output may be different length if there's overflow). For 35/35 points, you need to handle cases where the input strings can be of different lengths. You can assume you always have at least one character for each input, though.