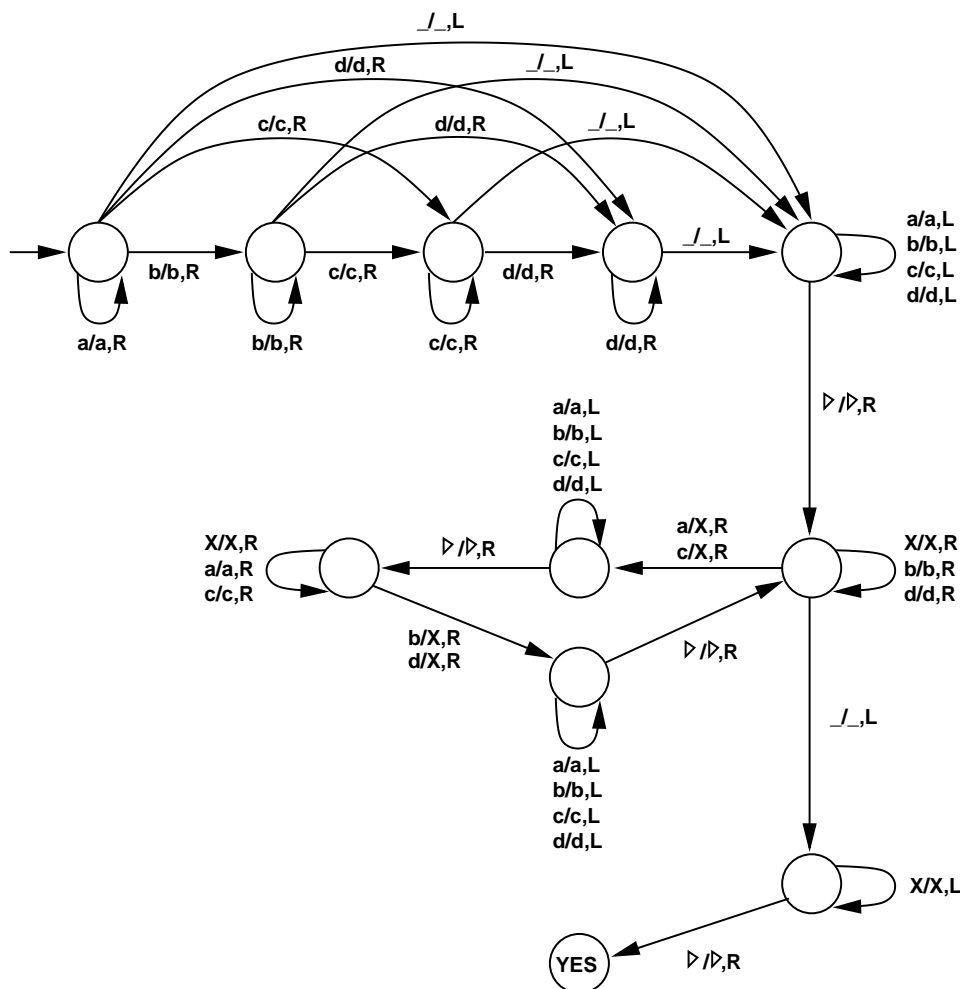


## COMP 610 Homework 2 Solution Sketches

- Construct a Turing Machine  $M$  that decides  $L = \{a^i b^j c^k d^\ell : i + k = j + \ell\}$  (ie  $M$  stops in state “YES” if the input is in  $L$  and “NO” if it isn’t). Explain how your Turing Machine works.

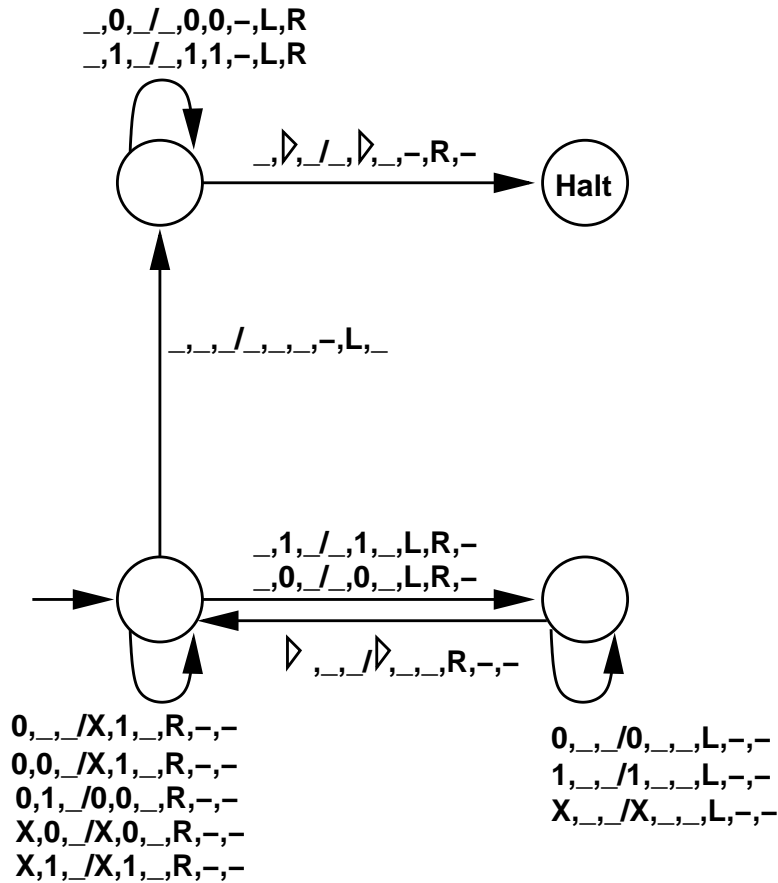


Any transition not shown in the diagram goes to the NO state. The top states verify that the string has the form  $a^i b^j c^k d^\ell$ . The next row of states alternates between 1) going through the string crossing out either a or c and 2) going through the string and crossing out b or d. The states at the bottom ensure that all characters have been crossed out.

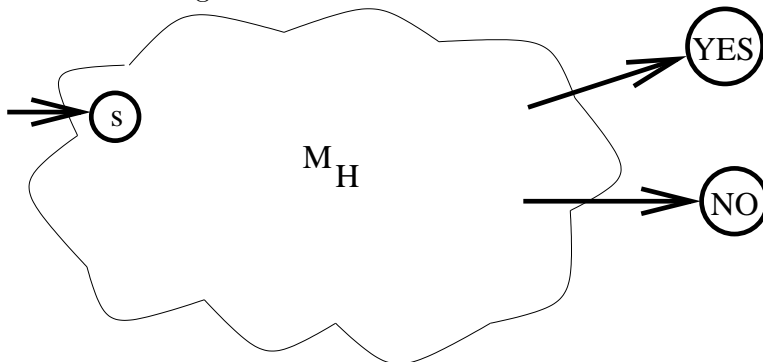
- Construct a  $k$ -tape Turing Machine  $M$  which starts with a number in unary on the first tape and ends with the same number in binary on the last tape. In other words, if  $M$  is started with a string of  $n$  0's on the first tape then  $M$  should halt with the number  $n$  in binary on the last tape. Explain how your Turing Machine works.

The diagram is a 3-tape Turing Machine. Any transition not shown in the diagram goes to the NO state. The loop in the start state crosses out every other “0” on the first tape and alternates the value on the second tape (basically dividing the number by 2 and finding out whether the value is even or odd. This should sound familiar if you know how to convert

from unary to binary, but results in the number written on the second tape backward. The state in the lower right position just returns the first read/write head to the beginning of the tape. The upper left state copies the value on the second tape to the third tape in reverse.



3. In the proof that HALTING is undecidable, the (assumed to exist) Turing Machine  $M_H$  is modified to a new Turing Machine  $D$ . Assume that  $M_H$  is represented by the image below, draw the Turing Machine  $D$ .



This answer is only a sketch. You should have actually drawn out the modifications to the Turing Machine.

The Turing Machine depicted should be modified by 1) adding states which replace the original string on the tape with two copies of the string on the tape in front of the state  $s$ ,

2) changing the YES state to a state which always returns to itself regardless of what is seen on the tape, and 3) changing the NO state to YES.

4. Let  $\text{REG} = \{M : L(M) \text{ is a regular language}\}$ . Prove that  $\text{REG}$  is not recursive by reducing  $\text{H}$  to  $\text{REG}$ . For those preferring the “question” version: Given a Turing Machine  $M$ , does  $M$  accept a regular language?

Given an instance of  $\text{HALTING}$   $(M; x)$  create an instance of  $\text{REG}$   $M'$  which on an input  $y$ : 1) checks if  $y = a^i b^i$  for some  $i$  and if it does moving to the YES state, 2) simulates  $M(x)$ , and 3) (if step 2 terminates) moves to the YES state.

Note that if  $M(x)$  halts then  $L(M') = \Sigma^*$ , but if  $M(x)$  doesn't halt then  $L(M') = \{a^i b^i\}$ . Since  $\Sigma^*$  is regular and  $\{a^i b^i\}$  is not regular,  $M'$  accepts a regular language iff  $M(x) \neq \nearrow$ .