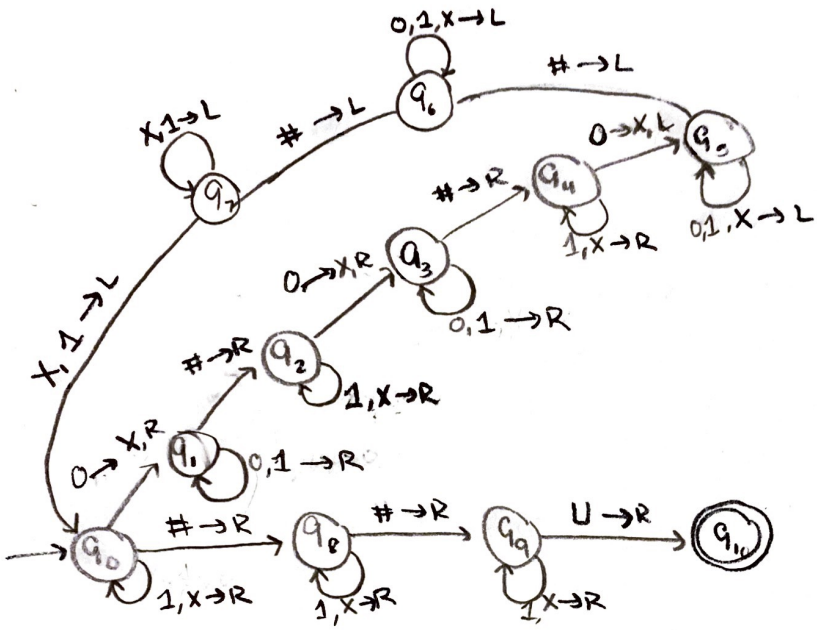
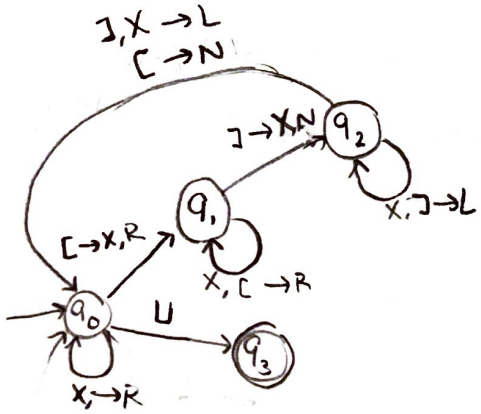
Jonathan Tso
HW 4
CS301

1. a) $L = \{a\#b\#c \mid a,b,c \in \{0,1\}^*$ and a,b,c, all same # of zeroes$\}$



b) $L = \{w \in \{[,]\}^* \mid w$ is a string of matched brackets$\}$

N = not left or right



$L = \{0,1\}$ $\Gamma = \{0,1\}$ and X is bin int > y}

1. Run through the tape to the right and mark all 0's as X's.
   Once you see a 1, go to the right to the #. If you don't, reject.
2. Run through the tape and mark as 0's as X's.
   If you see a 1, mark as X and go back to # and then back to the first X you see on the left.
   If you end up seeing empty char, accept.
3. Run right. If you see a 0, mark X. go to the right until the #. Then, if you see a 1, mark as X and go left to # and to the first X. Repeat this until you have no more 0/1's on the left or right side.

   case 1 →
   - If you run out of 0's/1's on left, reject
   - If you run out of 0's/1's on right, accept.
   - If you see a 0, mark as X, go left to # and then first X, and repeat 3.

   case 2 → If you see a 1, mark as X and go right to the #. Then, run right.
   - If you see a 1, mark as X. Go left to # and then first X, and repeat 3.
   - If you see a 0, mark as X. go left to # and then first X. Repeatedly run to right and left, marking 0's or 1's as X's until you have no more 0's or 1's.
     - If you have no more 0's or 1's on the left reject
     - If you have no more 0's or 1's on the right, accept.

This will halt once the characters on the right are all marked.

The idea is that if you see 1's on the left, a 1 on right will match. A 0 means you just need to make sure the right doesn't have more numbers

If you see a 0's on the left, a 0 will match on the right. A 1 m
the left must be a larger number

b) $L = \{u\#v \mid u \text{ is a substring of } v\}$

Let $\Sigma = \{0,1\}$

Let $\Gamma = \{0,1,A,B\}$

1. Run through the tape to right, until you hit that first 0 or 1.
   If 0, mark as A.
   If 1, mark as B.
2. Go to the right until you hit the #.
3. Go to the right, passing and marking characters until you see the same character that you changed. Turn that into an X.
4. Go left until you see an A or B, then, go right. Repeat step 1 and 2.
5. Go right and look for the first non X character. If it is the same, repeat from steps 1. If it is not the same character, go left until you hit the first A or B. If it's an A, turn it to a 0 and if it's a B, turn it into a 1. Go left until the empty character. Repeat from steps 1. If you find that all the characters on the left are A or B and you have a match on the right, accept. Else, reject.

This will halt once everything on the right has been marked X.

3. a) This machine only recognizes regular languages

Because we cannot write over the tape that contains, we are effectively unable to alter the input and can only read it. Additionally, reading left vs right provides no major benefit since we cannot alter the tape and so ~~only read~~ we can say we only read to the right, bc they do not 'retain' any previous information. This is also why it can't recognize CFL's. Because we cannot alter the string and are effectively reading the string in one direction, this behavior is similar to a DFA. Not only that, we know that if we can create a DFA, we can express the regular language corresponding to it. Bc of this we can say that this corresponds to recognizing regular languages.

b). CFL → V, Σ, R, S

Consider a turing machine M that decides a CFL. First, we know that CFL's can be converted to CNF, where then we can reliably say it will at some point halt.

Then, we know that we can convert this to a string where S is the starting variable. Additionally, all variables will have associated pathways via R rules. These, given by Σ, can be used to read any potential string<CNF,S> on the tape. As stated before, because we can halt, we know it will be decidable, given an accept and reject and thus, any CFL can be converted and decided by a turing machine.

4. We know that extended tape (more than 1 tape) is equivalent to a TM of a single tape (this was discussed in lecture).

We can consider a 2-D tape to be a stack of single tapes put together. Bc of this definition, they accept the same set of languages.