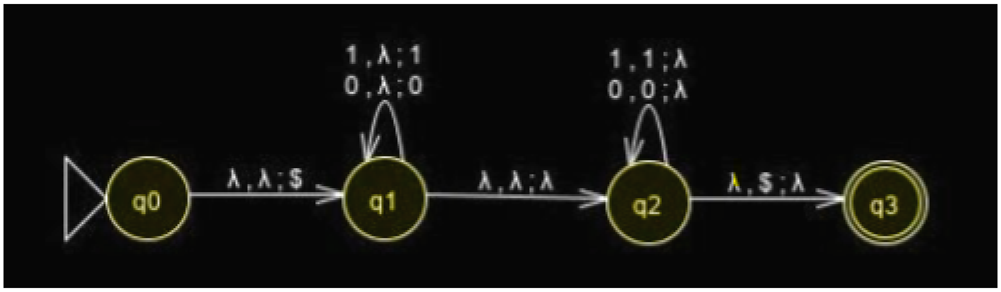
1. [5 pts] Given the following PDA state diagram, applied procedure we learned in class (Lemma 2.27,pp.140) to generate the following grammar rules:



1.1) Rules between q0 and q3, use A03 to represent the variable.

(A03) -> ε(A12)ε

1.2) Rules between q1 and q2, use A12 to represent the variable.

(A12) -> 0(A12)0
(A12) -> 1(A12)1

1.3) Rules between q0 and q2, use A02 to represent the variable.

(A02) -> (A01) (A12)
(A02) -> (A02) (A22) useless
(A22) -> ε useless

1.4) Rules between q1 and q3, use A13 to represent the variable.

(A13) -> (A12) (A23)
(A13) -> (A11) (A13) useless
(A11) -> ε useless

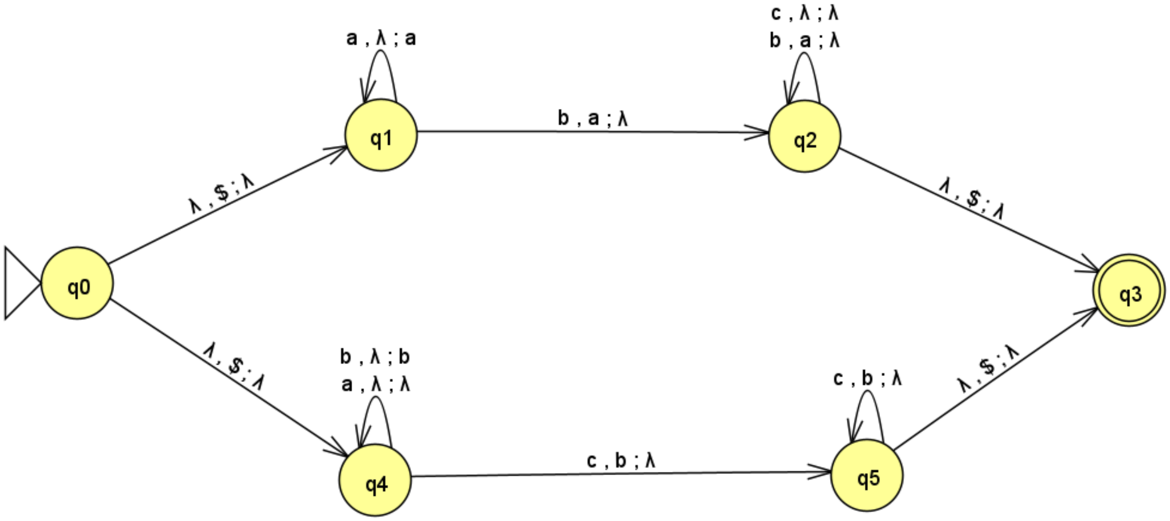
1.5) Rules between q1 and q1, use A11 to represent the variable.

(A11) -> ε

2. [6 pts] Decide whether the following languages are regular, context-free or not, if they are, draw the relevant DFA/NFA or PDA's state diagram; if not, use pumping lemma to prove it.

2.1) L = {a^i b^j c^k | $i = j$ or $j = k$}

The above language is not regular because no DFA/NFA/RE can be created. The above language L is context-free because we are able to create a Push Down Automata State diagram to represent it and if one is able to do that, then the language must be context-free.



2.2) L = {a^i b^j a^i b^j | i, j ≥ 0}

The above language is nonregular because a DFA/NFA/regular expression cannot be created. In order to determine if the language is context-free or not, use pumping lemma. We first assume that L is context-free, then according to the pumping lemma, there must exist a pumping length p, such that for any string w, w is in the set of L. w >=p, w can be written as w = u*v*x*y*z. It also satisfies the 3 conditions:

- **|vxy| <= p**
- **|vy| > 0 so v and y cannot both be empty**
- **A new string u*(v^i)*x*(y^i)*z is also in the language**

We can now pick a special string to perform our test cases. S = a^p b^p a^p b^p. S will be in the language as long as the number of a's in the first half is the same as in the second half and the same with the first and second half of b's. Before we run the cases we will split the string down the middle allowing us to refer to them as the first and second half of the string.

Case 1 & 2:

- **Case 1 is if v and y are only in the first half a's. In this case, picking any i > 1 would no longer make the a's in the first and second half equal which would make s not in the language. Case 2 is similar in that v and y are only in the first half's b's so the b's in the first and second half would not be equal which would make s no longer in the language.**

Case 3 & 4:

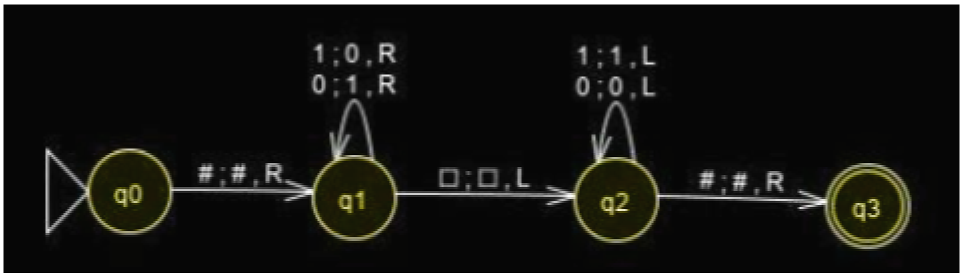
- **Case 3 is if v and y are partly in the first half's a's and b's. In this case, pumping v and y would increase the number of a's and number of b's by when i > 1. In this case the number of a's in the first half would not be the same as the second half and same with the number of b's. Case 4 is similar in that v and y would be partly in the a's and b's but in this case, it would be in the second half of s. This would lead to the same issues posed by case 3 but the second half of the string would be longer but it would still be outside the language**

Case 5:

- **Case 5 is if v and y are in the middle segment of the string where the v is in the first half b's and the y is in the second half a's. In this case, the number of a's and b's in both halves would not be the same and thus being outside the language.**

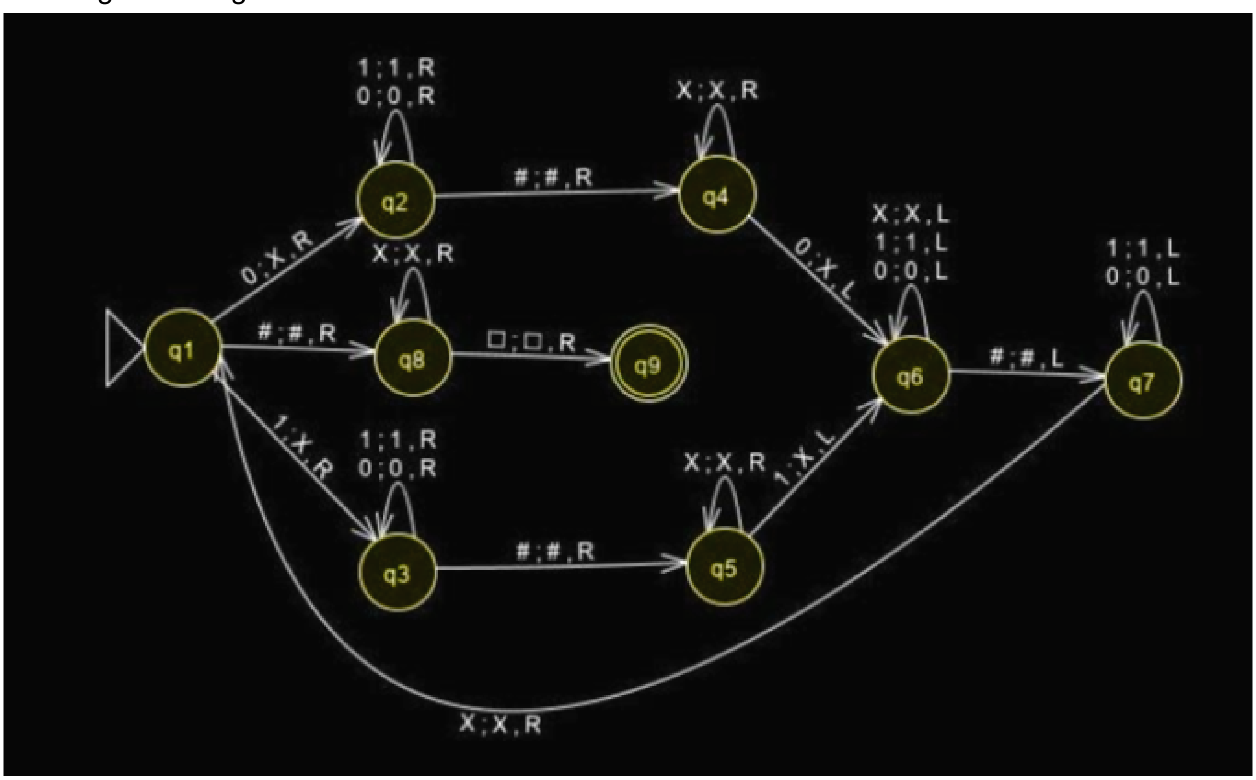
For any of these cases, vxy could not go past only 2 of the variables at a time because vxy can not be longer than p. Through the contradictions found in these cases, we find that the language L is not context free.

3. [3 pts] Assume there is a special symbol # placed at the left end of the Turing machine's tape (to mark the end of the type). Given the following TM's state diagram, explain in English what this TM does?



- **The Turing machine starts with the character # and then flips all the ones in the string to zeros and all the zeros in the string to ones**

4. [6 pts] Given the following TM state diagram, give the sequence of configurations for the following two strings:



4.1) 11

- **(q1)11**
- **X(q3)1**
- **X1(q3)□   NO Q3 w/ input □ transition**
- **REJECTED**

4.2) 01#01

- **(q1)01#01**
- **X(q2)1#01**
- **X1(q2)#01**
- **X1#(q4)01**
- **X1(q6)#X1**
- **X(q7)1#X1**
- **(q7)X1#X1**
- **X(q1)1#X1**
- **XX(q3)#X1**
- **XX#(q5)X1**
- **XX#X(q5)1**
- **XX#X(q5)1**
- **XX#(q6)XX**
- **XX(q6)#XX**
- **X(q7)X#XX**
- **XX(q1)#XX**
- **XX#(q8)XX**
- **XX#X(q8)X**
- **XX#XX(q8)□**
- **XX#XX□(q9)**
- **ACCEPTED**

4.3) Use set notation to describe the language accepted by this Turing machine.

TM = {w # w | w be any strings in the following alphabet (0,1)}