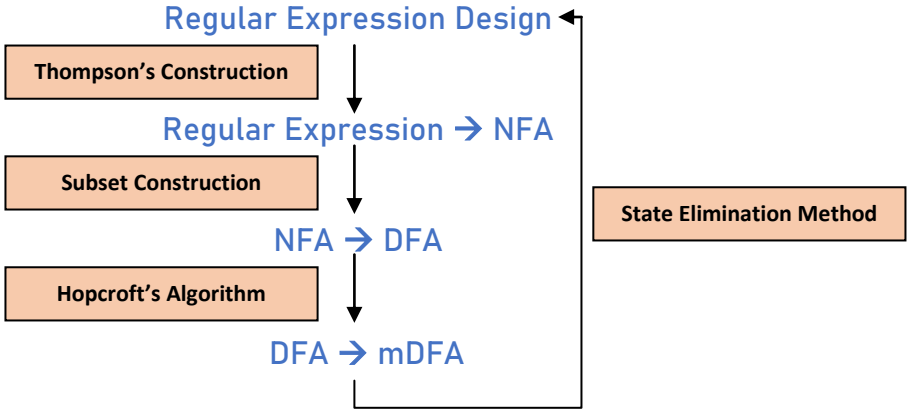
CSE331 Automata and Computability

Lecture 11.1: Conversion of DFA to Regular Expression

Presenter: Sabrina Zaman Ishita (SZI) *Scribe: Sabrina Zaman Ishita (SZI)*

In our previous lectures, we have shown you how to design different Regular Expressions, then we showed you how to convert regular expressions to NFA/ϵ-NFA through Thompson's Construction. Then you have been taught how to convert an NFA to DFA by subset-construction. We have further minimized this DFA by applying Hopcroft's Algorithm. Now, we will introduce you to the conversion mechanism of DFA to Regular Expression and we are going to use State Elimination Method to achieve this.



Following are some simple DFAs for which we can identify the Regular Expressions by simply looking at them:

But it is very difficult to figure out the Regular Expression of a DFA as given in Figure 1. For this we need follow algorithm. We are going to apply **State Elimination Method** to convert a DFA to its equivalent Regular Expression form.

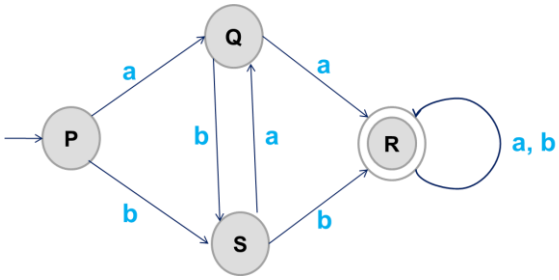
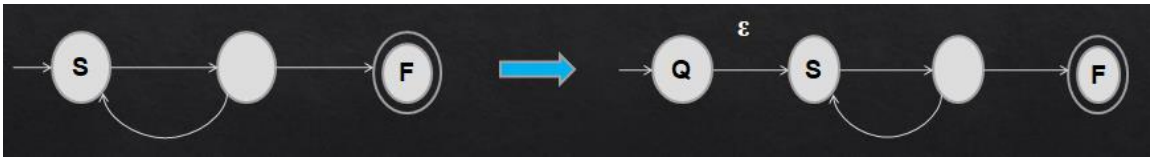


Figure 1:DFA

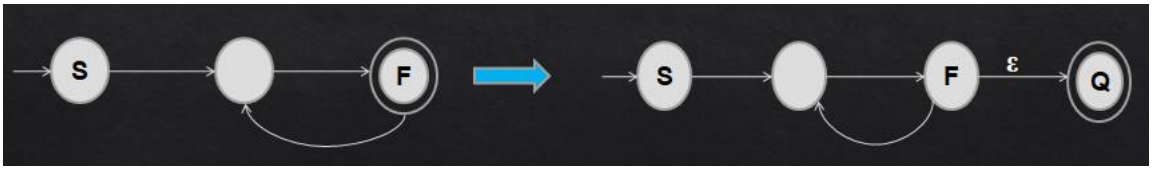
State Elimination Method:

Following rules are followed to apply this method:

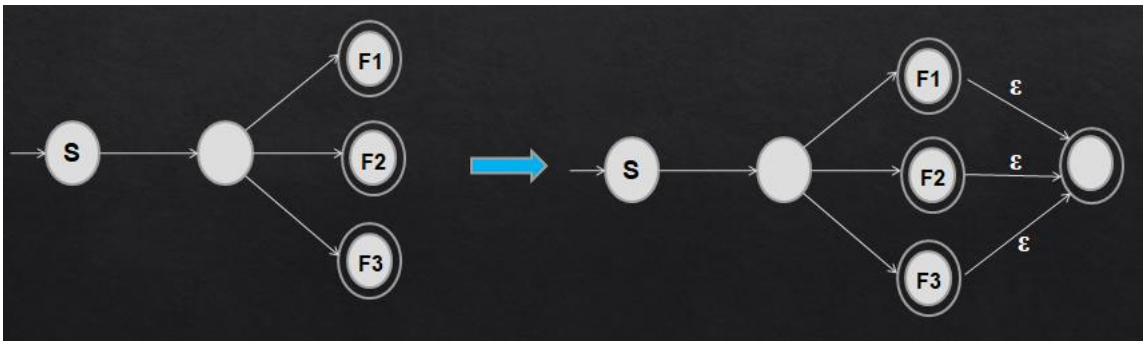
- If the *start state* has any incoming edge, **create new *start state* having no incoming edge** to it.



- If the *final state* has any outgoing edge, **create new *final state* having no outgoing edge** from it.



- If there are multiple *final state*s, **convert all the *final state*s into one single *final state***.

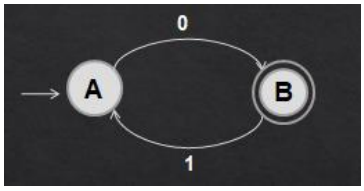


- Eliminate the intermediate states one by one (in any order) until the *start state* and *final state* exist.

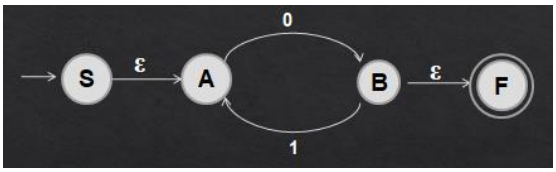
- ***Regular Expression*** = cost of transition between *start state* and *final state*.

Apply State Elimination Method on an Example:

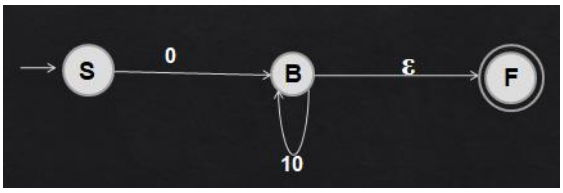
Example 1:



Let's add a new start state and a new final state as per the rules above.



Now, we need to eliminate state A and state B one by one in any order. Let's eliminate state A first and update the transitions which were through state A.



Now, let's eliminate state B which is very straightforward.

So, the equivalent Regular Expression for the given DFA is:

0 (1 0)*

If we had eliminated state B first, and the state A, we would have got different Regular Expression which is **(0 1)* 0**. But if you consider these regular expressions and try to figure out the strings they will be generating, you will get that they will be generating same set of strings.

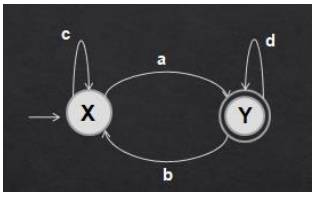
CSE331 Automata and Computability

Lecture 11.2: Conversion of DFA to Regular Expression

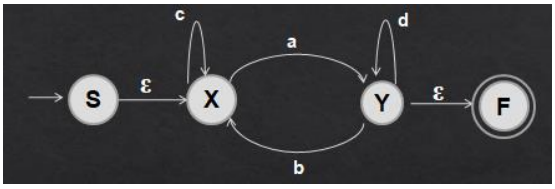
Presenter: Sabrina Zaman Ishita (SZI) *Scribe: Sabrina Zaman Ishita (SZI)*

This is a continuation note of our previous note on conversion of DFA to Regular Expression. Let's see some more examples on which we are going to apply State Elimination Method to convert DFA to Regular Expression.

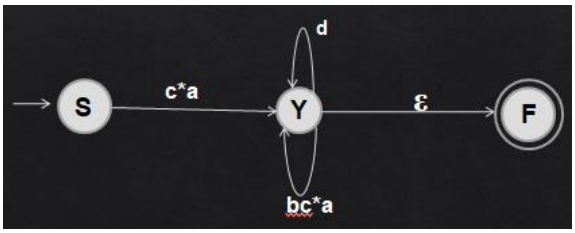
Example 2:



Let's add a new start state and a new final state as per the rules:



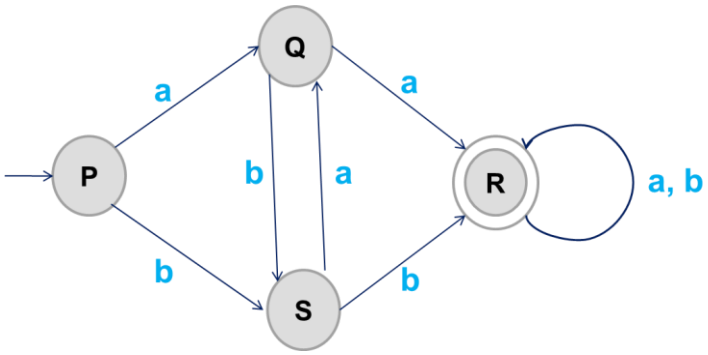
Now, let's eliminate state X first (you can eliminate state Y first and X later, doesn't matter)



Here, we can see that, the self loop to state X with 'c'-transition is considered and converted as c*. Now, we need to eliminate state Y.

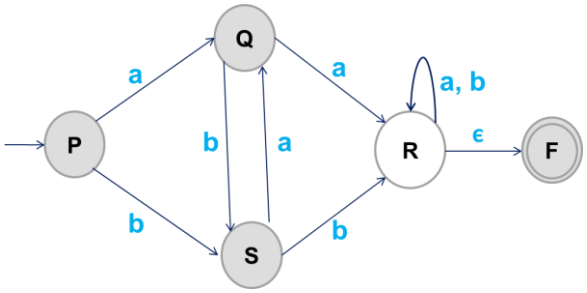
So, the final Regular Expression is: **c*a (d | bc*a)***

Example 3:

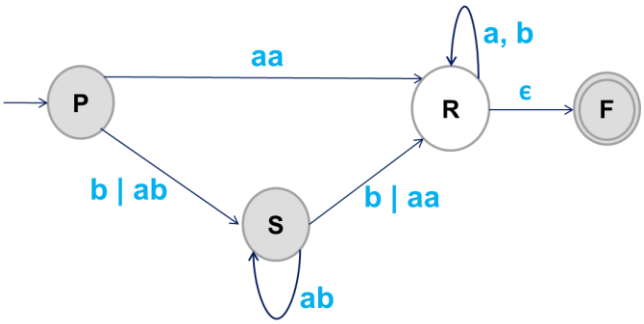


This is the very same example that we got stuck with earlier while figuring out the equivalent regular expression by simply looking at it. Let's convert this DFA to RE in step by step manner:

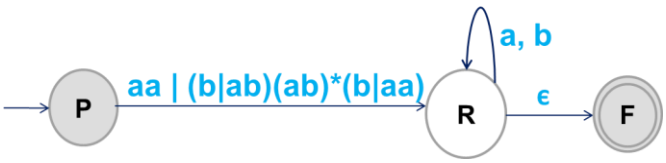
- Step 1: Add new final.



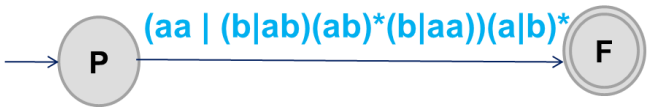
- Eliminate state Q (you can eliminate any state in any order). Update all the transitions which were connected with state Q. For each incoming transition to state Q, consider each outgoing transition from state Q. Following this, P→R (P->Q->R), P→S (P->Q->S), S→R (S->Q->R) and S→S (S->Q->S) transitions will be updated.



- Let's eliminate state S next.



- Now, eliminate the only intermediate state R:



So, the equivalent Regular Expression is:

(aa | (b|ab)(ab)*(b|aa)) (a|b)*