Assignment 2

Consider an alphabet made only of the input symbols a and b. Then consider

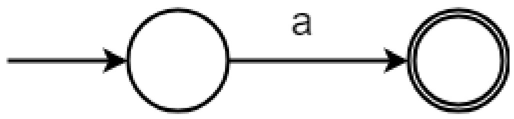
the following regular expression over this alphabet:

aba*(a|b)a

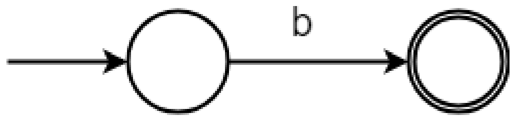
1) Use Thompson's Construction Method to construct an NFA that recognizes the strings that are in the language defined by the regular expression above. Explain step by step how you construct the NFA.

Using the Thompson's Construction Method, a NFA can be constructed from specific regular expression. Given the regular expression r: aba*(a|b)a, N(r) can be constructed for the NFA that accepts the language L(r).

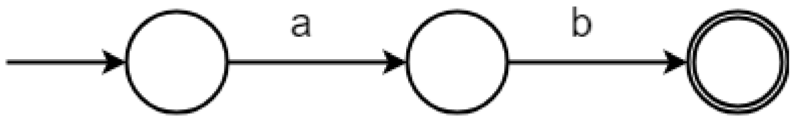
1. constructing the NFA **N(a)** for the regular expression **a**:



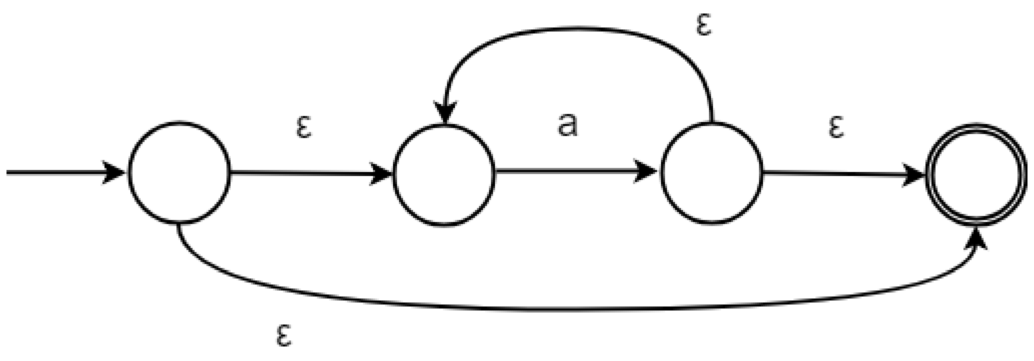
2. and the NFA **N(b)** for the regular expression **b**:



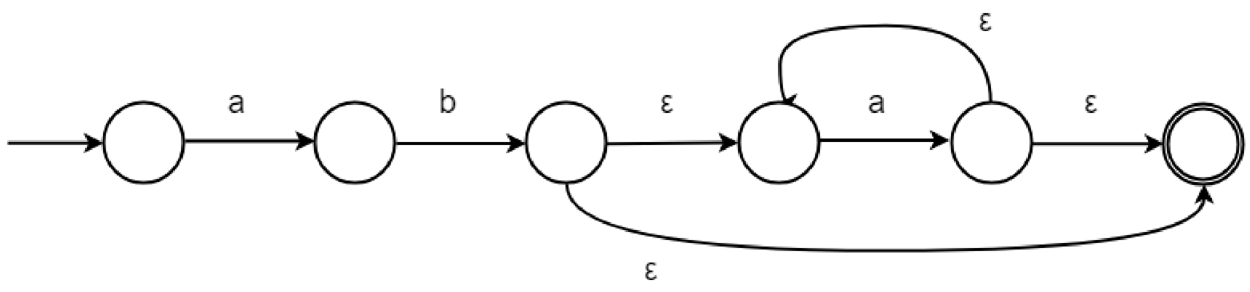
3. combine using the rule for concatenation to get the NFA **N(ab)**:



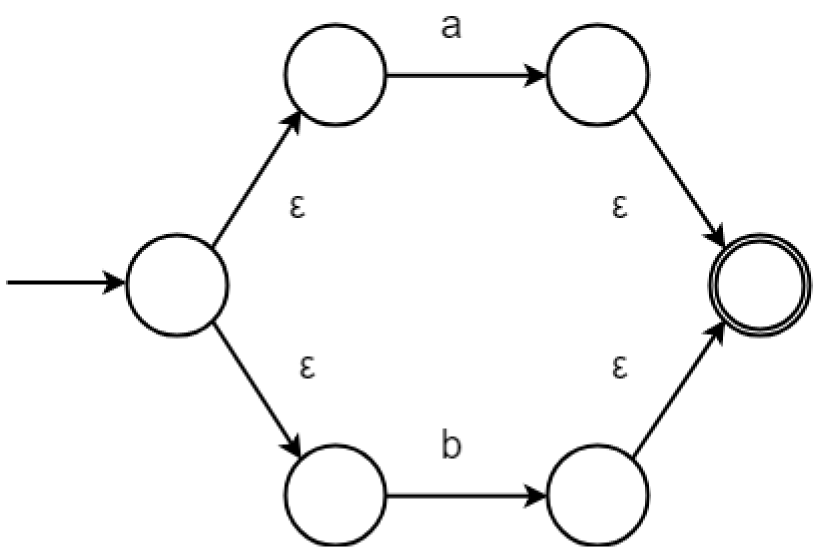
4. construct the NFA **N(a*)** using the construction rule for the Kleene star:



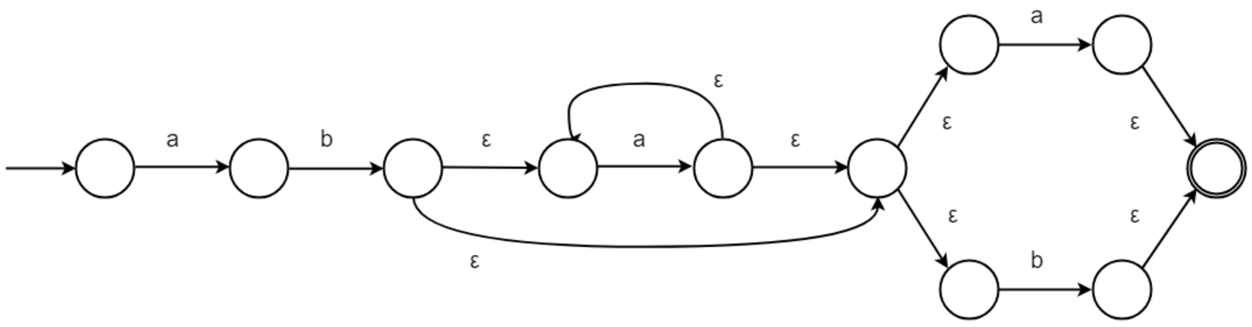
5. and again use the concatenation rule to combine it with the previous NFA to get **N(aba*)**:



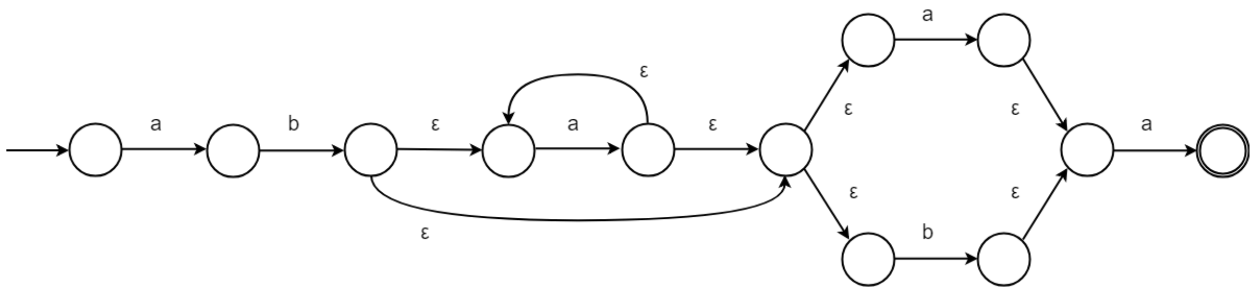
6. combine NFAs of N(a) and N(b) together using the rule for "|" to get **N(a|b)**:



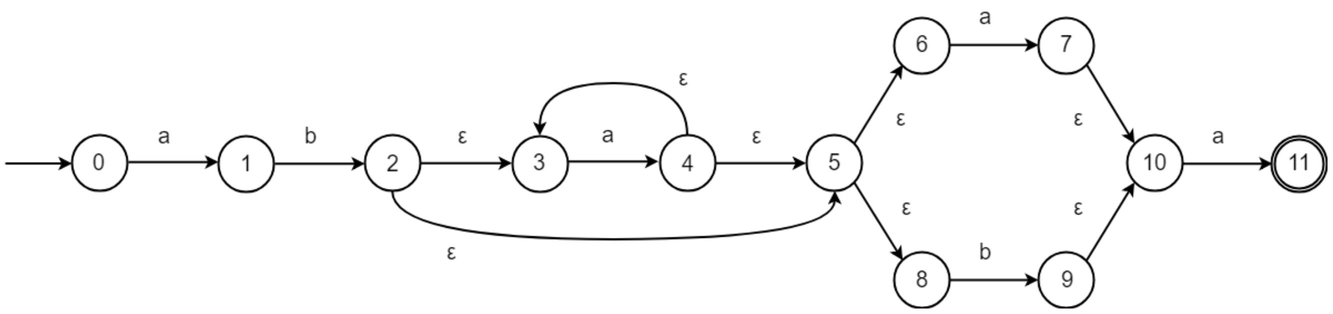
7. use the concatenation rule to combine it with the previous NFA to get **N(aba*(a|b))**:



8. again use the concatenation rule to combine it with N(a) to get **N(aba*(a|b)a)**:



2) Number the states of the NFA with numbers 0, 1, 2, etc., using a top-down first and then left-right order (start from the left, moving towards the right; when two states are one above the other, number the top state first, then the bottom state, then the other remaining states on the right using again the same top-down first and then left-right order).



Then explain step by step what happens when the NFA is given the following input string: **abb**

The NFA starts in its start state 0 and reads input symbols one by one when given the input string **abb**.

1. The NFA reads the first input symbol **a**, and it can move from state 0 to state 1 through the edge labeled with **a**.
2. Then the NFA reads the next input symbol **b**, and then it can move from state 1 to state 2 through the edge labeled with **b**.
3. At this phase the NFA may (or may not) move to state 3 or to state 5 through the ε-transitions without consuming any input. Therefore the NFA can be in possible states either be state 2, state 3, or state 5. However, with the next input **b** read, the NFA would be stuck if it is at state 2 or state 3. Because there is no transition out of state 2 and state 3 labeled with **b**, while at state 5 there are still ε-transitions out to other states. Thus, the NFA can move from state 2 to state 5 through the ε-transition. And continually, the NFA would be stuck at state 6 as here is no transition out of state 2 and state 3 labeled with **b**, but only **a**. Therefore the NFA can then move from state 5 to state 8 through the ε-transition, and then move from state 8 to state 9 through the edge labeled with **b**.
4. Now the NFA can stay at state 9 or move to state 10 through the ε-transition. But from neither state 9 nor state 10 can the NFA reach the final state (state 11) without one more input, specifically **a**, which is the label of the edge from state 10 to state 11. Therefore the NFA has to reject the whole input string. Since the string **aab** is not in the set of strings represented by the regular expression **aba*(a|b)a**, which the NFA is for, the result that the input been rejected is expected and correct.

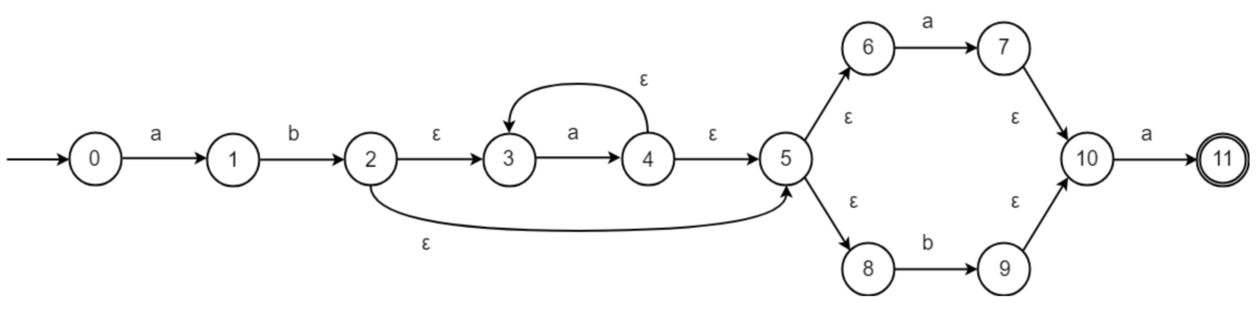
Then explain step by step what happens when the NFA is given the following input string: **abaa**

The NFA starts in its start state 0 and reads input symbols one by one when given the input string **abaa**.

1. The NFA reads the first input symbol **a**, and it can move from state 0 to state 1 through the edge labeled with **a**.
2. Then the NFA reads the next input symbol **b**, and then it can move from state 1 to state 2 through the edge labeled with **b**.
3. At this phase the NFA may (or may not) move to state 3 or to state 5 through the ε-transitions without consuming any input. Therefore the NFA can be in possible states either be state 2, state 3, or state 5, from which the NFA can then move to state 6 or to state 8. However, with the next input **a** read, the NFA would be stuck if it is at state 8. Because there is no transition out of state 8 labeled with **a**, while at state 3 and state 6 there are transition out labeled with **a**. Thus, the NFA can move from state 2 to state 3 through the ε-transition and then move from state 3 to state 4 through the edge labeled with **a**, or to state 5 through the ε-transition, and then, the NFA can move from state 5 to state 6 through the ε-transition, and then move from state 6 to state 7 through the edge labeled with **a**. Therefore, now the NFA has two possible states: state 4 or state 7.
4. Then the NFA reads the next input symbol **a**. If the NFA is in state 4, as steps above, it still has to move from state 4 to state 5, then to state 6 through ε-transitions, in which the NFA can move to state 7 through the edge with label same as input **a**. Or the NFA can loop back from state 4 to state 3 through the ε-transition and once again move to state 4 through edge labeled **a**. If the NFA move from state 5 to state 8, it will stuck there because the input is **a**, while there is no transition out with label **a**. However, in both these cases, the NFA cannot reach the final state 11, therefore it is not possible. The only possible case is that the NFA can move from state 7 to state 10 through the ε-transition, and then move from state 10 to state 11 through the edge labeled with **a**. There the NFA reaches the final state, with the whole input string has been consumed, which means the input string is accepted by the NFA. This result is expected and correct since the input string **abaa** is in the set of strings represented by the regular expression **aba*(a|b)a**.

3) Use the Subset Construction Algorithm to convert the NFA into a DFA. Explain each step of the construction, including the computation of epsilon-closures.

1. Starting with the NFA created above using Thompson's construction method for the regular expression **aba*(a|b)a**, with states numbered:

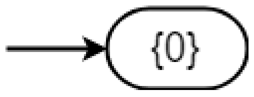


2. In order to apply the Subset Construction Algorithm, the ε-closure for states is needed to be computed. Therefore the table for the ε-closure function for this NFA is computed as:

NFA State s	ε-closure(s)
0	{0}
1	{1}
2	{2, 3, 5, 6 ,8}
3	{3}
4	{3, 4, 5, 6 ,8}
5	{5, 6, 8}
6	{6}
7	{7, 10}
8	{8}
9	{9, 10}
10	{10}
11	{11}

3. Basing on the table above, the corresponding DFA can be constructed using the subset construction algorithm.

   The start state of the DFA, according to the ε-closure(0), is:

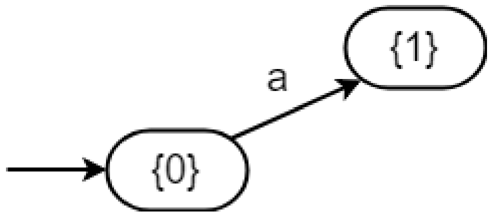


4. Then consider the possible transitions from this state on input symbol **a**:

NFA State s	moveN(s, a)
$0 \xrightarrow{a}$	{1}

NFA State s	moveN(s, a)
moveN({0}, a)	{1}
ε-closure(moveN({0}, a))	{1}

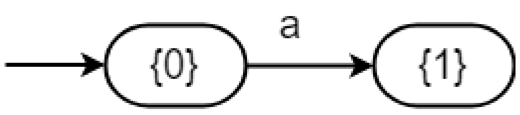
So add a new state to the DFA and a new transition on **a**:



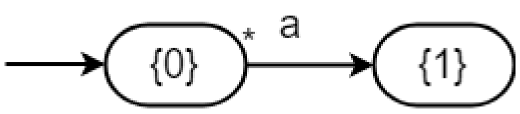
Similarly, consider the possible transitions from this state on input symbol **b**:

NFA State s	moveN(s, b)
$0 \xrightarrow{b}$	∅
moveN({0}, b)	∅
ε-closure(moveN({0}, b))	∅

The set just computed is empty, and there is no new state and no new transition is added to the DFA:



And the start state has now been completely processed:



5. Now process the second DFA state on input **a**:

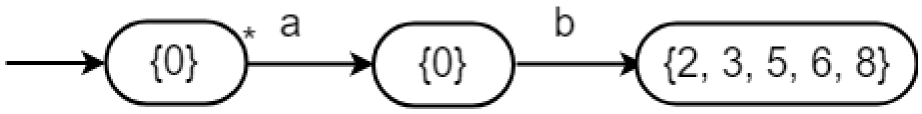
NFA State s	moveN(s, a)
$1 \xrightarrow{a}$	∅
moveN({1}, a)	∅
ε-closure(moveN({1}, a))	∅

The set just computed is empty, and there is no new state and no new transition is added to the DFA.

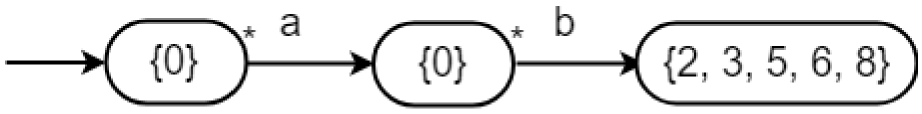
Process the second state on input symbol **b**:

NFA State s	**moveN(s, b)**
$1 \xrightarrow{b}$	{2}
moveN({1}, b)	{2}
ε-closure(moveN({1}, b))	{2, 3, 5, 6, 8}

So add a new state to the DFA and a new transition on **b**:



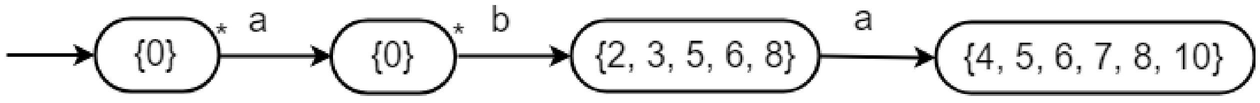
And the second state has now been completely processed:



6. Next process the third DFA state on input **a**:

NFA State s	**moveN(s, a)**
$2 \xrightarrow{a}$	∅
$3 \xrightarrow{a}$	{4}
$5 \xrightarrow{a}$	∅
$6 \xrightarrow{a}$	{7}
$8 \xrightarrow{a}$	∅
moveN({2, 3, 5, 6, 8}, a)	{4, 7}
ε-closure(moveN({2, 3, 5, 6, 8}, a))	{4, 5, 6, 7, 8, 10}

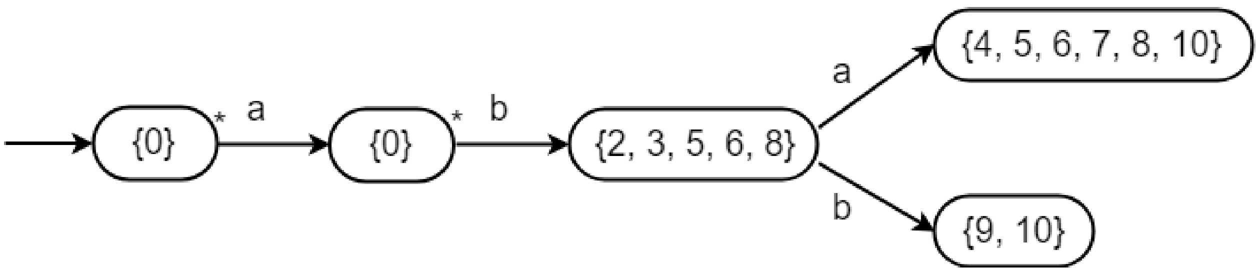
So add a new state to the DFA and a new transition on **a**:



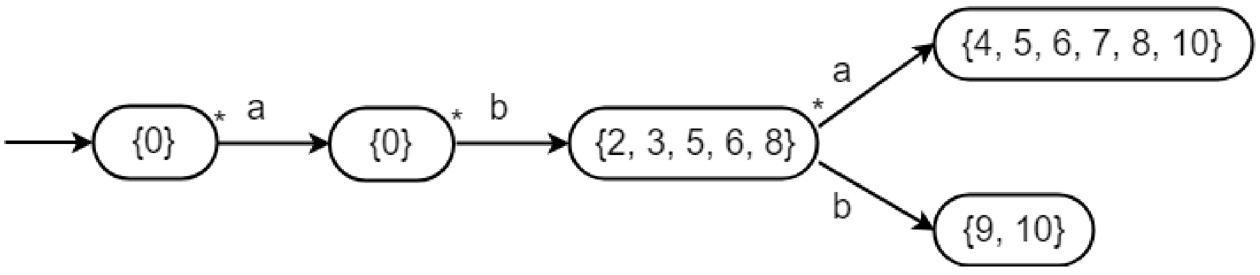
Process the third state on input symbol **b**:

NFA State s	moveN(s, b)
$2 \xrightarrow{b}$	∅
$3 \xrightarrow{b}$	∅
$5 \xrightarrow{b}$	∅
$6 \xrightarrow{b}$	∅
$8 \xrightarrow{b}$	{9}
moveN({2, 3, 5, 6, 8}, b)	{9}
ε-closure(moveN({2, 3, 5, 6, 8}, b))	{9, 10}

So add a new state to the DFA and a new transition on **b**:



And the third state has now been completely processed:

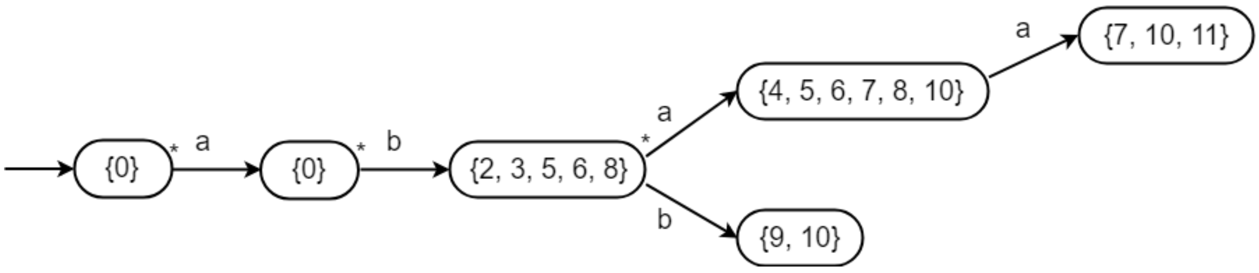


7. Next process the fourth DFA state on input **a**:

NFA State s	moveN(s, a)

NFA State s	moveN(s, a)
$4 \xrightarrow{a}$	∅
$5 \xrightarrow{a}$	∅
$6 \xrightarrow{a}$	{7}
$7 \xrightarrow{a}$	∅
$8 \xrightarrow{a}$	∅
$10 \xrightarrow{a}$	{11}
moveN({4, 5, 6, 7, 8, 10}, a)	{7, 11}
ε-closure(moveN({4, 5, 6, 7, 8, 10}, a))	{7, 10, 11}

So add a new state to the DFA and a new transition on **a**:

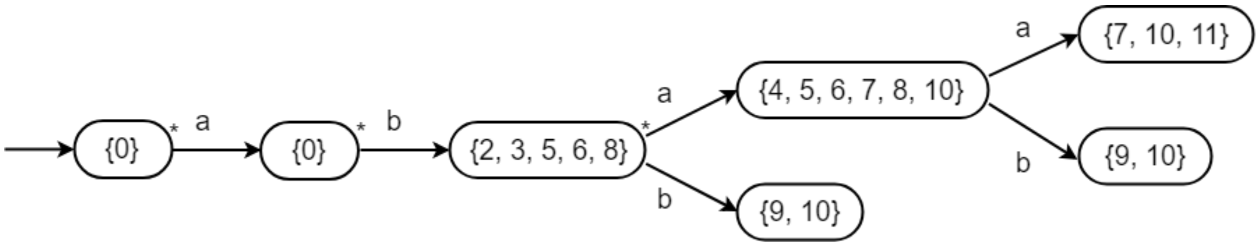


Process the third state on input symbol **b**:

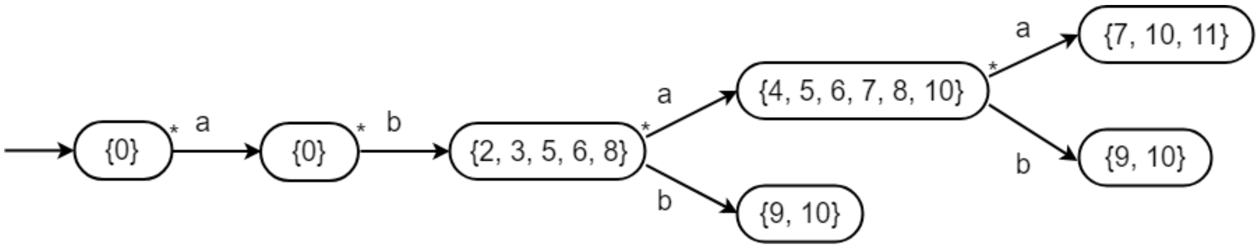
NFA State s	moveN(s, b)
$4 \xrightarrow{b}$	∅
$5 \xrightarrow{b}$	∅
$6 \xrightarrow{b}$	∅
$7 \xrightarrow{b}$	∅

NFA State s	moveN(s, b)
$8 \xrightarrow{b}$	{9}
$10 \xrightarrow{b}$	∅
moveN({4, 5, 6, 7, 8, 10}, b)	{9}
ε-closure(moveN({4, 5, 6, 7, 8, 10}, b))	{9, 10}

So add a new state to the DFA and a new transition on **b**:



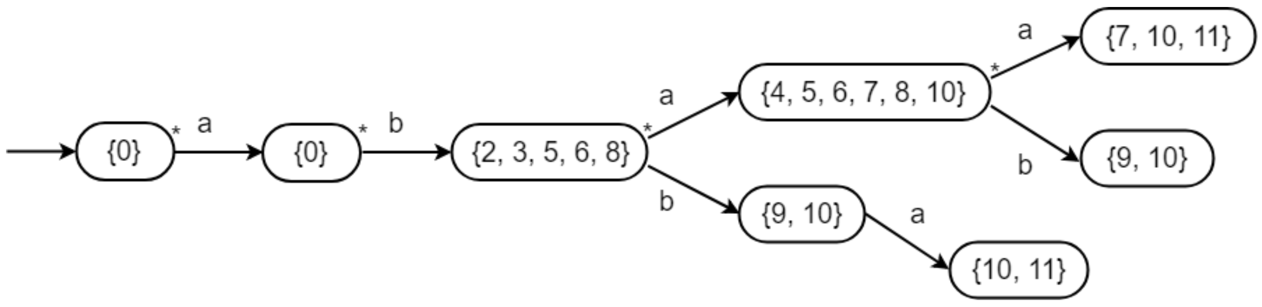
And the fourth state has now been completely processed:



8. Similarly, process the fifth state of the DFA on input **a**:

NFA State s	moveN(s, a)
$10 \xrightarrow{a}$	{11}
moveN({10}, a)	{11}
ε-closure(moveN({10}, a))	{10, 11}

So add a new state to the DFA and a new transition on **a**:

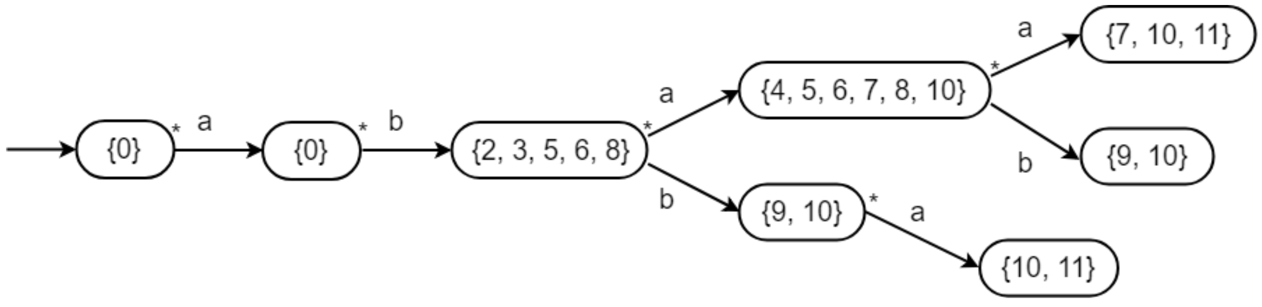


Process the fifth state on input symbol **b**:

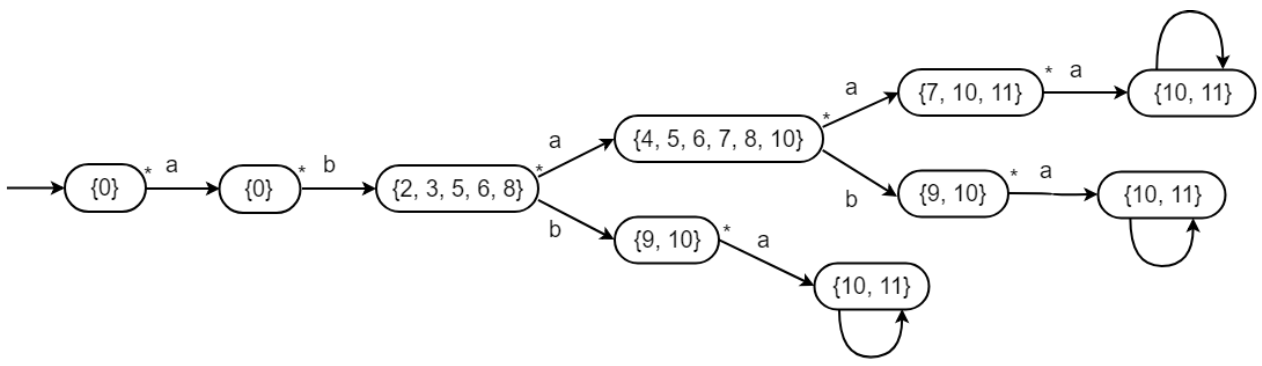
NFA State s	moveN(s, b)
$10 \xrightarrow{b}$	∅
moveN({10}, b)	∅
ε-closure(moveN({10}, b))	∅

The set just computed is empty, and there is no new state and no new transition is added to the DFA.

And the fifth state has now been completely processed:

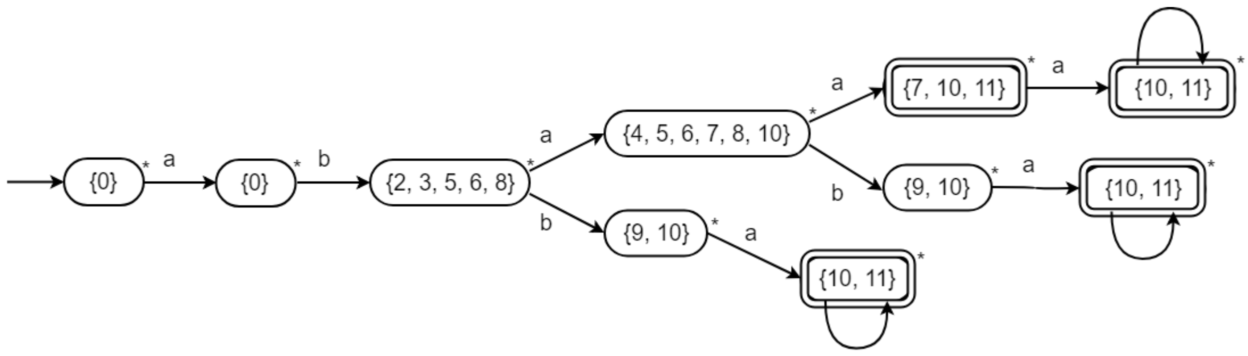


9. Similarly we process the other states of the DFA until we get:

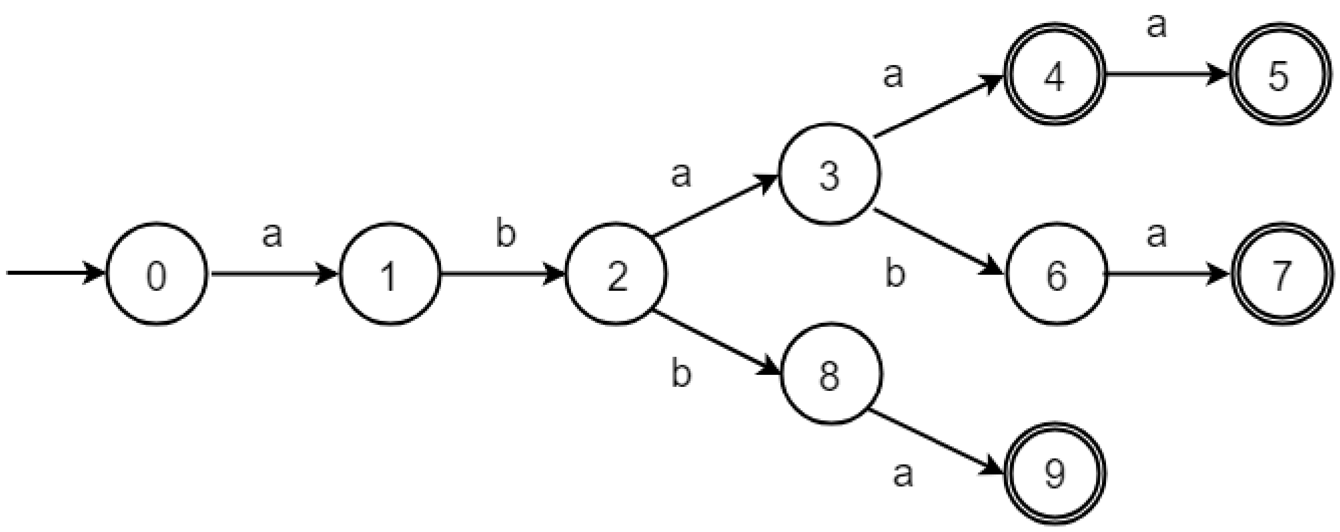


10. All the states in the DFA have been processes and step 2 of the subset construction algorithm is over.

The third and last step of the subset construction algorithm is to compute the final states of the DFA. A state of the DFA is a final state if and only if it contains a final state of the NFA. The NFA has only one final state, the state 11. All four states of the DFA that contain NFA state 11 are therefore marked as final states of the DFA:



4) Take the DFA you constructed in Question 3 and number the states with numbers 0, 1, 2, etc., using a top-down first and then left-right order.



Then write the transition table for the DFA.

	a	**b**
→ 0	{1}	∅
1	∅	{2}
2	{3}	{8}
3	{4}	{6}
* 4	{5}	∅

	a	**b**
* 5	∅	∅
6	{7}	∅
* 7	∅	∅
8	{9}	∅
* 9	∅	∅

Then explain step by step what happens when the DFA is given the following input string: **abb**

1. The DFA starts in state 0.
2. The DFA reads the first input symbol **a**. Then the DFA moves from state 0 to state 1.
3. The DFA reads the next input symbol **b**. Then the DFA moves from state 1 to state 2.
4. The DFA reads the next input symbol **b**. Then the DFA moves from state 2 to state 8.
5. The whole input string has been consumed by the DFA, but the DFA is in state 8, which is not a final state. Therefore the input string **abb** is rejected.

Then explain step by step what happens when the DFA is given the following input string: **abaa**

1. The DFA starts in state 0.
2. The DFA reads the first input symbol **a**. Then the DFA moves from state 0 to state 1.
3. The DFA reads the next input symbol **b**. Then the DFA moves from state 1 to state 2.
4. The DFA reads the next input symbol **a**. Then the DFA moves from state 2 to state 3.
5. The DFA reads the next input symbol **a**. Then the DFA moves from state 3 to state 4.
6. The whole input string has been consumed by the DFA, the DFA is in state 8, which is a final state. Therefore the input string **abb** is accepted.

Compare these last two results with the results of Question 2.

1. In both the NFA and the DFA, the input string **abb** is rejected for the same reason: the whole input string has been consumed while the NFA/DFA has not reached a final state.

2. In both the NFA and the DFA, the input string **abaa** is accepted for the same reason: the whole input string has been consumed and the NFA/DFA reaches a final state.

The DFA's transition process of states is clearer than the NFA's.