CS 208 : Automata Theory and Logic 15/02/2024

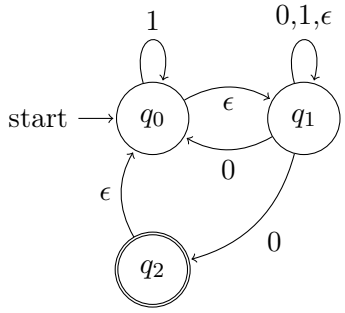
Lecture - 17

Topic: Eliminating ϵ transitions and Text Search using ϵ-NFA

Scribed by: Bhavya Sri Kottana (22B0981)
Checked and compiled by:

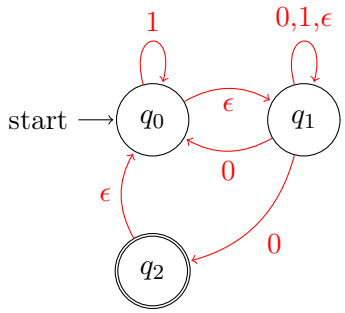
1 Eliminating ϵ-transitions

Let's try to understand this by an example,
Consider the string, $\epsilon\,1\,\varepsilon\,0\,0\,\epsilon\,\epsilon\,0$ and one of the paths followed by it is ,
$q_0(start) \xrightarrow{\epsilon} q_1 \xrightarrow{1} q_1 \xrightarrow{\epsilon} q_1 \xrightarrow{0} q_1 \xrightarrow{0} q_2 \xrightarrow{\epsilon} q_0 \xrightarrow{\epsilon} q_1 \xrightarrow{0} q_2$(accepting) in ϵ-NFA below



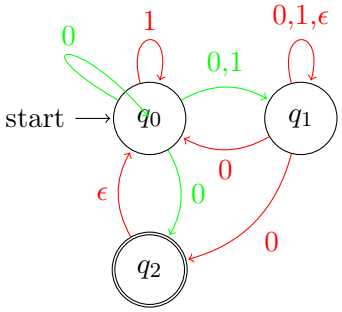
But given input strings will not contain ϵ bits. On removing these ϵ bits the resultant bit string is said to be accepted by ϵ-NFA So now to find an equivalent NFA without ϵ such that it accepts same language as ϵ-NFA,
Consider the first two bits in the example string(i.e, $\epsilon\,1$),but since the input string will not contain ϵ bits , so on input string 1, we are able to reach q_1 state from q_0 state, but on simply removing the ϵ edges we could see that we can't reach q_1 from q_0 , hence we need to add new edge from q_0 to q_1 on input bit 1.
From this we could see that,if we are at state q then on input bit x, we could reach all the states that can be reached from the states in ECLOSE(q) on bit x.(This is because we could reach all the ECLOSE(q) states by ϵ edges). So if we remove ϵ edges then we have to add new edges on each bit x from q to all the states which can be reached from ECLOSE(q) on bit x
The above discussed conclusion is not complete yet because, now we ensured that we could reach all the states on bit x that could be reached by taking ϵ edge followed by x bit edge in the original ϵ-NFA. What if our string ended with an ϵ? Then according to the above conclusion we couldn't reach the same state as before after removing ϵ edge. But since we are only concerned to have the same accepting language as before, we would consider every state q , whose ECLOSE(q) contains accepting state as accepting state in the newly constructed NFA.The initial states remains same

Let us construct NFA by eliminating ϵ edges from the above taken example, the ϵ-closure of each state is,
ECLOSE(q_0) = $\{q_0, q_1\}$
ECLOSE(q_1) = $\{q_1\}$
ECLOSE(q_2) = $\{q_0\ , q_1, q_2\}$

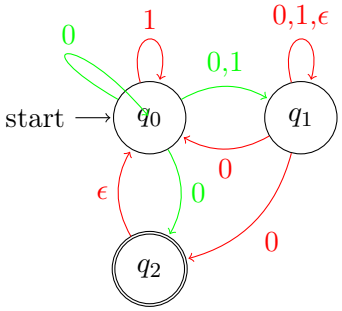


The ϵ-NFA

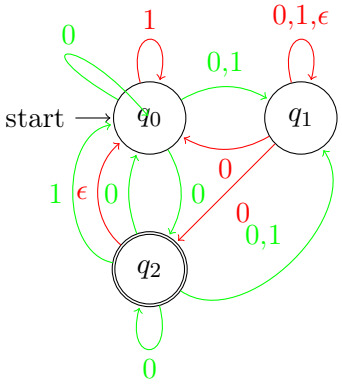
Consider state q_0
ECLOSE(q_0) = $\{q_0, q_1\}$
In the given ϵ-NFA,
$q_0 \xrightarrow{1} \{q_0\}, q_1 \xrightarrow{1} \{q_1\}$
$q_1 \xrightarrow{0} \{q_0, q_1, q_2\}$
so, in the new constructed NFA we should have,
$q_0 \xrightarrow{1} \{q_0, q_1\}$, $q_0 \xrightarrow{0} \{q_0, q_1, q_2\}$



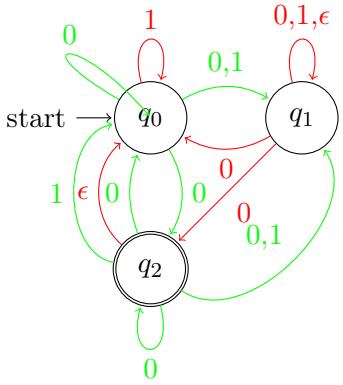
Consider state q_1
ECLOSE(q_1) = $\{ q_1\}$
So , no new edges to add



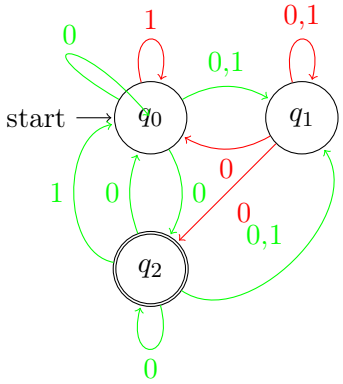
Consider state q_2
ECLOSE(q_2) = $\{ q_0, q_1, q_2\}$
In the given ϵ-NFA,
$q_0 \xrightarrow{1} \{q_0\}, q_1 \xrightarrow{1} \{q_1\}$
$q_1 \xrightarrow{0} \{q_0, q_1, q_2\}$
so, in the new constructed NFA we should have,
$q_2 \xrightarrow{1} \{q_0, q_1\}$, $q_2 \xrightarrow{0} \{q_0, q_1, q_2\}$



Since the accepting state q_2 is present only in ECLOSE(q_2), in the new NFA, the accepting state is q_2 only



Now remove the ϵ edges to get the new NFA which accepts the same language as the orignal ϵ-NFA



Summary:
Step 1: write ECLOSE(q) for all states q
Step 2: For every state q , find the union of all the states that can be reached from ECLOSE(q) states on each bit x and add the edges from q to those states in the union set if they are not present
Step 3: Mark all the states whose ϵ-closure contains accepting state as accepting states
Step 4 :remove all the ϵ edges

Hence an equivalent NFA without ϵ can be constructed from ϵ-NFA . Also we have seen that an equivalent DFA can be constructed from NFA in previous classes.

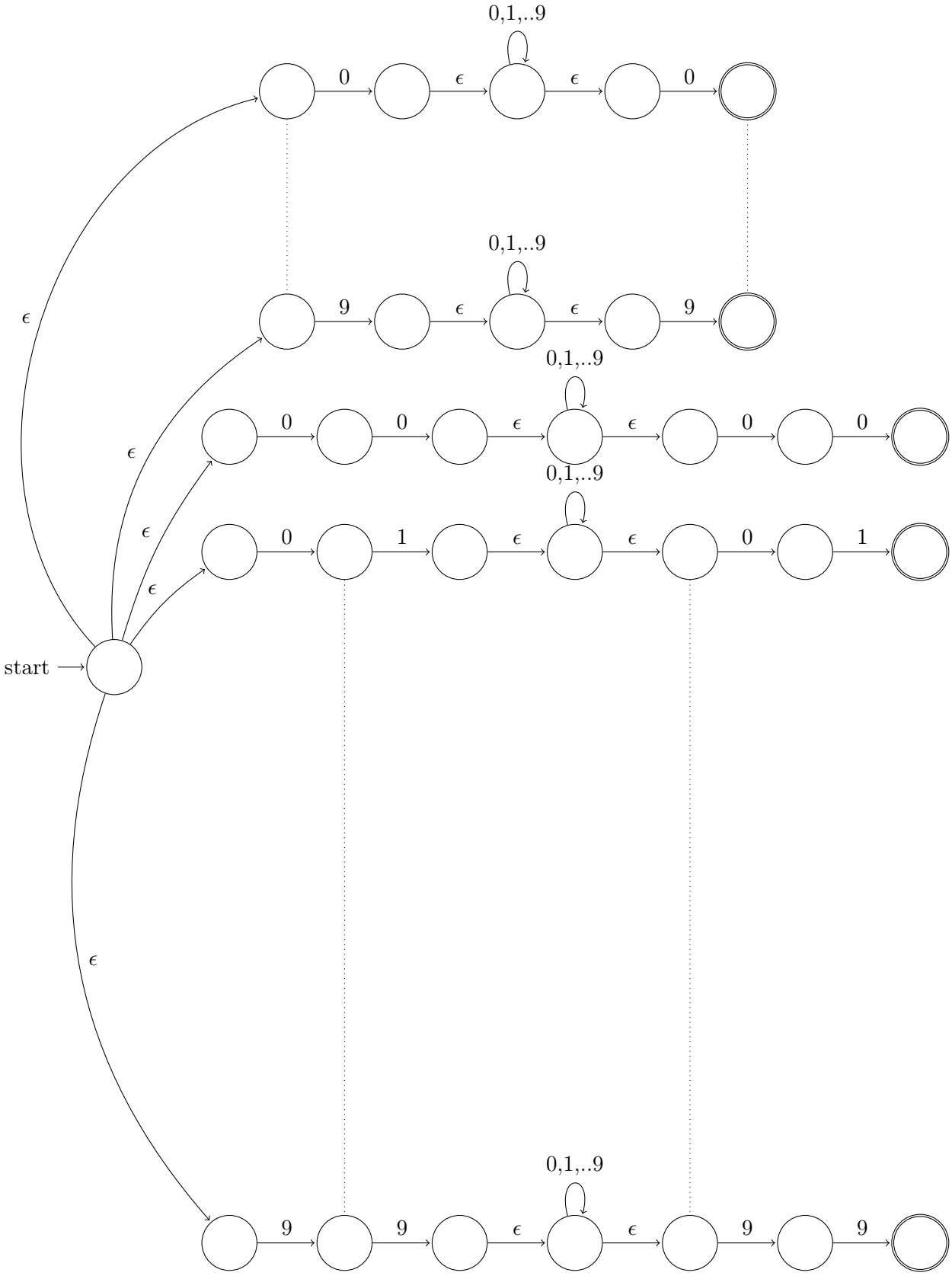
$$\text{NFA with}\,\epsilon \equiv \text{NFA without}\,\epsilon \equiv DFA$$

1.1 Example:

We generally try to come up with solutions using NFA or ϵ-NFA as it is more intuitive than DFA. In the example below we can see that ϵ-NFA construction is more intuitive than constructing DFA directly

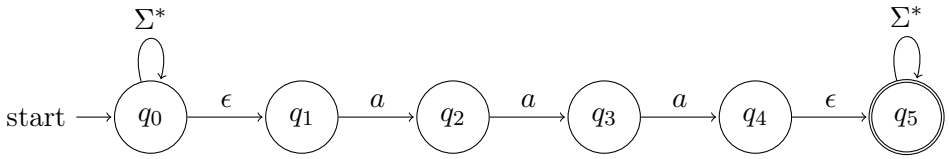
$$L = \{\omega \in \Sigma^* \mid \omega = u.v.w, u = w \in \Sigma^+, v \in \Sigma^*, |u| <= 2\}$$

Now intuitively, we thinks that the first digit and the last digit can be equal or the first two digits and last two digits has to be equal and we try to squeeze rest of the bits in between, we will use this idea to construct an ϵ-NFA.


0,1,..9
0
ϵ
ϵ
0
0,1,..9
9
ϵ
ϵ
9
ϵ
0,1,..9
0
0
ϵ
ϵ
0
0
ϵ
0,1,..9
0
1
ϵ
ϵ
0
1
ϵ
start
ϵ
0,1,..9
9
9
ϵ
ϵ
9
9

2 Application: Text Search

If we are given a text and asked to match all the words that contain pattern 'aaa', this can be solved by automata The NFA of this is ,



The ϵ-NFA can be converted into NFA without ϵ efficiently without blow up of states and given a string and NFA , we can construct only that part of DFA to decide if it is accepting or not in O(n.k), where n is the size of the NFA and k is the length of the string, which we have seen in last class.Since we can program DFA, the above solution can also be programmed after converting to DFA to get all the words that contain pattern in the text

But to represent NFA we need language and syntax like we have in propositional logic. That language is Regular Exressions,this will be discussed in next class.