CS 208 : Automata Theory and Logic 14/02/2024

Lecture - 16

Topic: DFA and $\epsilon-$edge NFA

Scribed by: Lovesh Mahar (22B0975)
Checked and compiled by:

We found a DFA equivalent A^* of NFA A In previous class. Now we will show that $L(A^*) = L(A)$, i.e., any language accepted by A^* is also accepted by A and they are the only accepted languages by A.
Since $L(A)$ and $L(A^*)$ are sets, therefore to show $L(A^*) = L(A)$, we will have to show that,

1. $L(A^*) \subseteq L(A)$
2. $L(A) \subseteq L(A^*)$

We will show that, for every $n \geq 0$ and for every $\omega \in \Sigma^*$ such that $|w| = n$, NFA can reach state $q \in Q$ on reading ω iff DFA A^* reaches state $S \in 2^Q$ on reading ω, such that $q \in S$. By proving this statement we can show that $L(A) = L(A^*)$.
We will prove this by applying induction on n,
Base case : $n = 0$, means ω is ϵ. From the definition of the initial state of A^*, it will be a set of initial states of A, and we know that ϵ will be one of the initial states of A.
Hypothesis : Claim hold for $0 \leq n < k$
Induction : We have to show that the claim will hold for $n = k$.
Let $|\omega| = k$, we can write $\omega = \omega' a$, and ω' must be present in Σ^*.

Since, $|\omega'| = k - 1$ and our hypothesis holds for $n < k$. we can say that if NFA starting from state q_0 reaches q on reading w', then DFA on reading w' must also reach a state q' which is a set of states of NFA and it contains state q. Now from here, when NFA reads 'a' and reaches some state $\hat{q}$, then based on how we have constructed DFA from NFA, we can say that DFA on reading the letter 'a' must reach some state $\hat{q}'$ which will contain $\hat{q}$ and vice versa, hence proving the statement.

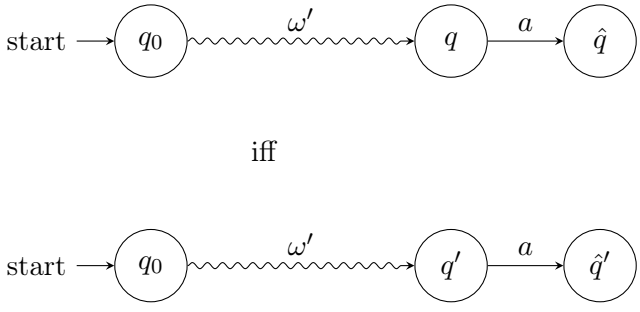

Figure 1: Language accepted by NFA and equivalent DFA will be same

Uses of NFA $\rightarrow$ NFA is used in the lexical analyzer of all the compilers. Their job is to tokenize the program.

1 NFA with $\epsilon-$edges

A variation of NFA is NFA with ϵ edges, it may expand the language by making words accepted that were otherwise unaccepted.

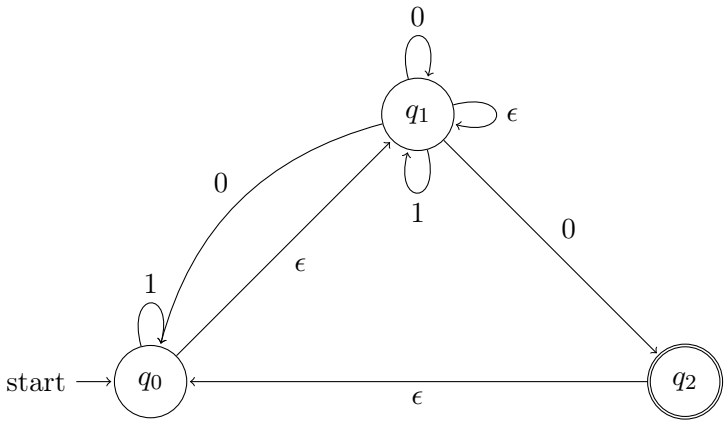

Figure 2: Automaton with Epsilon Edges

$\epsilon-$edges bring non-determinism to the NFA, as you can sit on a node and take one of the $\epsilon-$edges possible from that node to jump to another node without consuming any letter of the input.
Figure 2 shows how ϵ edges are used to connect states of an automaton for free(without consuming any letter from input), we can see that $10 \notin L$ without the ϵ edge between q_0 and q_1, but with the presence of this ϵ edge $10 \in L$.
$\epsilon-$edge also allows us to connect two automatons. In Figure 3, the accepting states of L_1 are connected to the start states of L_2 automaton, which generates an automaton for accepting $L_1 \cdot L_2$. Here $\cdot$ is the concatenation operator. So if $\omega_1 \in L_1$ and $\omega_2 \in L_2$, then $\omega_1 \cdot \omega_2 \in L_1 \cdot L_2$ will be accepted by this new automaton.

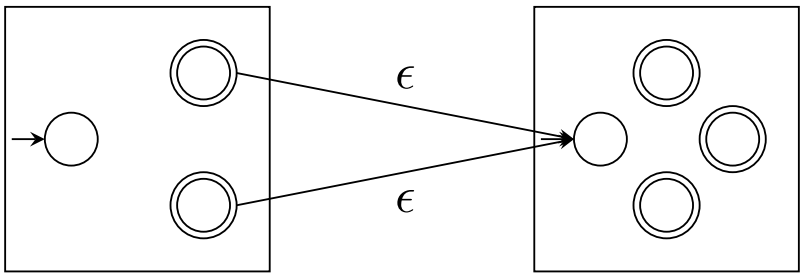

Figure 3: L_1 accepting automaton and L_2 accepting automaton connected

Now, we will try to find an equivalent DFA of this NFA having ϵ edges. For that, we will first find an NFA without ϵ edges which will preserve the original NFA, and this obtained NFA can then be constructed into a DFA.
Initially, just look at the $\epsilon-$edges only and find for each state its $\epsilon-$closure, which is the set of the states that we can reach from it by taking only $\epsilon-$edges.
$\epsilon-$closure for the states in the figure 1 will be,

$$q_0 \implies \{q_0, q_1\} = \epsilon - \text{closure } (q_0)$$
$$q_1 \implies \{q_1\} = \epsilon - \text{closure } (q_1)$$
$$q_2 \implies \{q_0, q_1, q_2\} = \epsilon - \text{closure } (q_2)$$

From each node, we can go to every node that is present in the $\epsilon-$closure of that node for free. So wherever non-ϵ edges take us from these nodes in $\epsilon-$closure, we can reach from the node itself whose closure it was. So all these states will be connected to this node, in the new NFA.
The starting states of this new NFA will be the $\epsilon-$closures of the start states of the original NFA and final states of the new NFA will be the states in whose $\epsilon-$closures the original final state belongs.
So NFA without $\epsilon-$edges for the NFA in Figure 2 will look like as shown in Figure 4

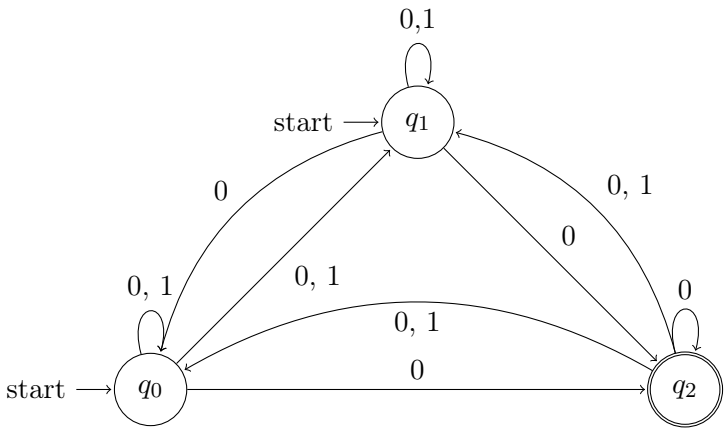

Figure 4: Automaton without Epsilon Edges