CSC301

Lecture 6 summary

1910456

Up to now, we have studied what languages are (generally speaking). We have seen how alphabets, strings are related.

There is a set of language called regular language. We have explored the characteristics of regular languages. We can express them through DFAs. We have also discussed NFAs and the distinctions.

Regular expressions are a way to describe languages formally.

Regular Expressions to NFA

Any DFA is an NFA, (subset of NFAs).

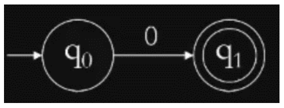
If we can show that with a regex, we can create an NFA, and with an NFA/DFA we can make a regex, we can complete the “cycle”.

We can generate strings from Regexes.

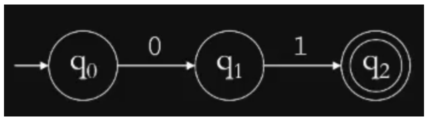
In automata, we do not generate, but we consume/read/check strings.

Therefore, they convey a similar message. (Regular Language).

$R_1 = 0$ is a regular expression.



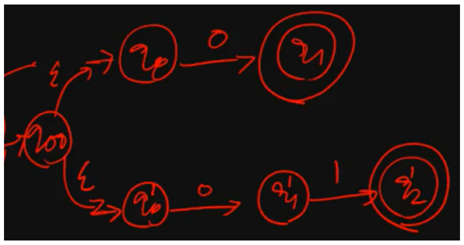
$R_2 = 01$ is also a regular expression.



We note that it is simply the NFA above but there is an extra state that checks for 1.

$R_3 = 0+01$

It is the combination of R_1 and R_2. With ε starting both states at once through a branch.

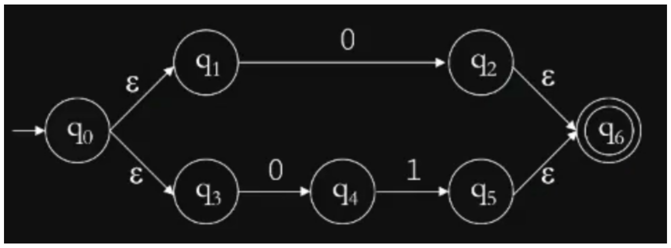


So, we basically divide our work into smaller expressions. Firstly, we make NFA for 0, then 01, then branch them for 0+01.

We can reduce the number of states by taking the "intuitive approach" but for larger regexes, it will be much harder if we do not follow a set of rules. Also, if we have to modify a regex, we cannot use the intuitive approach at all, or else we will have to redo the whole NFA.

In an NFA, ε is there in the definition of the transition function, it is not necessarily in the language/alphabet.

To prepare our NFA to be modular, we create them so that there is only one start state and one final state. So, for R_3, we combine the final state into one with two empty transitions.



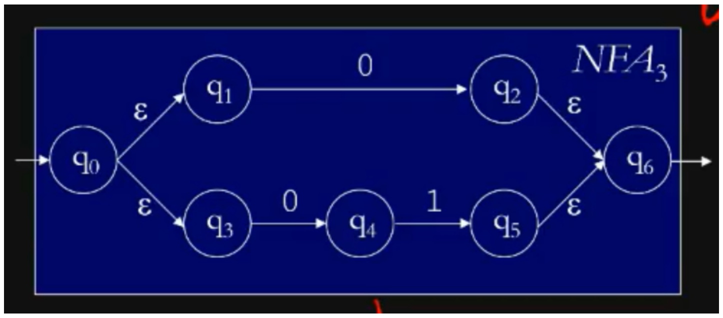
$R_4 = (0 + 01)^* = \{\varepsilon, 0, 01, 00, 0101, 001, \dots\}$

Firstly, since ε is accepted, for any Regex, the empty transition must go to the final state. Secondly, after reaching the final state, if we do not stop, we go back to the initial state.

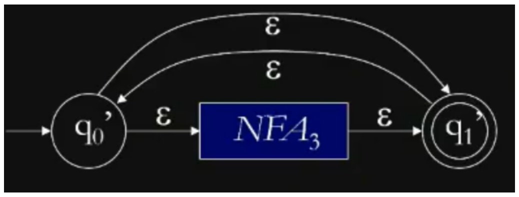
Meaning, if we reach the final state after getting 01, we can try 01 again and end up in the final state, (0101).

So, to go back to the initial state, we add another empty transition from final to initial state.

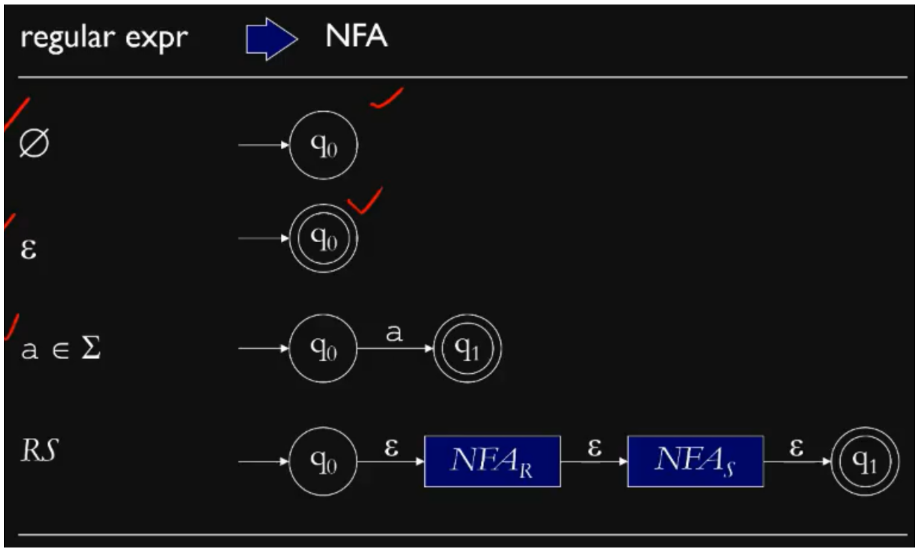
After we create the NFA for R_3, we may think of it as a black box named NFA_3, which has only one input and output.

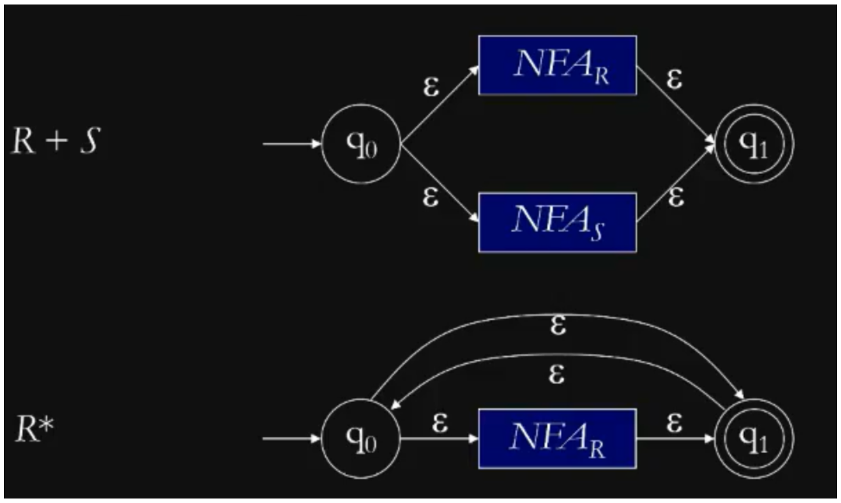


Now, to create R$_4$, we simply add the two empty transitions between the initial and final states. To keep modularity, we create two new states each time, one final and one initial and connect them with the I/O of the black box through empty transitions again.



General Method





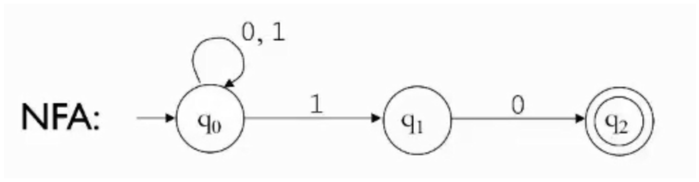
Examples shown.

NFA to DFA

For a DFA each state must have all transitions shown.

All DFAs are NFAs, even though they do not have empty transitions and multiple states for the same transition.

We recall, for NFAs, there might be/ might not be multiple transitions for the same symbol.



To convert NFAs to DFAs, there are two steps:

1. Eliminate ε- transitions.
2. Then convert the simplified NFA.

We are shown step 2 first.

Since, its an NFA, one symbol may result in us landing on multiple states.
So, when converting to DFAs, we create a new state: For ease, we name those new states the supposed states that we would be in the NFA.

For example. If we consume 1 in q_0 of the NFA, we transition to q_1 but also stay on q_0. Therefore, in our DFA, we create a new state named $\{q_0, q_1\}$ and go to that state when we consume 1.

Since, it's a DFA, we now have to mention what will happen for 0 and 1 in state $\{q_0, q_1\}$.

If we get 0 in state $\{q_0, q_1\}$, we stay at q_0 but also transition to q_2. Therefore, we must make a new state again, aptly named $\{q_0, q_2\}$.

If we get 1 in state $\{q_0, q_1\}$, we stay at $\{q_0, q_1\}$.

We will look at it in detail in the next lecture.