Lecture 5 Summary

This lecture starts with a lemma:

Lemma: if A is a regular language, then A = L(R) for some regular expression R.

Proof by Algorithmic Construction

A is regular, so there exists a Deterministic Finite State Automaton that recognizes A.

Convert M to a Generalized Non-Deterministic Finite State Automaton (Which means that each edge connecting two states may be associated with a regular expression instead of separate elements of Σ.

Claim: Any Deterministic Finite State Automaton or Non-Deterministic Finite State Automaton is a Generalized Non-Deterministic Finite State Automaton.

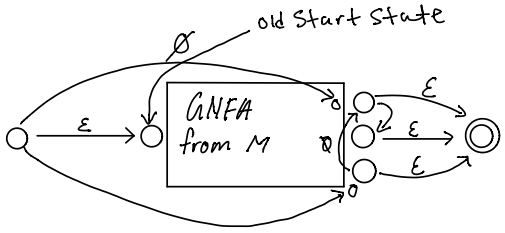
Use Generalized Non-Deterministic Finite State Automaton of a particular type:

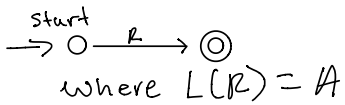
Start state has an edge to every state in the machine except for itself and any accept state, and no in arrows

Only a single accept state and it has in arrows from every state except the start state and itself, and it has not out arrows.

Except for the start and accept states every state has an arrow to every other state, including itself

We will transform Generalized Non-Deterministic Finite Automaton (GNFA) from M to satisfy these conditions.





First add a new accept state. The accept states of the original machine M will become regular states, and they will all share an edge leading from themselves to the new accept state of the new machine using ε.

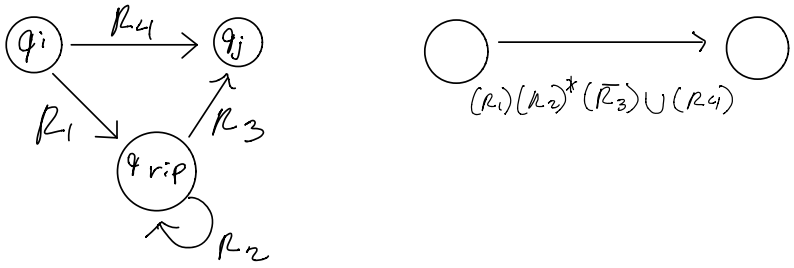
A new start state is added. And that new start state will lead to the start state of the original M using ε.

Add new edges to every state other state in the machine which originated from the new starting state, as well as between every other state among themselves. The transition across these edges will be ø.

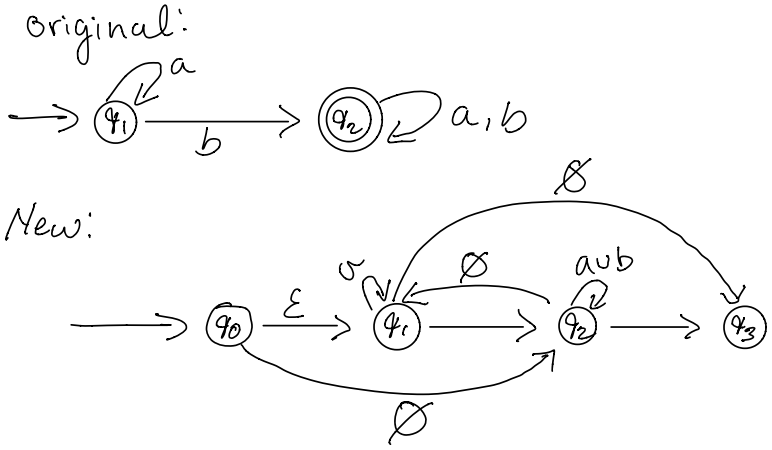
Convert the GNFA for A to a regex R, by removing one state at a time and updating some regex on some edges.

Pick some state q_rip of the GNFA A to a regex R by removing one state at a time, and updating some regex on some edges.

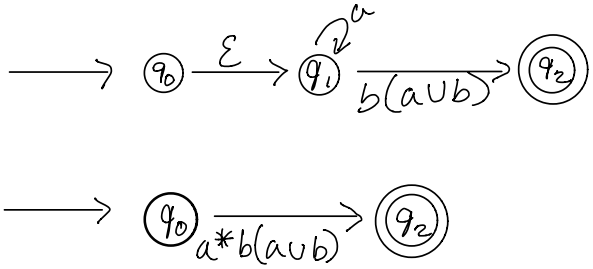
The for every pair of states q_i, q_j, update the regex on the edge from q_i to q_j. The new expression represents any string which could be processed when going from q_i to q_j.



Converting a DFA to a GNFA:



Rip edges



Next in the lecture, it is proved that there are languages that are **not** regular:

$$B = \{ 0^n 1^n : n \geq 0 \}$$

Claim: B is not regular.

Proof: By contradiction

Suppose that B is regular, so there exists some DFA M that recognizes B. So M has ome finite number of p of states.

Now B has some strings where n > p. Let s be one of those strings.

If we compute M on s, within the first p+1 states visited, we must have visited some states q twice (pigeonhole principle).

Let x be the part of the processed before the first visit to q.

Let y be the part of s processed after x, until q is entered the second time.

Let z be the remainder of s, processed just after y.

Now consider: s' = xyyz. Is s' ∈ B?

Claim: s' not in B

s' = xyyz is a string of only 0's.

So xyyz has more 0's than 1's

So s' cannot be in B.

Claim: s' ∈ B

If y is the part of the s' that loops back into q, then y performed again must take s' back to q, and allows it to continue on to z and the accept state.

So, s' must be in B

This is a contradiction, so, B cannot be a regular language.