**Main topics of the week:**

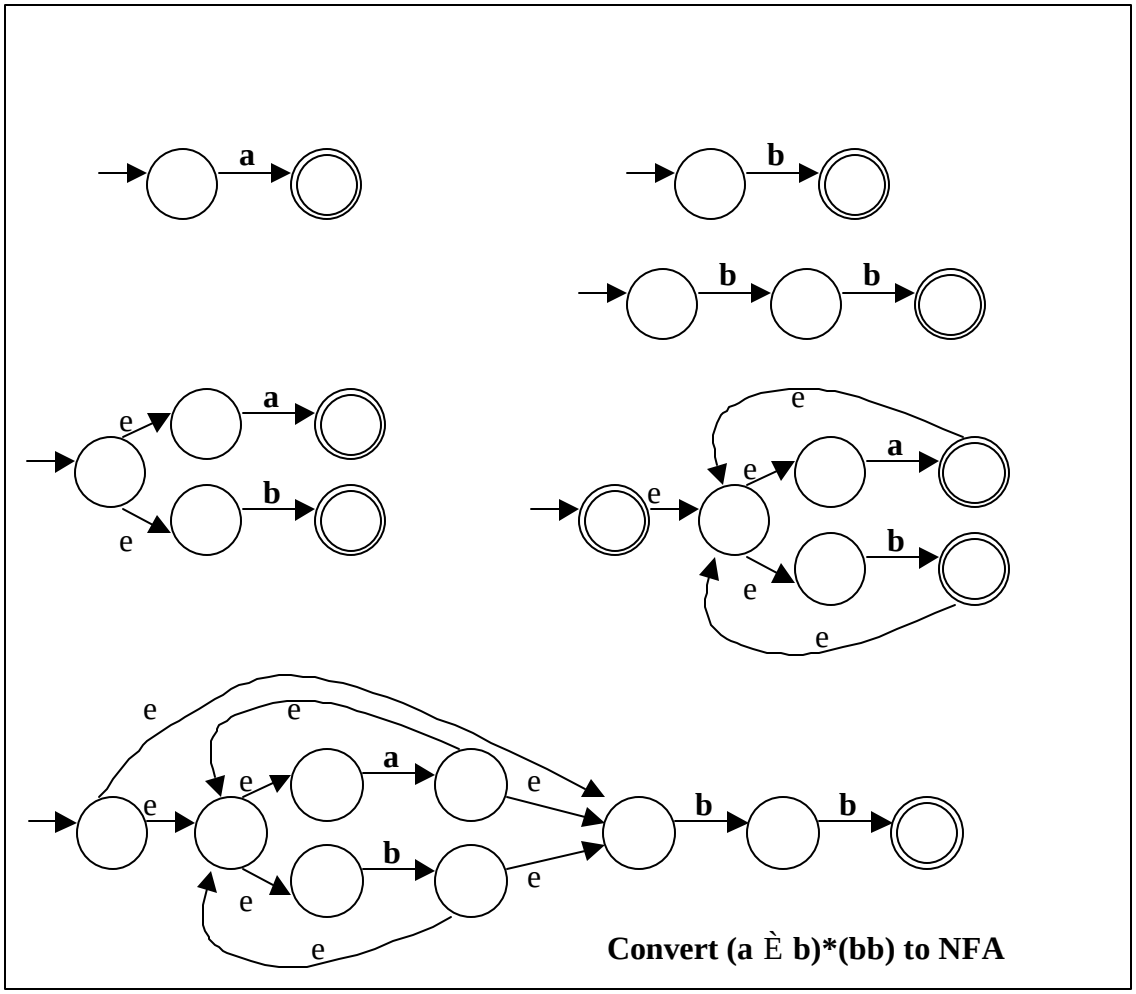
- **Examples and inductive definition of regular expressions**
- **Equivalence of regular languages and regular expressions**
- **Construction of NFA from a regular expression**
- **Construction of GNFA from DFA**
- **Reduction of GNFA to a regular expression**

**Regular Expression Identities**: There are a few basic identities we can observe about regular expressions:

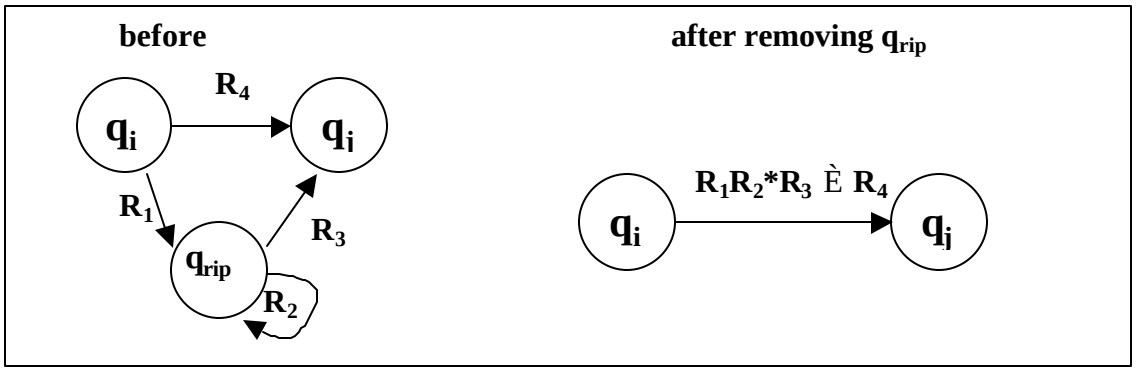
1. $R \cup \varnothing = R$
2. $R \circ \varepsilon = R$

These behave like identity elements, e.g., adding the $\varnothing$ to a regular expression doesn't change it, and composing it with $\varepsilon$ doesn't change it. Composing a regular expression with $\varnothing$ reduces it to $\varnothing$, however, composing a regular expression with $\varnothing^*$ does not change it since $\varnothing^*$ is just the regular expression $\varepsilon$.

**Example of constructing an NFA from a regular expression:**

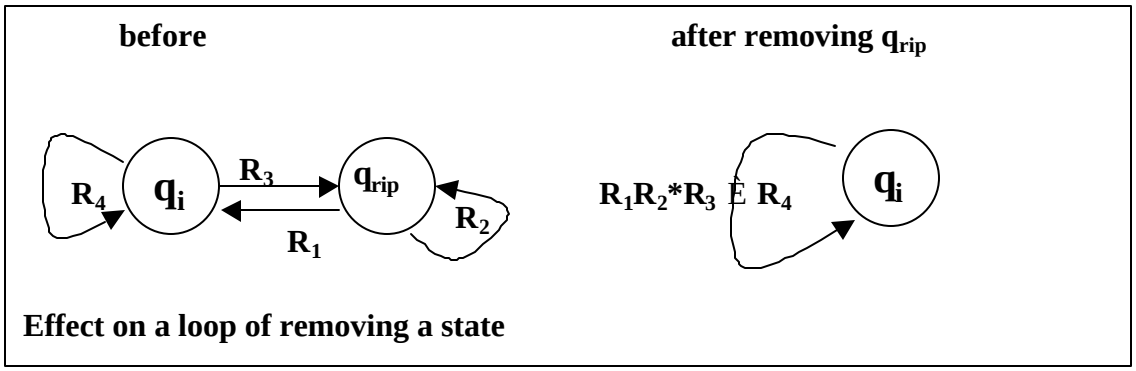


**Ripping states out of a GNFA**: The pictured technique shows how to rip out a state from a GNFA without affecting the language recognized by the GNFA (at the cost of more complicated RE labels on the remaining state transitions). If we can rip out states one at a time until we are down to just the start and accept state, we will reduce to a very simple GNFA that has one arrow labeled with a regular expression. The following diagram shows this reduction.




The diagram only shows what happens to the transition from $q_i$ to $q_j$. What we are doing is enhancing the regular expression that goes between the remaining states with the things we lose when we rip out a state. This includes what gets us into the ripped out state, looping in the ripped out state, and out of the ripped out state. Essentially, we look at what the ripped out state contributes as a possible path between the remaining states, and union it with the original direct path between those two states. Of course, we're showing this in isolation, with only one direction. In a real setting there would be both directions as well as all the other states the ripped out state is connected to. But in terms of re-labeling arrows, what we show is exactly what happens for all the affected arrows.

One special case is worth considering: when $q_i$ and $q_j$ are the same state. In the following diagram, we show what this looks like. Here, the $R_4$ label from $q_i$ to $q_j$ is just the loop on $q_i$, and that is the arrow affected by removing $q_{rip}$.



**Example of reduction for DFA with two states**. We first convert the DFA to a GNFA (four states), then reduce the states to three and then to two to end up with the RE equivalent to the DFA. The first conversion to a GNFA is by adding start and accept states and ε transitions in and out of them to/from the original start/accept states. We also combine arrows with union and add arrows with ∅. Here we won't bother with the ∅ arrows, but just remember that a missing arrow is labeled ∅. Also we don't have any multiple arrows to combine with union. In our four state GNFA, we go about eliminating states. First we eliminate state 2, and since we could get from 1 to accept via 2, we use our rule to compose the regular expression of the arrows from 1 to 2 to 2 to accept, getting bb*ε. We suppress the ε as we have seen earlier is an identity. Likewise, we are really taking the union of this with the arrow directly from 1 to accept (which is labeled ∅) and again by our identity, we can suppress this part. When we eliminate 2, we are also collapsing a path from 1 back to itself, so we use the rule in the degenerate case of $q_i=q_j$ to adjust the loop on 1 to be a ∪ bb*a. Now we go about eliminating 1 in the same way and add ε(a ∪ bb*a)* to our composition, with this finally resulting in (a ∪ bb*a)*bb*.