Transforming a RegEx into an equivalent NFA

LING 570

Thompson's construction

- https://en.wikipedia.org/wiki/Thompson%27s_construction

- RegEx: recursive definition (see the next slide)

  ➔ Apply rules recursively

Definition of Regular Expression (as in formal language theory)

- The set of regular expressions is defined as follows:
  - (1) Every symbol of Σ is a regular expression
  - (2) ε is a regular expression
  - (3) If $\mathbf{r_1}$ and $\mathbf{r_2}$ are regular expressions, so are $(\mathbf{r_1})$, $\mathbf{r_1}\,\mathbf{r_2}$, $\mathbf{r_1} \mid \mathbf{r_2}$, $\mathbf{r_1}^*$
  - (4) Nothing else is a regular expression.

- Note:
  - r+ = r r*
  - r? = (r | e)

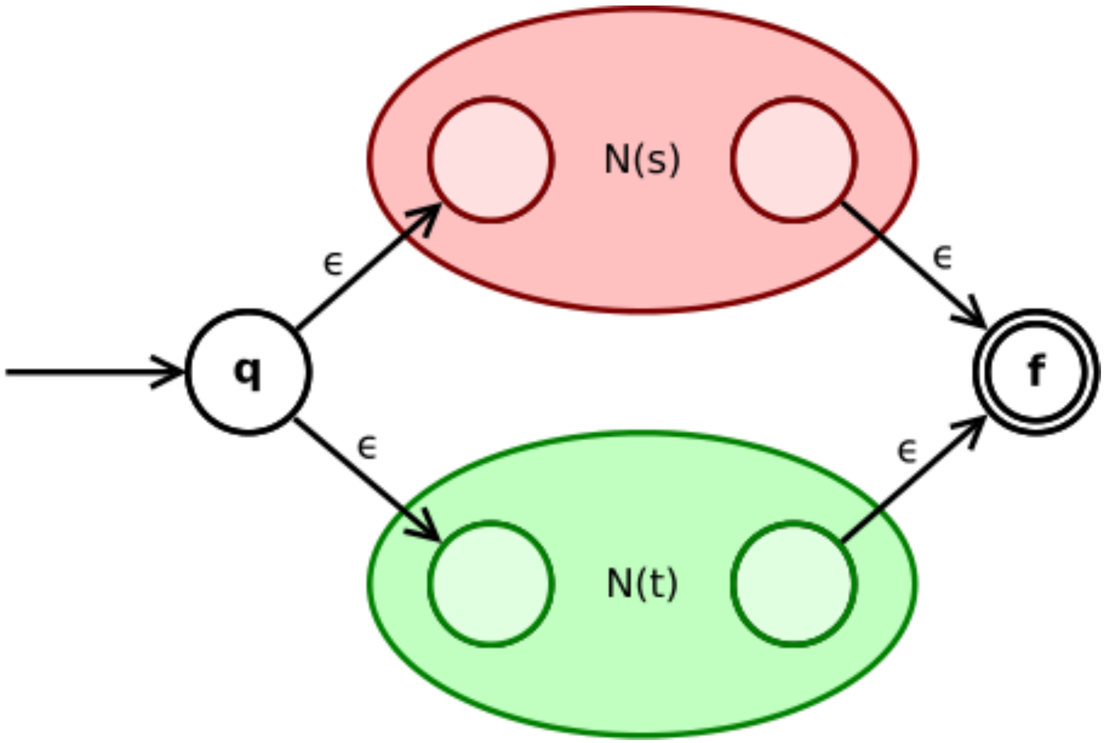
Base case

$r = \varepsilon$

$r = a$

Union expression

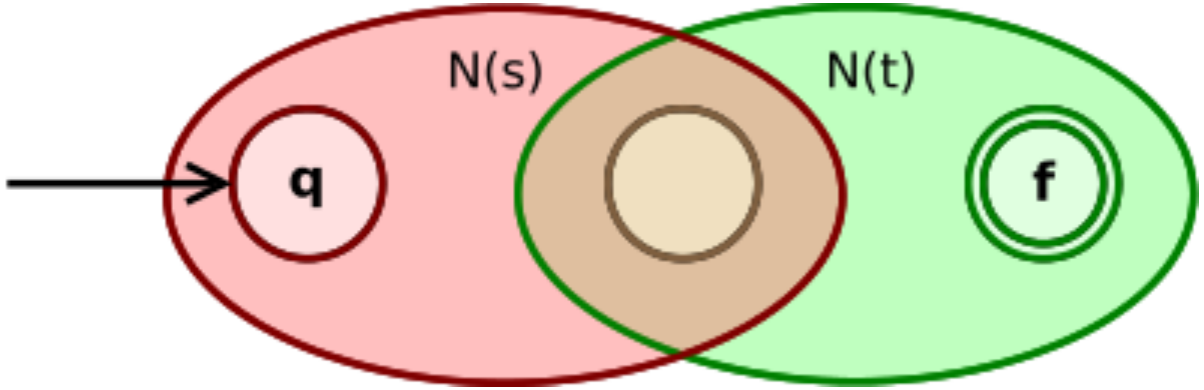
$r = s \mid t$



N(s) and N(t) are FSAs for s and t, respectively.

Concatenation expression

r = s t



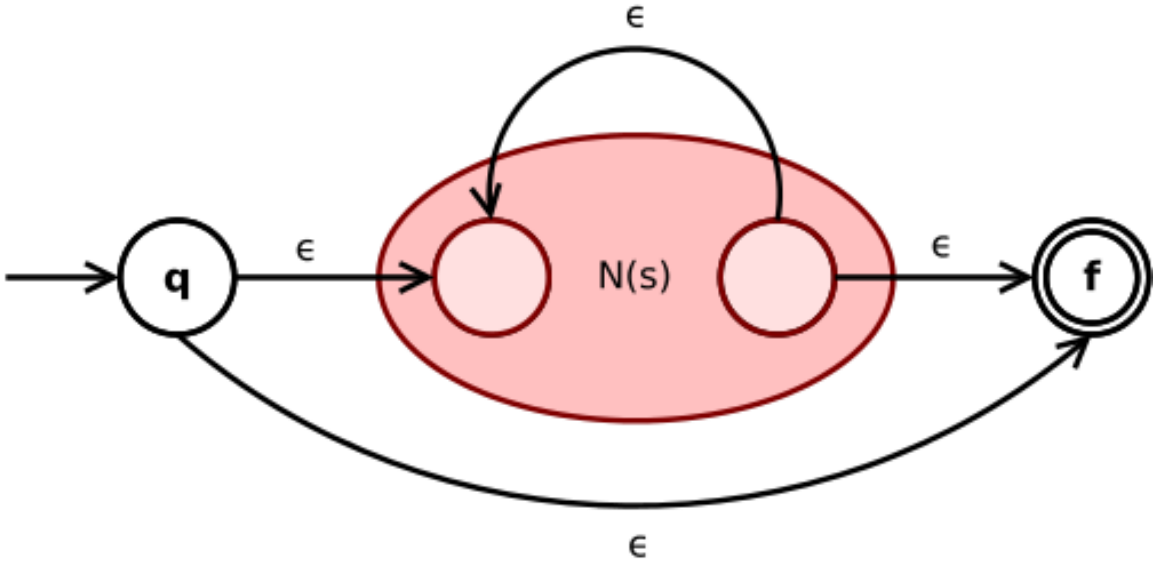
The initial state of *N*(*s*) is the initial state of the whole NFA. The final state of *N*(*s*) becomes the initial state of *N*(*t*). The final state of *N*(*t*) is the final state of the whole NFA.

Here we assume each FSA has only one initial state. If that's not the case, you can always create a new initial state that goes to each of the original states with e-transition. The same can be done to have a single final state.

Kleene star expression

s = (e | ab)

r = s*



Create a new start state q and a new final state f.

An ε-transition connects new initial and final state of the NFA with the sub-NFA *N*(*s*) in between.

Another ε-transition from the final state to the initial state of *N*(*s*) allows for repetition of expression *s* according to the star operator.