

# Transforming a RegEx to FSA

LING 570

# Thompson's construction

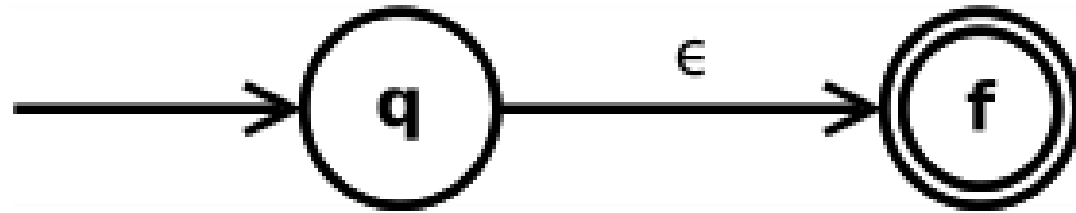
- [https://en.wikipedia.org/wiki/Thompson%27s\\_construction](https://en.wikipedia.org/wiki/Thompson%27s_construction)
- RegEx: recursive definition (see the next slide)
  - ➔ Apply rules recursively

# Definition of Regular Expression (as in formal language theory)

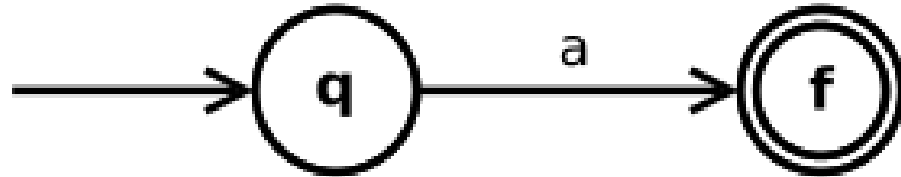
- The set of regular expressions is defined as follows:
  - (1) Every symbol of  $\Sigma$  is a regular expression
  - (2)  $\epsilon$  is a regular expression
  - (3) If  $r_1$  and  $r_2$  are regular expressions, so are  $(r_1)$ ,  $r_1 r_2$ ,  $r_1 \mid r_2$ ,  $r_1^*$
  - (4) Nothing else is a regular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

Base case

$r = \epsilon$

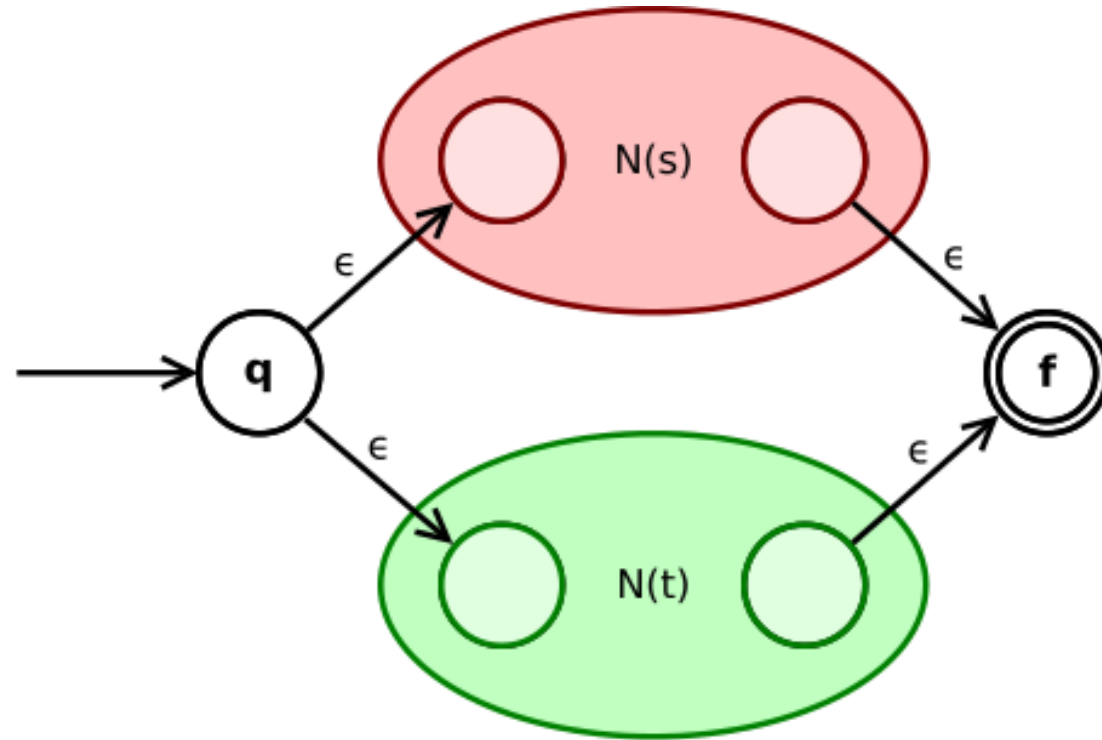


$r = a$



# Union expression

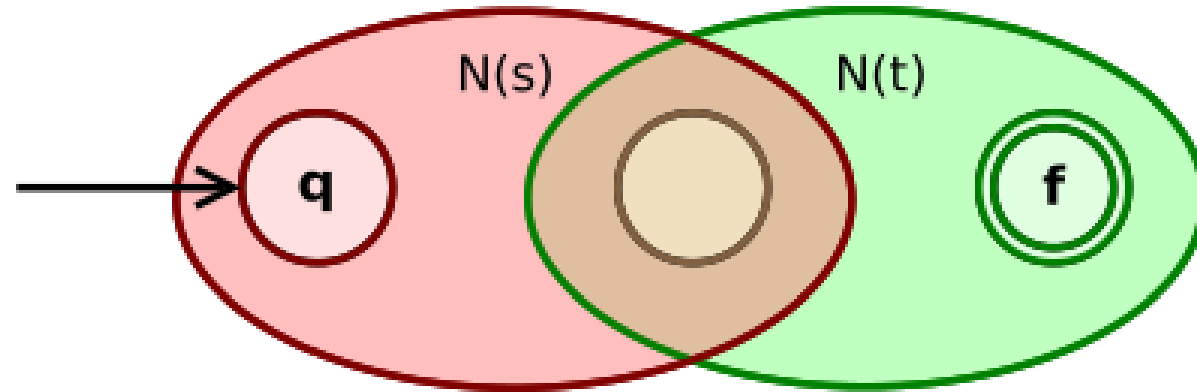
$$r = s \mid t$$



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

# Concatenation expression

$r = s t$

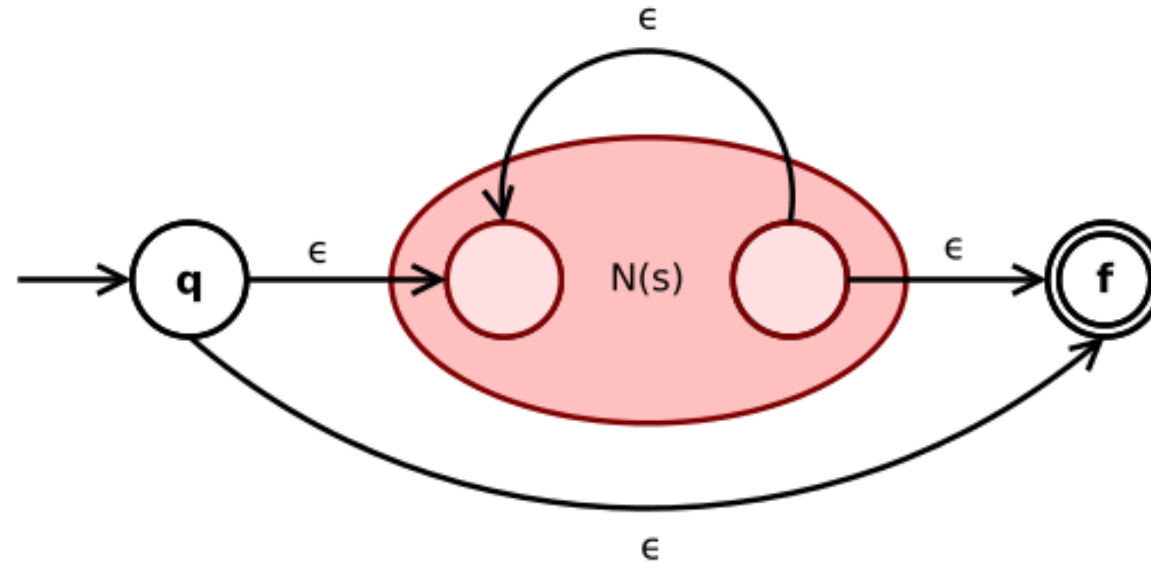


The initial state of  $N(\textcolor{red}{s})$  is the initial state of the whole NFA. The final state of  $N(\textcolor{red}{s})$  becomes the initial state of  $N(\textcolor{green}{t})$ . The final state of  $N(\textcolor{green}{t})$  is the final state of the whole NFA.

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  $\epsilon$ -transition. The same can be done to have a single final state.

# Kleene star expression

$$r = s^*$$



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

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

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