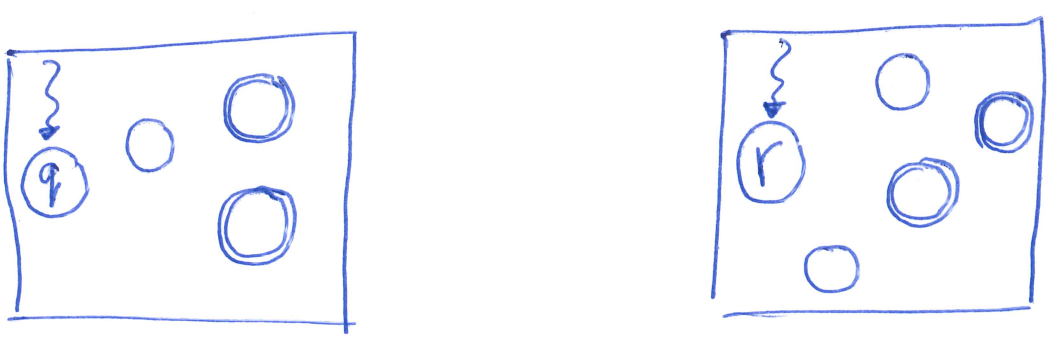
Class of Regular Languages is closed under $\cup, \circ, *$

Thanks to the equivalence of DFAs and NFAs, a language is regular iff an NFA accepts it. The characterization in terms of NFAs is very convenient to prove closure under $\cup, \circ, *$ operations

Theorem If A, B are regular languages, then so are $A \cup B$, $A \circ B$, A^*.

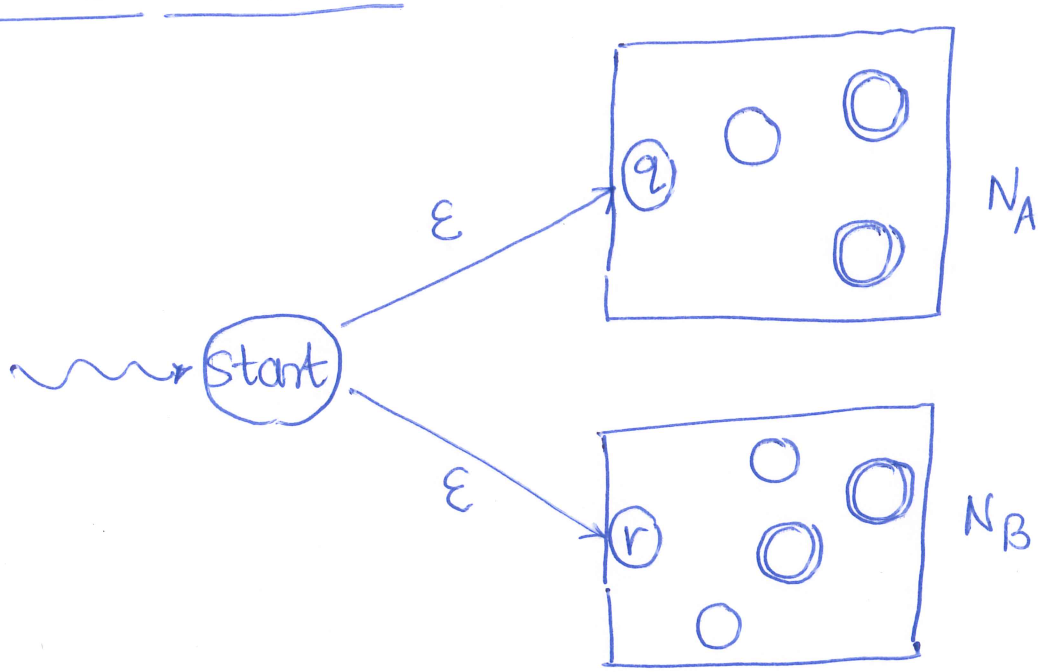
Proof Let N_A, N_B be NFAs that accept A and B respectively. We'll construct an NFA N that accepts $A \cup B$, $A \circ B$, or A^* (as the case may be).



N_A N_B

NFA for $A \cup B$

N

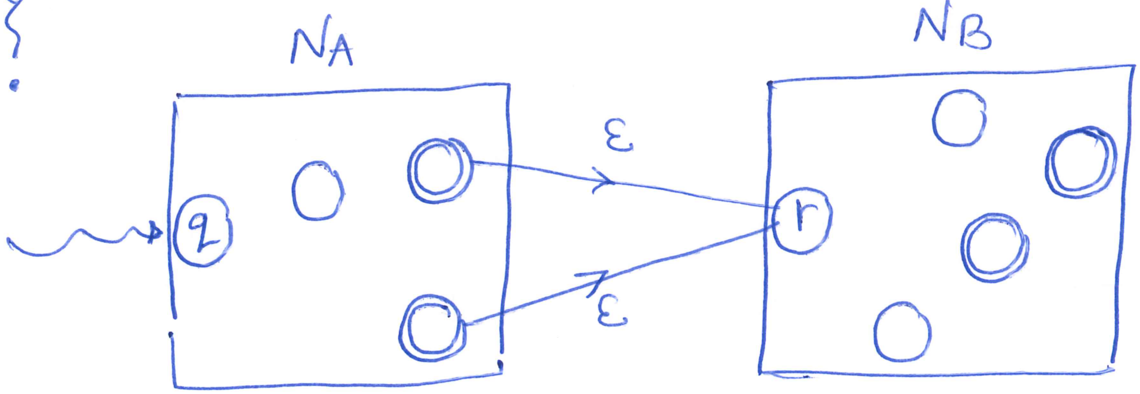


N has a new start state (start) and ε-moves from it to the start states q, r of N_A, N_B respectively.

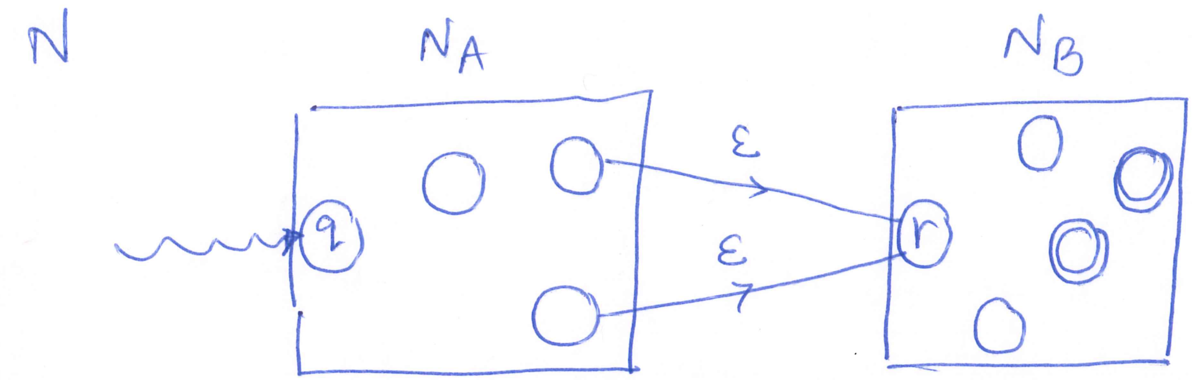
NFA for $A \circ B$

Would this work?

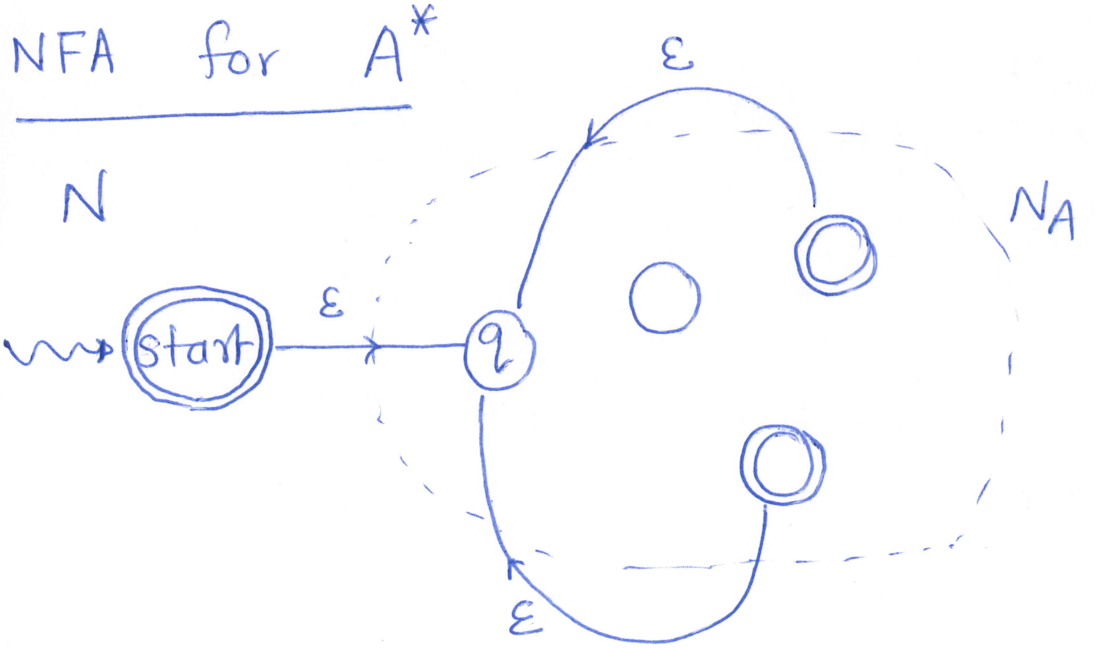
N?



Idea is to let q be the start state of N and introduce ε-moves from accept states of N_A to r. This works, but we must turn the accept states of N_A into non-accept states of N (why?). The final construction is:



NFA for A^*



Note that A^* consists of all strings $x_1 x_2 \cdots x_k$ s.t. $x_i \in A$ for every $1 \le i \le k$. We can start with N_A and add ε-moves from

its accept states to its start state q.
This ensures that

x_1 takes N_A from q to an accept state
then using ε-move, back to q, then
x_2 takes N_A from q to an accept state
then using ε-move, back to q,
---- etc .. etc and finally
x_k takes N_A from q to an accept state.

Thus N takes $x_1 x_2 \dots x_k$ as input and takes q to an accept state. Therefore N accepts A^*. (Almost!).

One needs to handle the string ε separately.
Note that $\varepsilon \in A^*$ (corresponding to $k=0$) even if $\varepsilon \notin A$. To fix this, we add the new start state Start and add the ε-move

$$\rightarrow \text{Start} \xrightarrow{\varepsilon} q.$$

which is also designated as accept state

<u>Exercise</u> - Give a formal construction of N in terms of N_A, N_B.

- Write the full argument (in English) that N accepts <u>precisely</u> the language $A \cup B$, $A \circ B$, or A^*, as the case may be.

Regular Expressions

DFA/NFAs give a characterization of regular languages in terms of a computational model. We'll see now an equivalent characterization that is <u>syntactic</u>, in terms of regular exprs.

<u>Examples</u> $\Sigma = \{0,1\}$.

(1) The regular expression $\{0 \cup 1\}^* 001$ describes the language

$L = \{ w \mid w \text{ ends with suffix } 001 \}$.

(2) $0\{0 \cup 1\}^* 0 \quad \cup \quad 1\{0 \cup 1\}^* 1$ describes

$L = \{ w \mid$ the first and the last symbol of w is the same. $\}$

③ $(01 \cup 10 \cup 00 \cup 11)^*$ describes

$L = \{ w \mid |w| \text{ is even} \}$.

Formal Inductive Definition

Def. R is a regular expression (over alphabet Σ)

if $R = a$ for some $a \in \Sigma$

or $R = \varepsilon$

or $R = \emptyset$

or $R = (R_1 \cup R_2)$

or $R = (R_1 \circ R_2)$

or $R = (R_1^*)$

where R_1, R_2 are regular expressions defined already.

Note While writing, we often omit the parantheses ( ) or the $\circ$ sign.

The language defined / described by an expression R is defined inductively in a natural manner.

Def The language $L(R)$ defined by regular expression R is

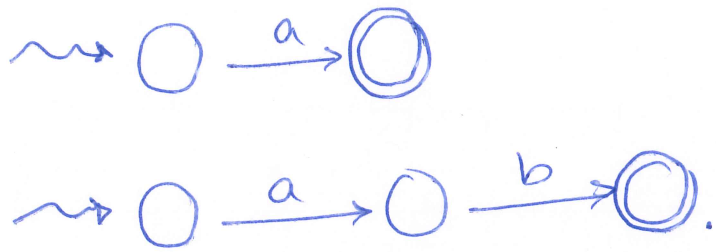
$$L(R) = \begin{cases} \emptyset & \text{if } R = \emptyset \\ \{\varepsilon\} & \text{if } R = \varepsilon \\ \{a\} & \text{if } R = a, \ a \in \Sigma \\ L(R_1) \cup L(R_2) & \text{if } R = R_1 \cup R_2 \\ L(R_1) \circ L(R_2) & \text{if } R = R_1 \circ R_2 \\ L(R_1)^* & \text{if } R = R_1^* . \end{cases}$$

Theorem A language L is regular <u>iff</u> $L = L(R)$ for some regular expression R.

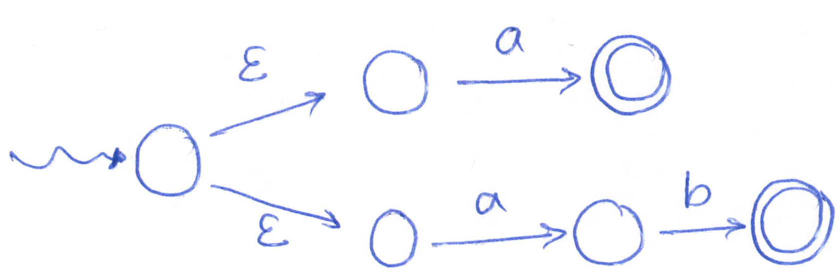
Proof of $\Leftarrow$:

It is easily observed that if R is a regular expression then $L(R)$ is regular. This follows simply from the above inductive definition of $L(R)$ and that the class of regular languages is closed under $\cup, \circ, *$.

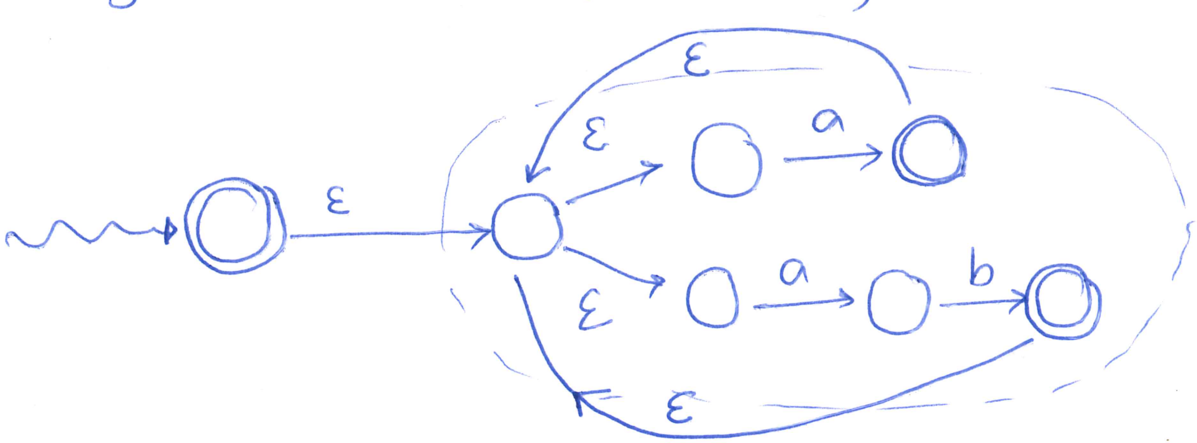
If one wishes, one can build an NFA accepting $L(R)$ by "parsing" the expression "bottom-up". E.g. for the expression $\underline{(ab \cup a)^*}$, we can first build NFAs for $\underline{ab}$ and $\underline{a}$ as



Then we build NFA for $ab \cup a$ using the NFA construction for $\cup$:



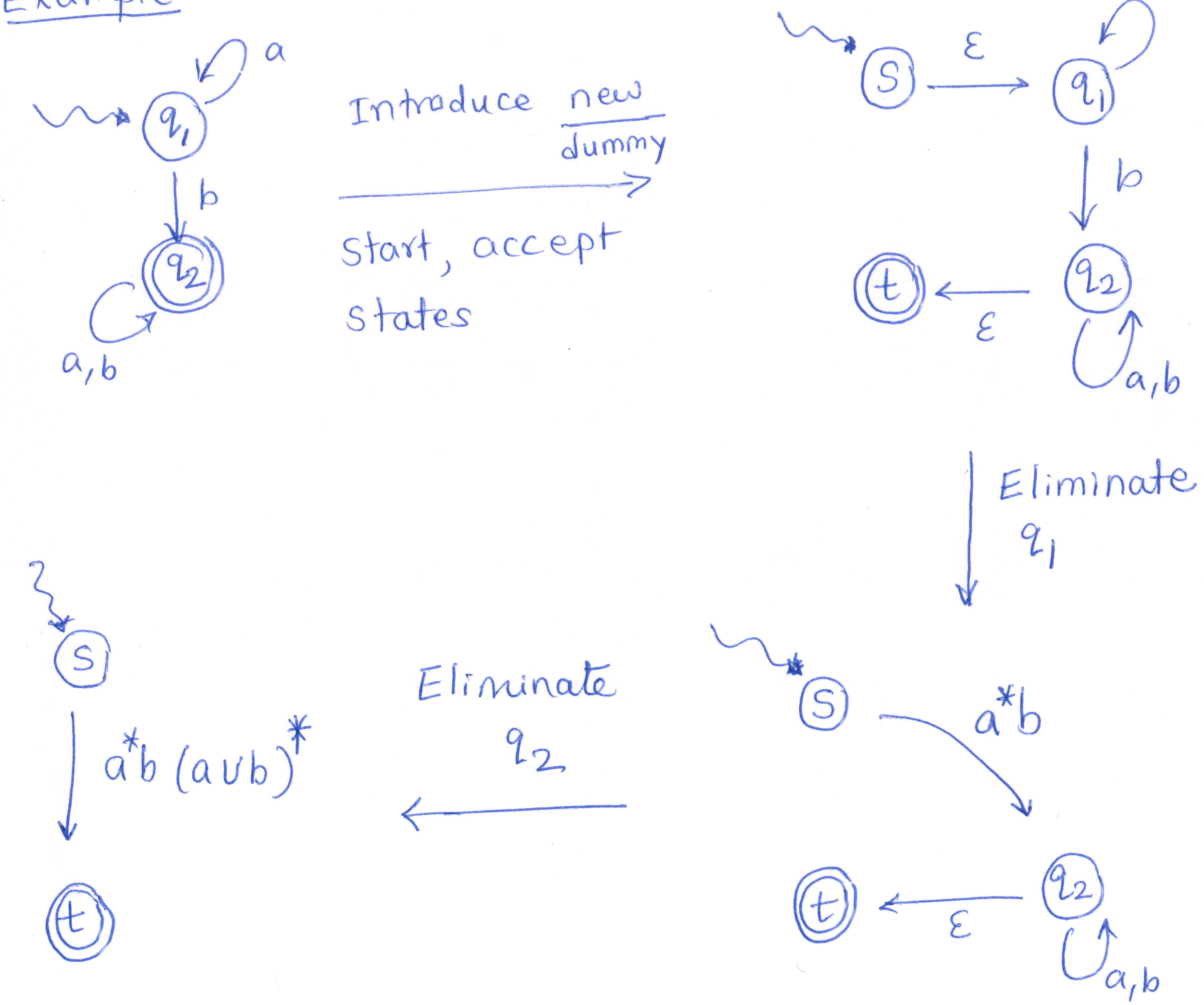
Finally NFA for $(ab \cup a)^*$, using the construction for $*$.



Proof of $\Rightarrow$

We now show that given an NFA, we can construct an equivalent regular expr.

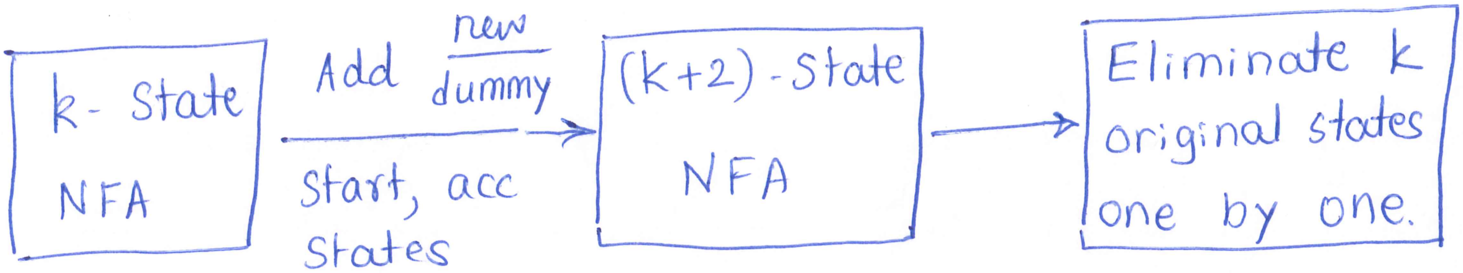
Example $\Sigma = \{a, b\}$.



The equivalent regular expression is

$$a^*b\,(a \cup b)^*.$$

The general construction can be sketched as:

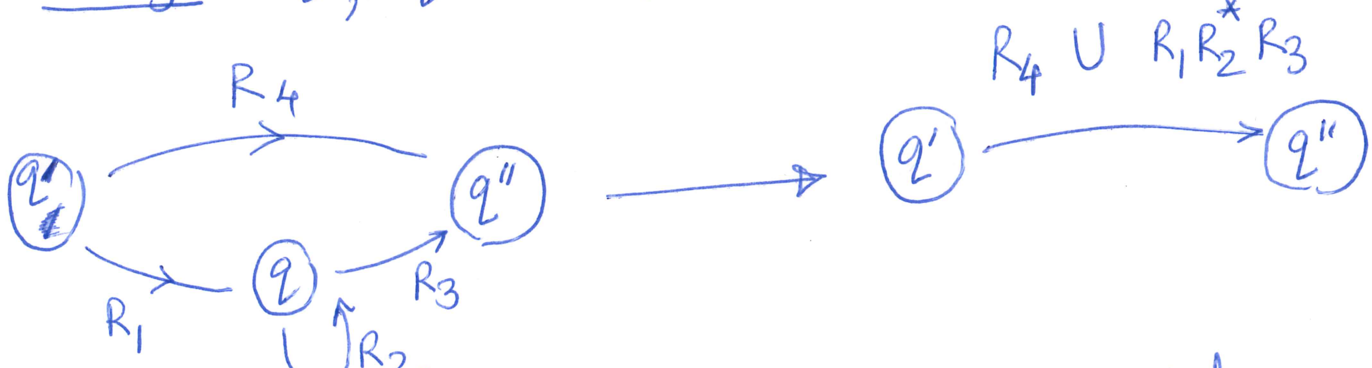


<u>Note</u> - The final regular expression may depend on order of elimination.

- During this process, one has NFAs whose transition arrows are labeled by regular expressions. Such NFAs may be referred to as Generalized NFAs, GNFAs.

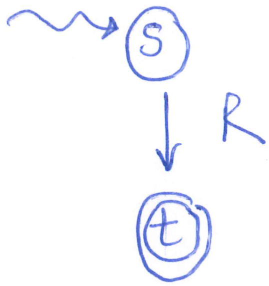
Eliminating State (q)

This involves the following operation for <u>every</u> $q', q'' \neq q$:



After these operations, (q) is deleted.

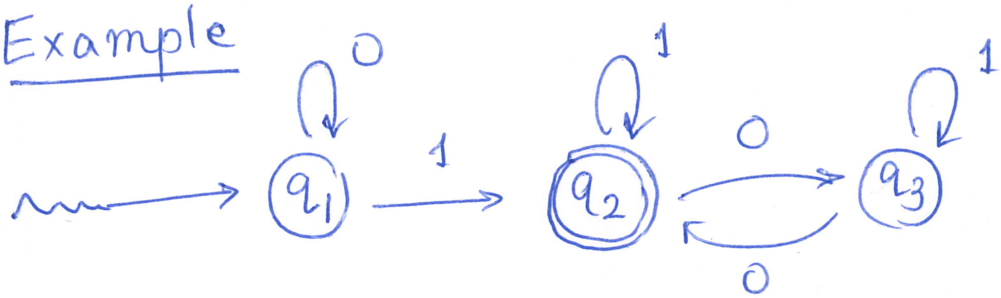
It may be the case that q', q'' are the same. After all elimination steps, one is left with the GNFA



where s, t are the dummy start, accept states.

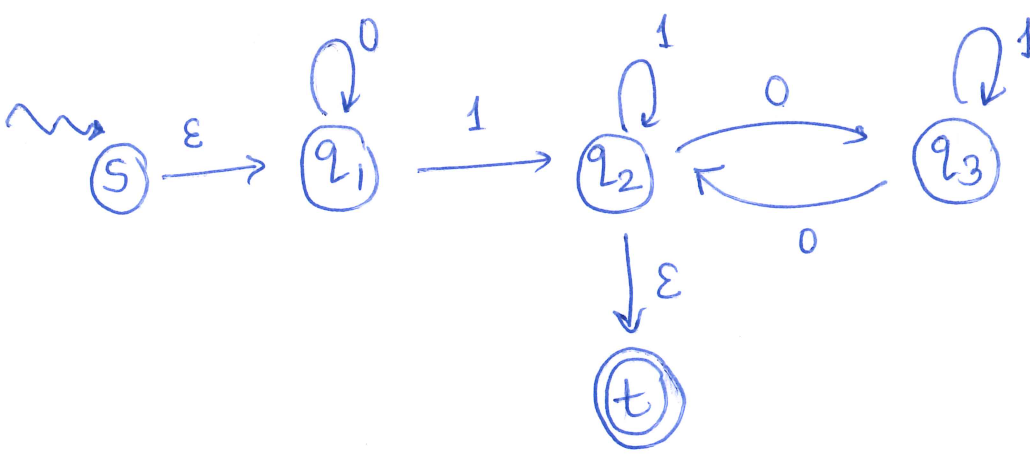
R is the final regular expression equivalent to original NFA.

Example

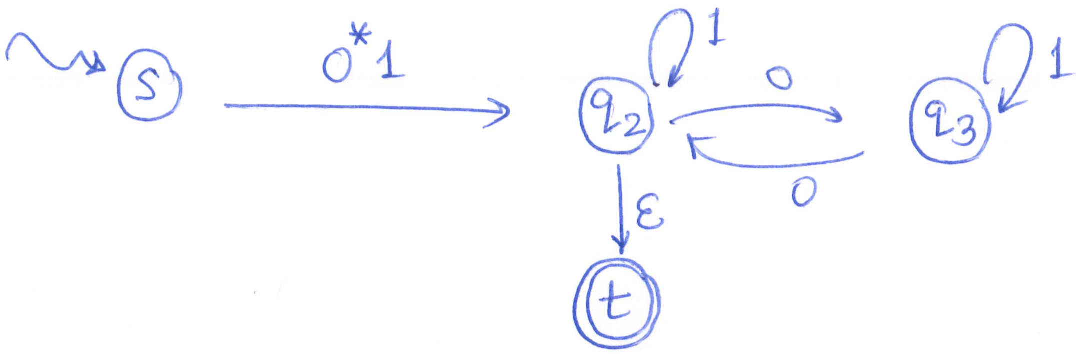


N

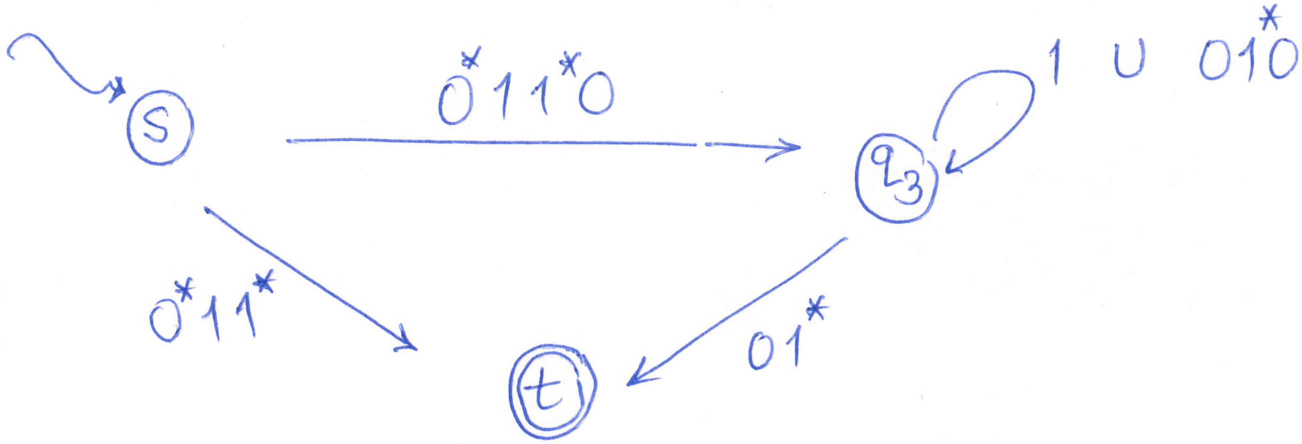
Add dummy states



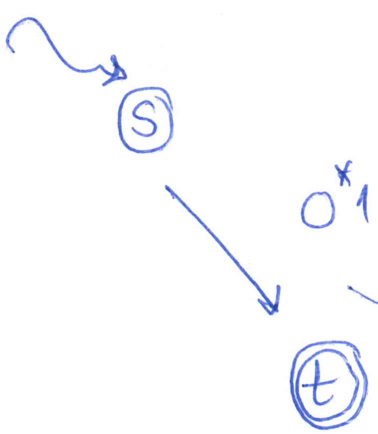
Eliminate q_1



Eliminate q_2



Eliminate q_3



$$\underbrace{0^*11^* \cup \left(0^*11^*0\,(1 \cup 01^*0)^*\,01^*\right)}_{R}.$$

R is a regular expression that is equivalent to the NFA N.