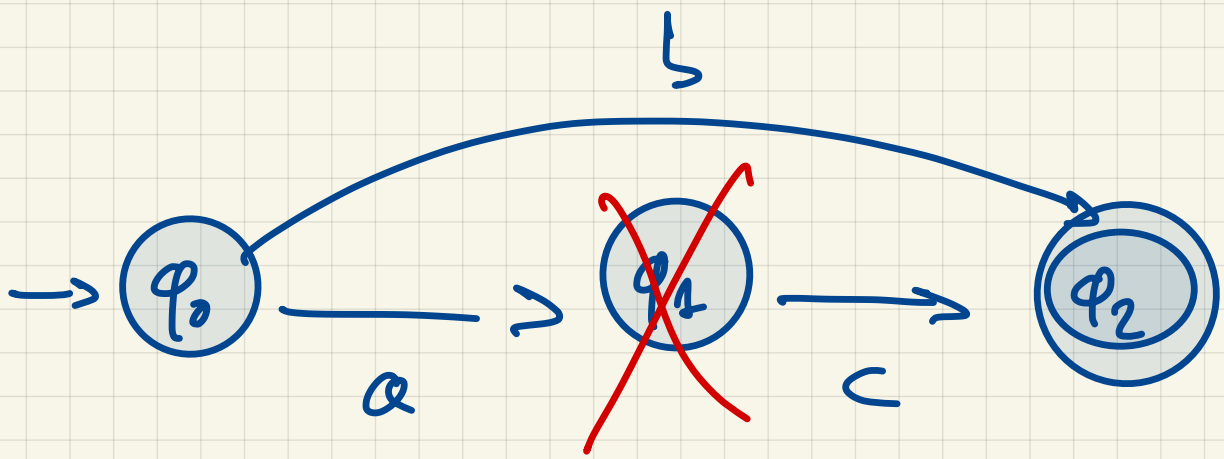
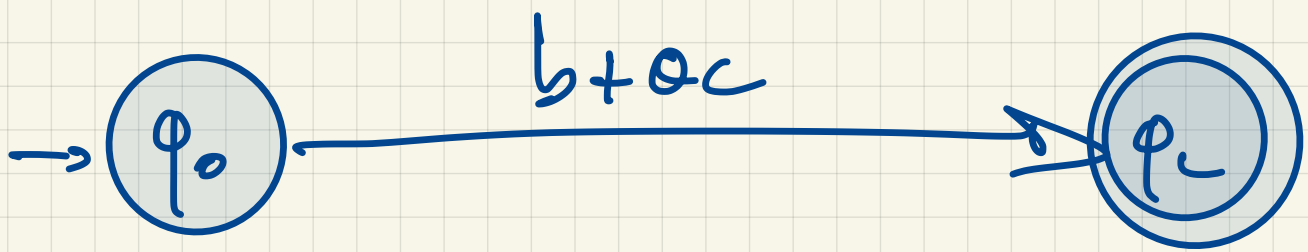
Octoler, 3rd

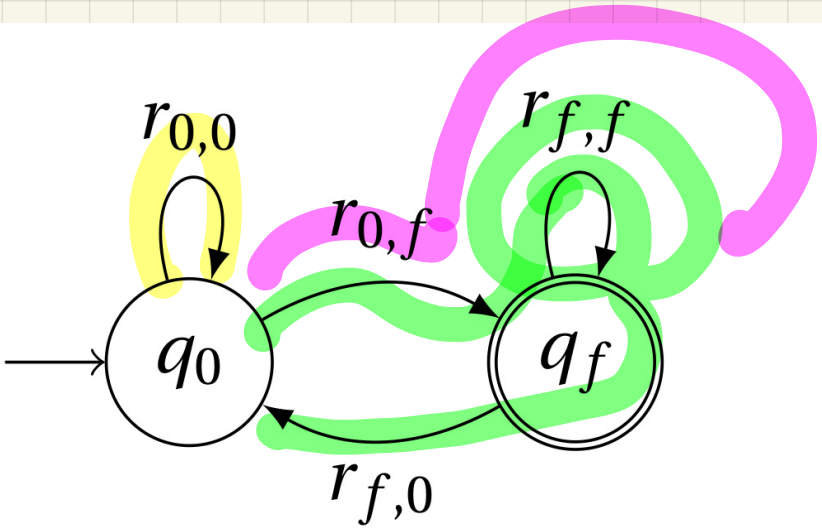
From FA to Reg. Exp.

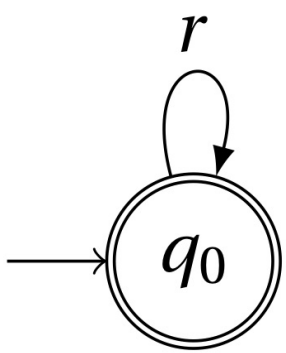




Idea of the transformation:

Iteratively remove states from the automaton until we end up with a very simple automaton with one or two states.

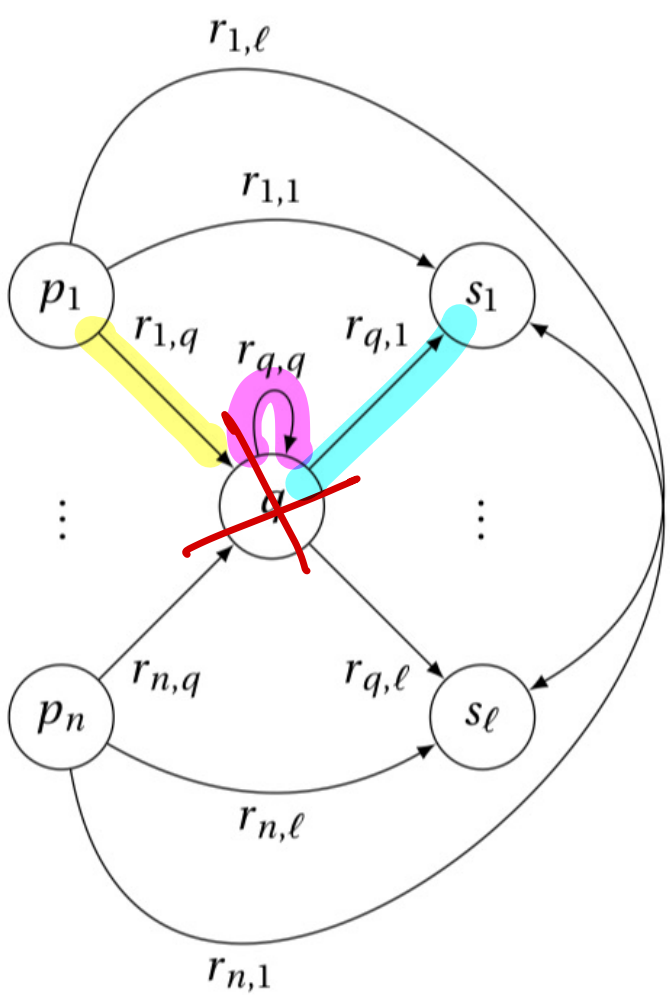


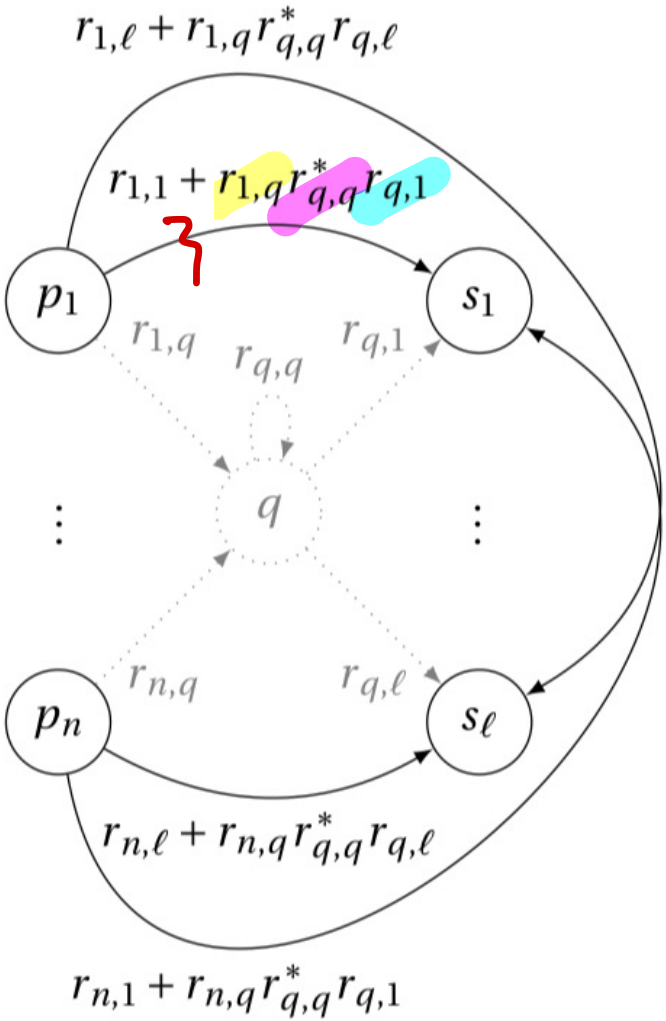


$$\left(r_{0,0} + r_{0,f} \cdot r_{f,f}^{*} \cdot r_{f,0}\right)^{*} \cdot r_{0,f} \cdot r_{f,f}^{*}$$

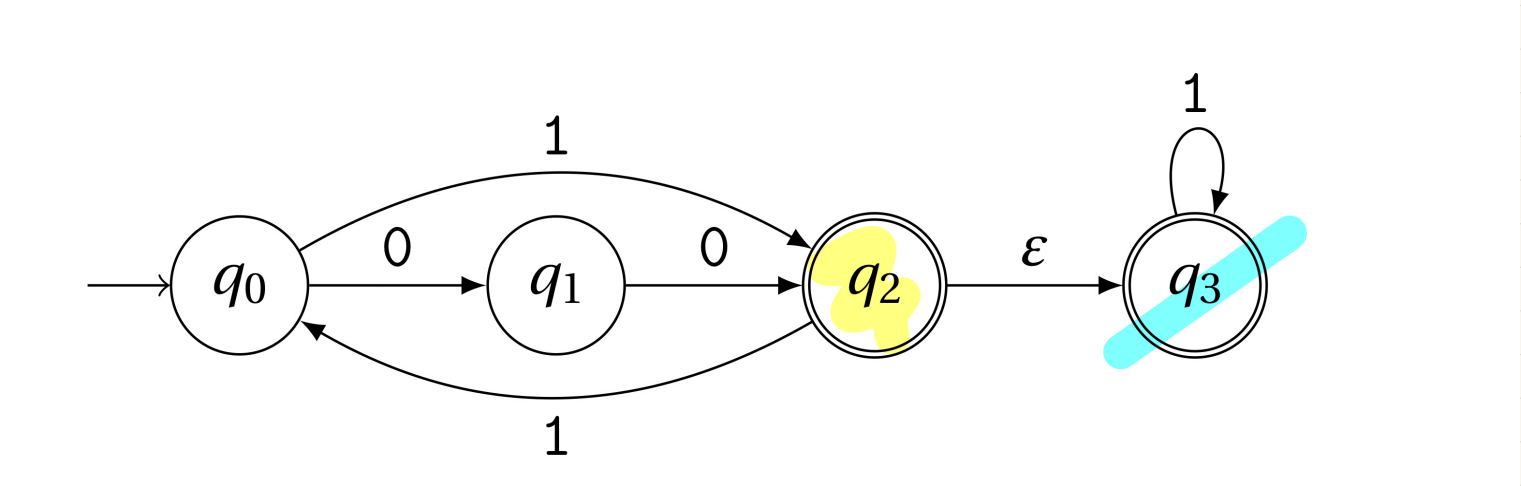
$$r^{*}$$

How do we remove states?





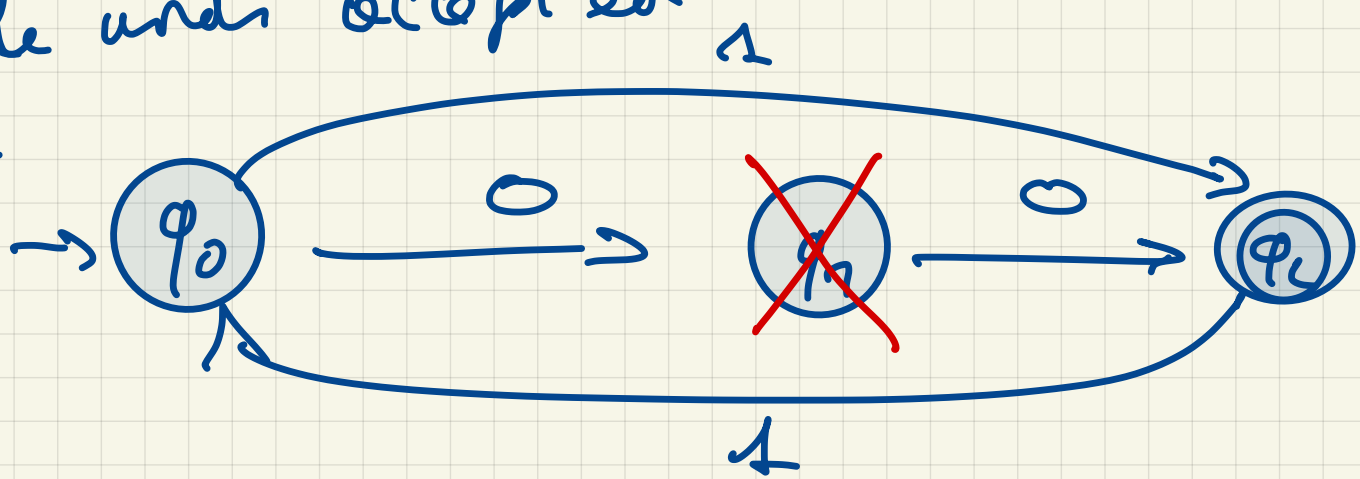
Esempio

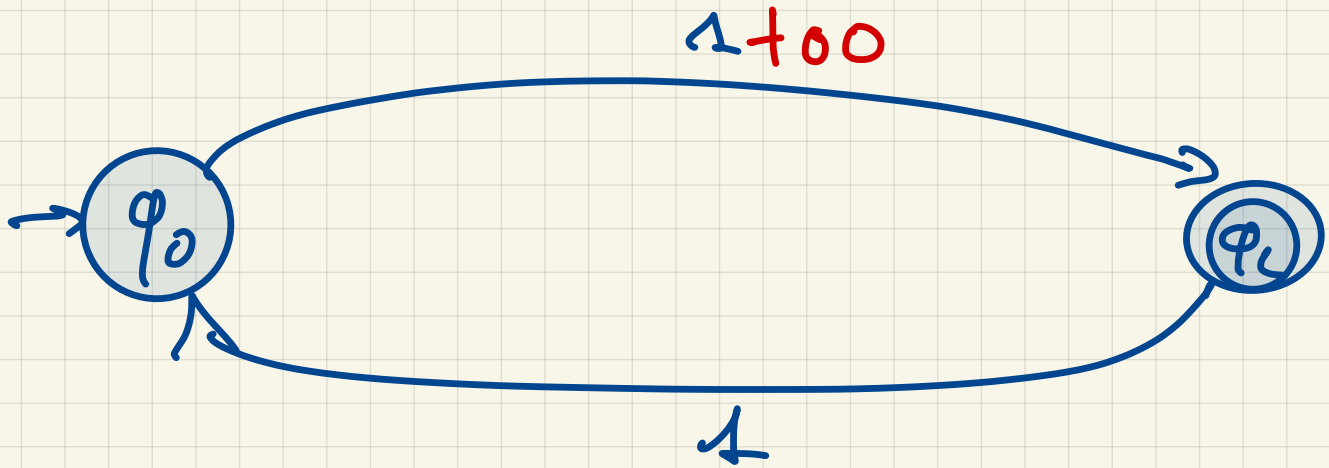


We will compute a first regular exp that accepts all the words which the automaton accepts in p_2.

Then a second one where ... the automaton accepts in p_3

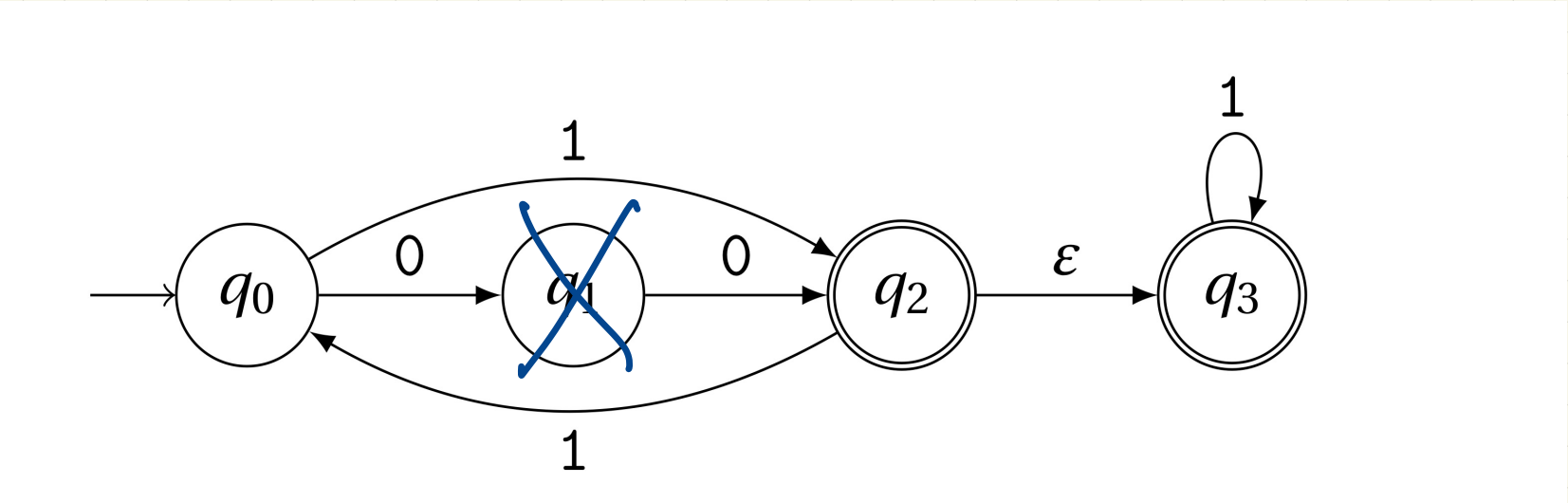
All the words accepted in q_2

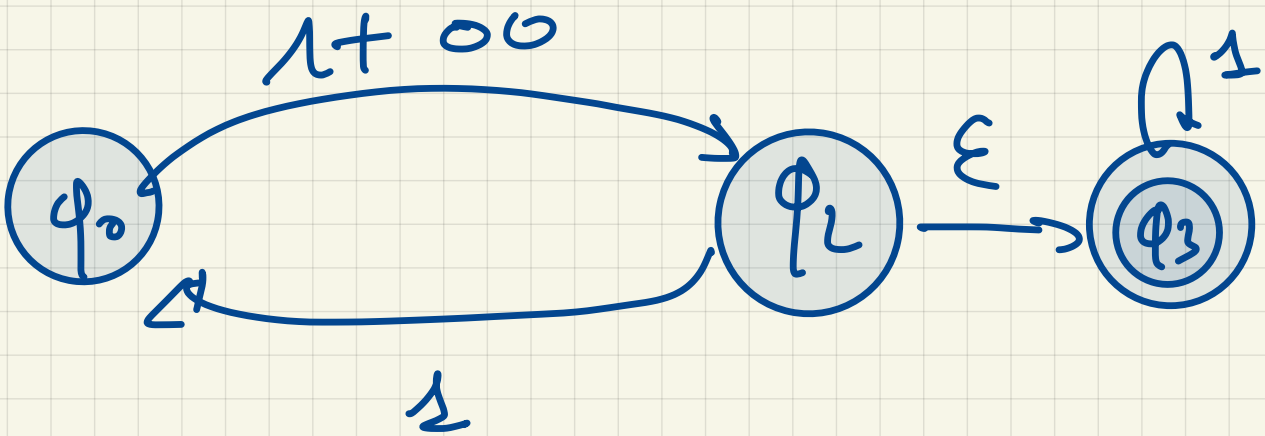


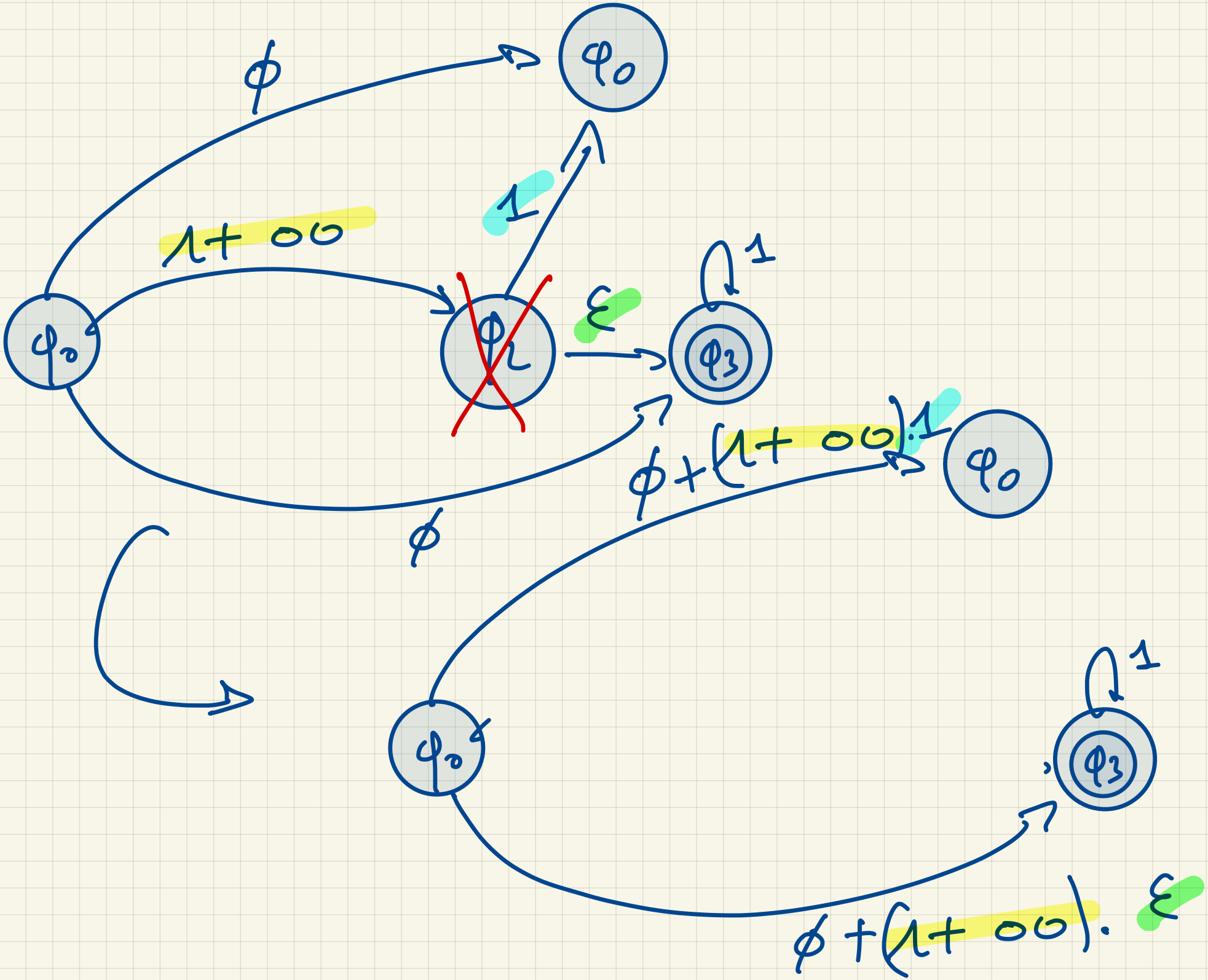


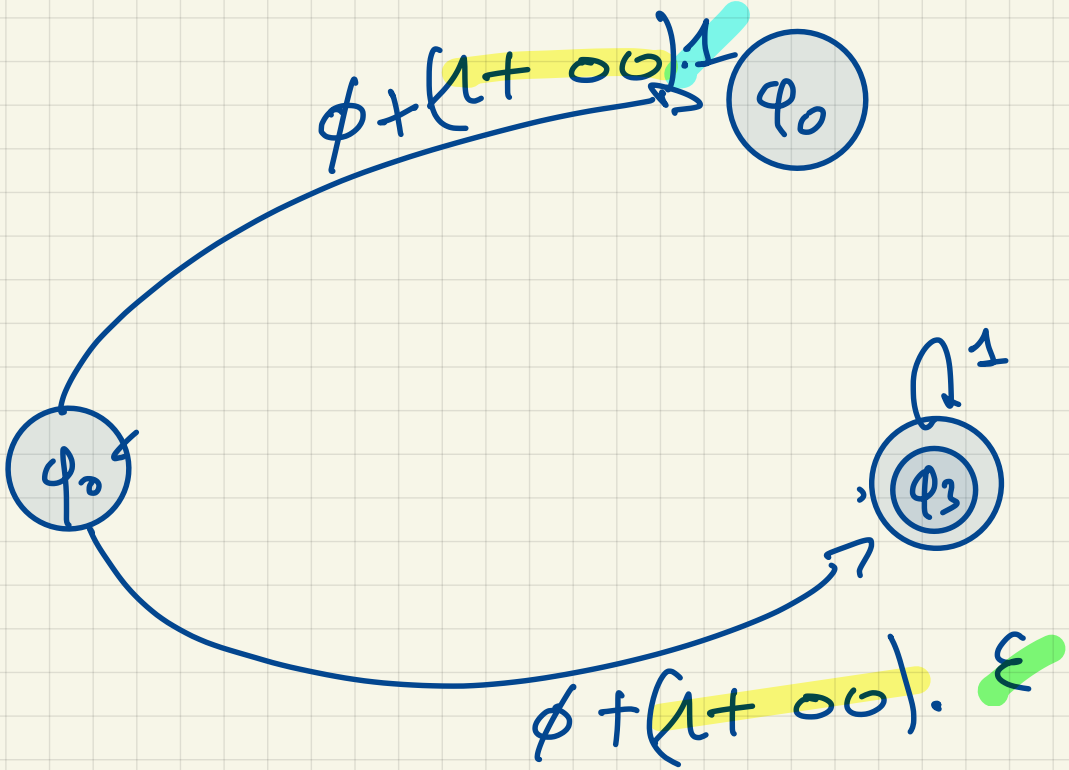
$$(11 + 001)^* (1 + 00)$$

All the words accepted in q_3.



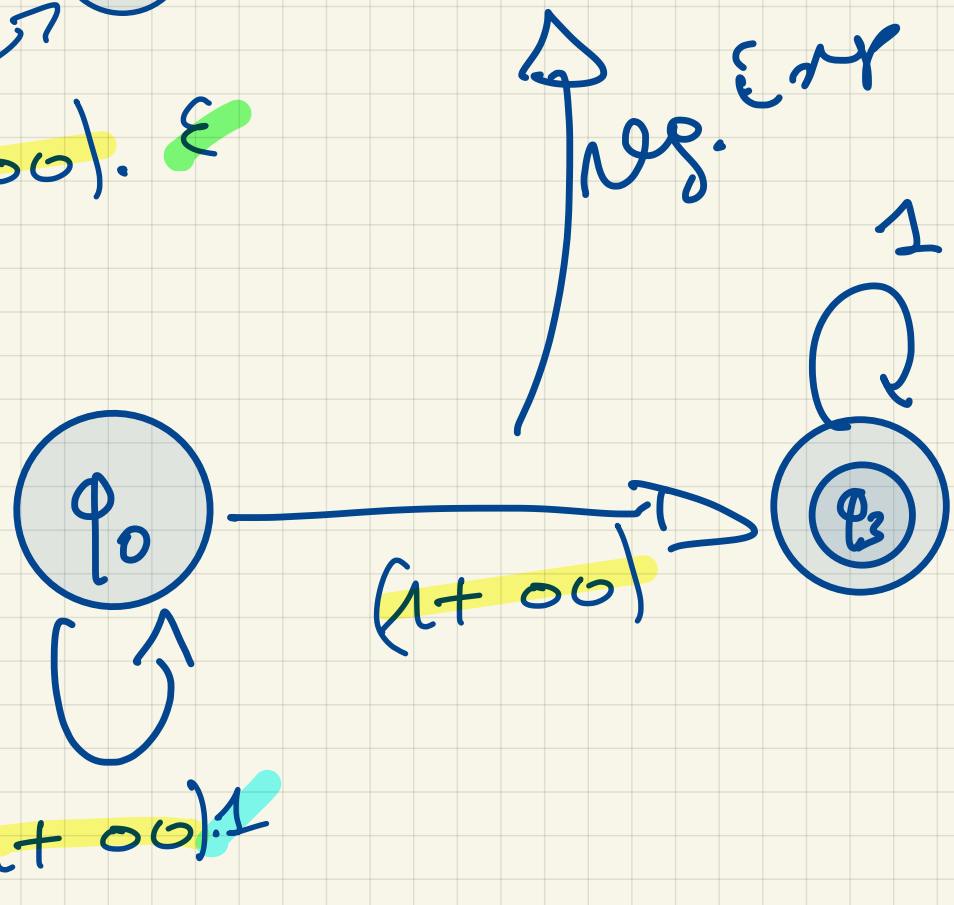



ϕ
q_0
1
$1+00$
q_0
q_2
ε
q_3
1
ϕ
$\phi+(1+00)\cdot 1$
q_0
q_0
q_3
1
$\phi+(1+00)\cdot\varepsilon$



$$\left((1+00)\cdot 1\right)^* (1+00)\, 1^*$$

Simplify



Final solution:

$$(11 + 001)^* (1 + 00)$$

$$+$$

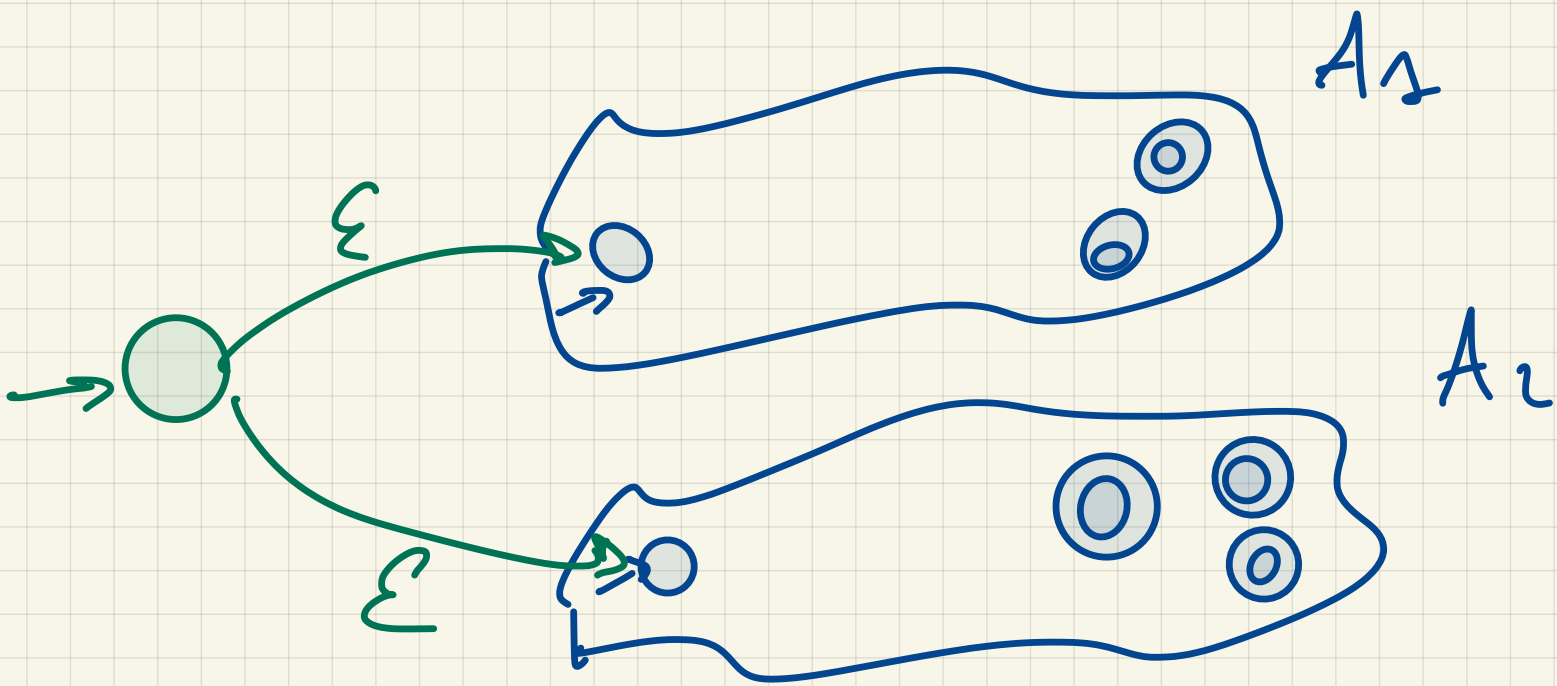
$$\left((1 + 00) \cdot 1\right)^* (1 + 00)\, 1^*$$

Operations on regular languages

Union

Given A_1, A_2, compute A

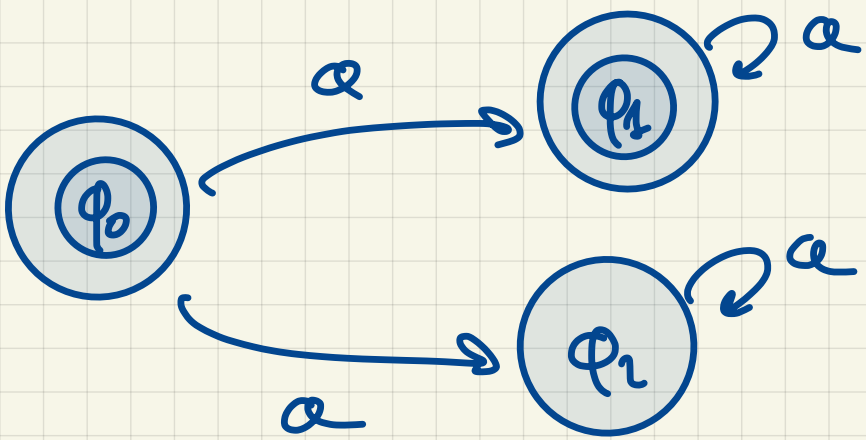
$$L(A) = L(A_1) \cup L(A_2)$$



Complement

Given A, Compute B s.t. $L(B) = \Sigma^* \setminus L(A)$.

First idea: replace all accepting states by non-accepting and vice-versa.



Doesn't work on ε-NFA and NFA!

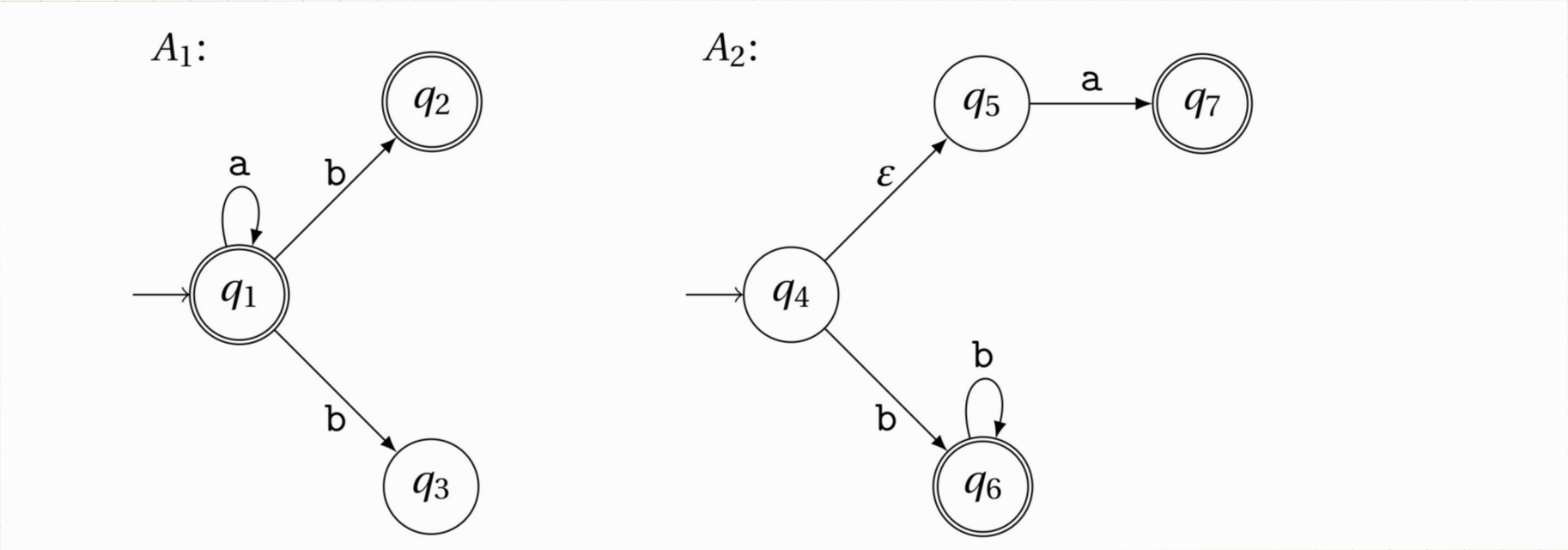
We need to determinise first!

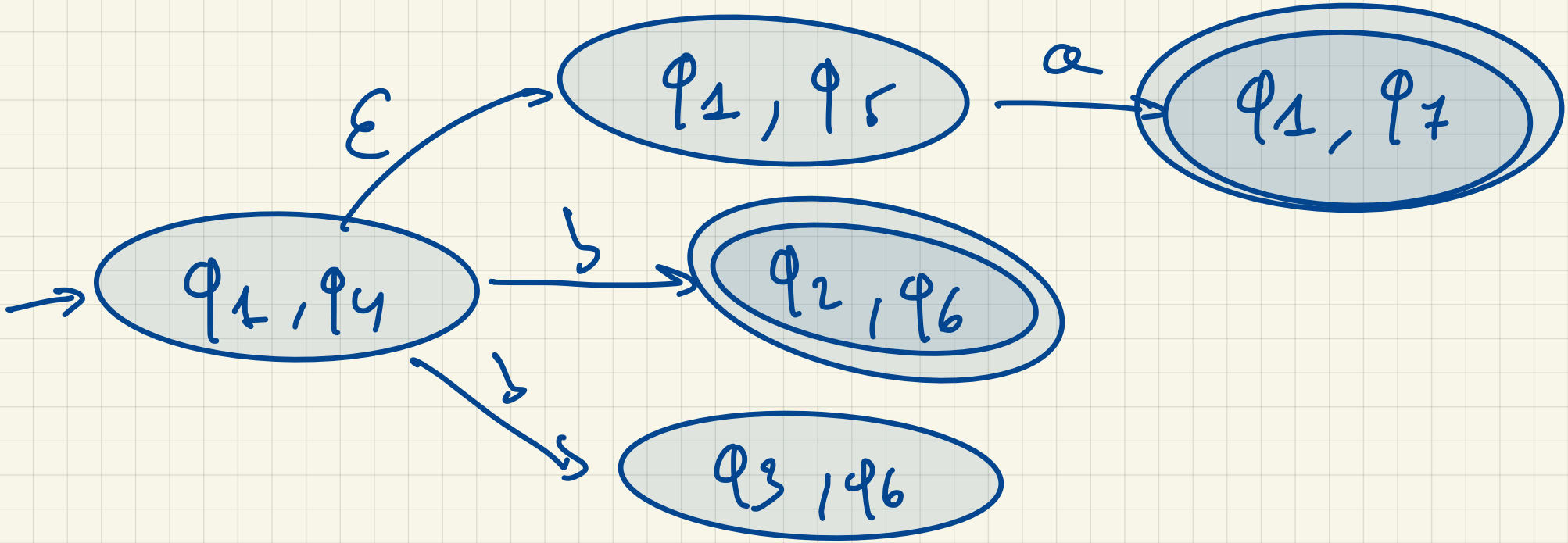
Intersection

Given A_1 & A_2

Compute B

$$L(B) = L(A_1) \cap L(A_2)$$


A_1:
q_1
q_2
q_3
a
b
b
A_2:
q_4
q_5
q_6
q_7
ε
a
b
b


q1, q4
q1, q5
q1, q7
q2, q6
q3, q6
ε
a
b
b

Language Inclusion

Given A_1 and A_2, check whether

$$L(A_1) \subseteq L(A_2).$$

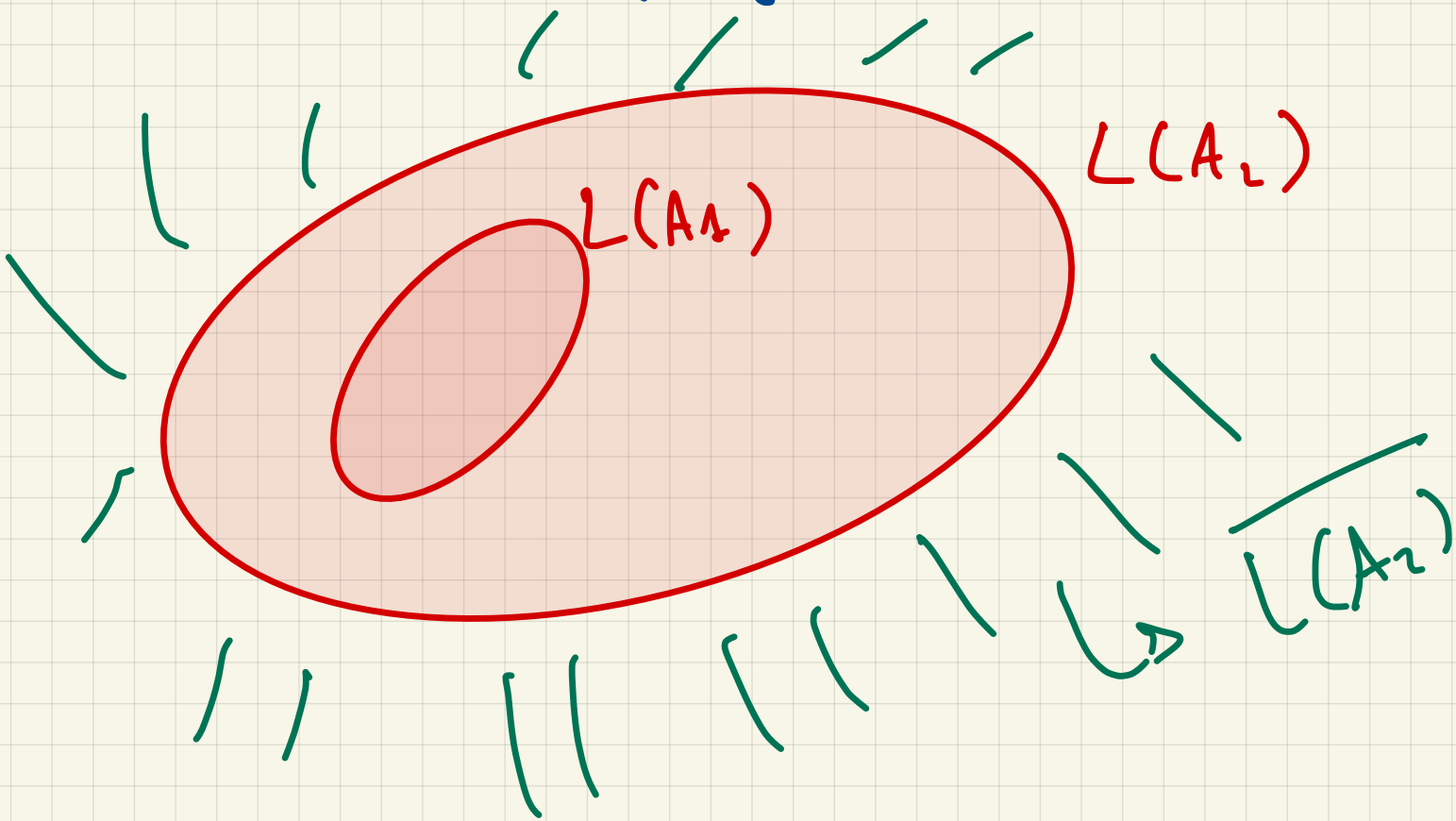
all behaviours of some system

Property = all the good behaviours.

$$L(A_1) \subseteq L(A_2)$$

iff

$$L(A_1) \cap \overline{L(A_2)} = \phi.$$



Some languages are not regular...

Let's have a look again at the language of well-parenthesised expressions.

$$L_{()} = \{ (), \ (()), \ (()()), \ldots \}$$

Let's show it is not regular!

Proof By contradiction:

We assume that $L_{()}$ is regular.

Since $L_{()}$ is regular, there is a DFA $A_{()}$ that accepts it, which has n states.

We consider the word

$$\underbrace{(\ldots(}_{n}\ \underbrace{)\ldots)}_{n} \in L_{()}$$

which is accepted by $A_{()}$.

Let's have a look at the path that accepts this word

$$\underbrace{q_0 \xrightarrow{(} q_1 \xrightarrow{(} \cdots \xrightarrow{(} q_n}\xrightarrow{)} q_{n+1} \cdots \xrightarrow{)} q \in F$$

I visit $n+1$ states but there are only n states in my automaton!!

At some point I visit twice the same state!!

$$q_0 \xrightarrow{(} q_1 \xrightarrow{(} \underbrace{q_i \xrightarrow{((((} q_j}_{q_1 = q_2} \xrightarrow{(} q_m \xrightarrow{)} q_{m+1} \cdots \xrightarrow{)} q \in F$$

$$q_0 \xrightarrow{(((} q_i \xrightarrow{((((} q_m \xrightarrow{)))))} q \in F$$

$$q_i = q_j \xrightarrow{((((} q_j$$

Then the yellow path also exists and accepts a word $\notin L(\,)$

Grammars

(1)	Exp	$\rightarrow$	Exp + Exp
(2)	Exp	$\rightarrow$	Exp $*$ Exp
(3)	Exp	$\rightarrow$	(Exp)
(4)	Exp	$\rightarrow$	Id
(5)	Exp	$\rightarrow$	Cst

Variables

Terminals (Σ)

Rewriting rules.

Derivation:

$$\text{Exp} \overset{2}{\Rightarrow} \text{Exp} * \text{Exp}$$
$$\overset{1}{\Rightarrow} \text{Exp} * \text{Exp} + \text{Exp}$$
$$\overset{4}{\Rightarrow} \text{Id} * \text{Exp} + \text{Exp}$$
$$\overset{5}{\Rightarrow} \text{Id} * \text{Cst} + \text{Exp} \overset{4}{\Rightarrow} \text{Id} * \text{Cst} + \text{Id}.$$