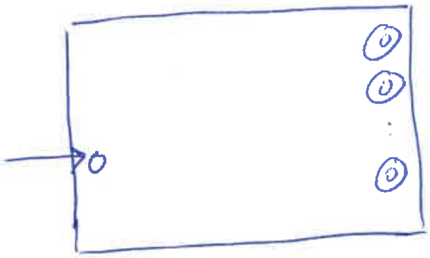
MATH 2301 — 05 Oct 2022

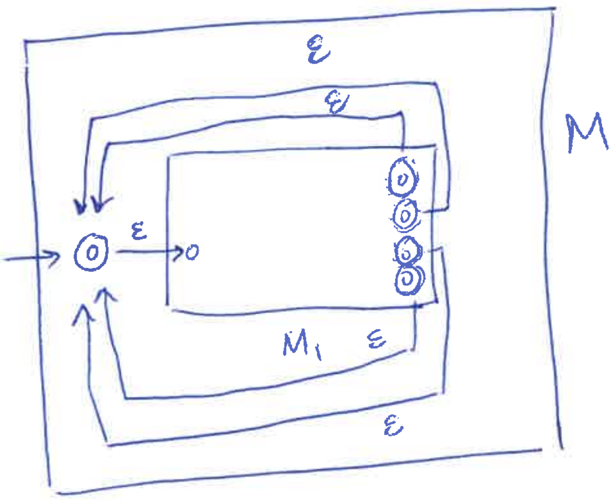
** Regex → NFA

Construction of NFA for $r = (r_1)^*$, given M_1 that recognises r_1



M_1

$L(M_1) = L(r_1)$

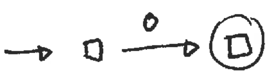


M accepts anything that successfully goes through M_1 0 or more times, ending up at one of the accept states marked.

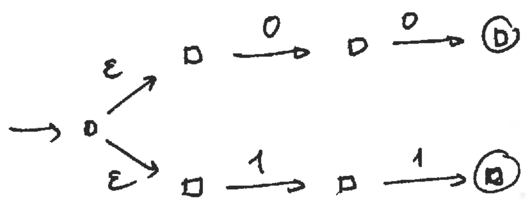
Example: $(00 \cup 11)0 = r$

← NFA recognising $r = 00$

← NFA recognising $r = 11$

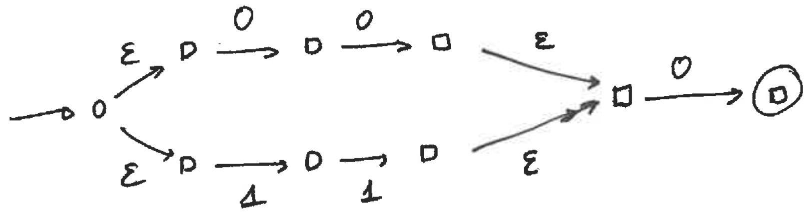


← NFA recognising $r = 0$



recognises $(00 \mid 11)$

Finally:



NFA recognises exactly $r = (00 \mid 11)0$.

Upshot: Given any regex r, there is an NFA M ~~the~~ such that $L(r) = L(M)$.

Q: What about DFAs?

Goal: Understand the relationship between what NFAs can do and what DFAs can do.

* Early observation:

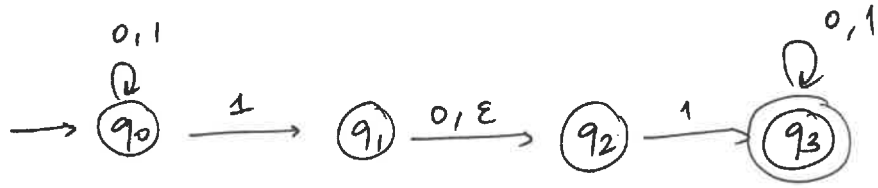
Proposition: Let M be an NFA which possibly has ε-arrows

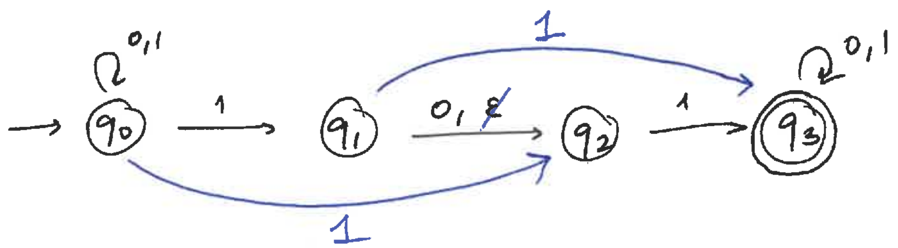
Then there is an equivalent NFA M' which has no ε-labelled arrows, i.e. $L(M') = L(M)$

Proof through an example (proof idea):

(Next page)

** Example





Idea: Any time there are arrows

$$A \xrightarrow{\varepsilon} B \xrightarrow{a} C \quad \text{or}$$

$$A \xrightarrow{a} B \xrightarrow{\varepsilon} C,$$ we add in a direct

arrow $A \xrightarrow{a} C$, and continue doing this.

At the end, we'll have an equivalent NFA without ε.

Note: The new machine M' has no ε-labels.
The transition function of M' can be thought of as a function

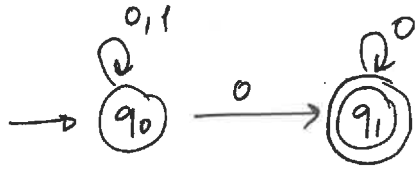
$$\delta : Q \times \Sigma \to \mathcal{P}(Q)$$

Theorem: Given any NFA M, there is an equivalent DFA M', such that $L(M) = L(M')$.

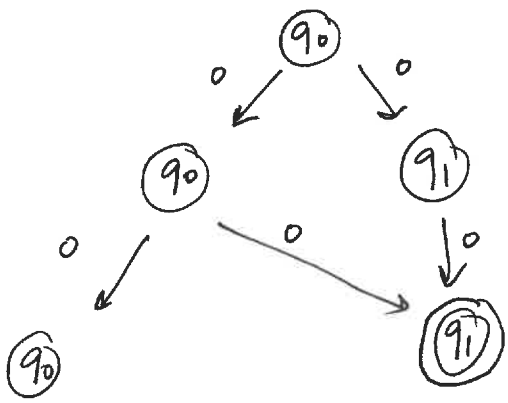
Pf (with an example)

Let M be an NFA. Let us assume further that M does not have any ε- arrows. (by previous Prop.)

Eg.



Example calculation tree for $w = 00$



ACCEPT.

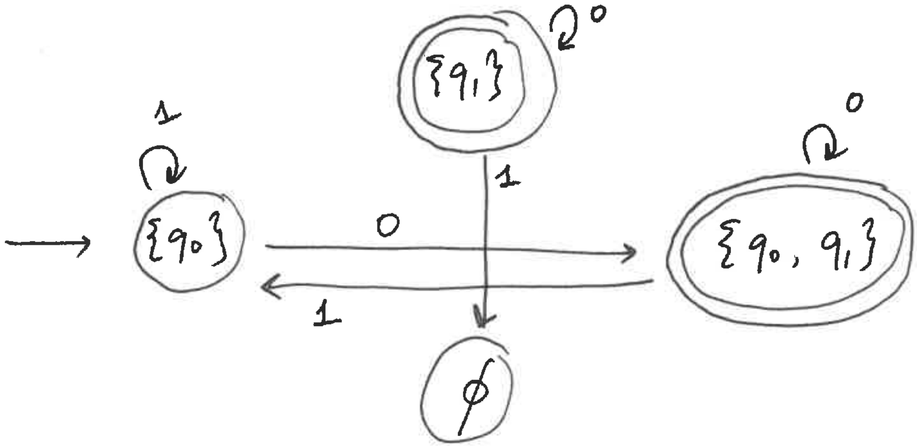
Calculation tree moves from a subset of Q to another subset of Q, after reading a single letter.

Let us construct an equivalent DFA. M'

- State set of $M' = \mathcal{P}(Q)$
  $[\{\phi, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}]$

- Start state $= \{q_0\}$

- Accepting states $= \{B \subseteq Q \mid B$ contains an accepting state of $M\}$



Transition function should "follow the calculation tree"

[More tomorrow ...]