Regular Languages

Kleene's Theorem

Definition. A language is **regular** iff it is equal to $L(M)$, the set of strings accepted by some deterministic finite automaton M.

Theorem.

(a) For any regular expression r, the set $L(r)$ of strings matching r is a regular language.

(b) Conversely, every regular language is the form $L(r)$ for some regular expression r.

Kleene Theorem, part (a) [p64]

Use Mathematical Induction to prove

$$\forall n.\ P(n)$$

where

$P(n) =$ $\boxed{\text{for all reg. exp. abstract syntax trees } r \text{ of size} \leq n\text{, there is an } NFA^{\varepsilon}\ M \text{ with } L(M) = L(r)}$

(Can use subset construction [p59] to get a DFA PM with $L(PM) = L(M) = L(r)$.)

Regular expressions (abstract syntax)

(concrete ")

The 'signature' for regular expression abstract syntax trees (over an alphabet Σ) consists of

- binary operators $r_1 \mid r_2$ *Union* and $r_1 r_2$ *Concat*
- unary operator *Star* r^*
- nullary operators (constants) ε *Null*, $\emptyset$ *Empty* and a Sym_a (one for each $a \in \Sigma$).

(i) **Base cases:** show that $\{a\}$, $\{\varepsilon\}$ and $\emptyset$ are regular languages.

(ii) **Induction step for $r_1|r_2$:** given NFA$^\varepsilon$s M_1 and M_2, construct an NFA$^\varepsilon$ $Union(M_1, M_2)$ satisfying

$$\boxed{L(Union(M_1, M_2)) = \{u \mid u \in L(M_1) \vee u \in L(M_2)\}}$$

Thus if $L(r_1) = L(M_1)$ and $L(r_2) = L(M_2)$, then $L(r_1|r_2) = L(Union(M_1, M_2))$.

(i) **Base cases:** show that $\{a\}$, $\{\varepsilon\}$ and $\varnothing$ are regular languages.

(ii) **Induction step for $r_1|r_2$:** given NFA$^\varepsilon$s M_1 and M_2, construct an NFA$^\varepsilon$ $Union(M_1, M_2)$ satisfying

$$\boxed{L(Union(M_1, M_2)) = \{u \mid u \in L(M_1) \vee u \in L(M_2)\}}$$

Thus if $L(r_1) = L(M_1)$ and $L(r_2) = L(M_2)$, then $L(r_1|r_2) = L(Union(M_1, M_2))$.

(iii) **Induction step for r_1r_2:** given NFA$^\varepsilon$s M_1 and M_2, construct an NFA$^\varepsilon$ $Concat(M_1, M_2)$ satisfying

$$\boxed{L(Concat(M_1, M_2)) = \{u_1u_2 \mid u_1 \in L(M_1) \,\&\, u_2 \in L(M_2)\}}$$

Thus $L(r_1r_2) = L(Concat(M_1, M_2))$ when $L(r_1) = L(M_1)$ and $L(r_2) = L(M_2)$.

(i) **Base cases:** show that $\{a\}$, $\{\varepsilon\}$ and $\emptyset$ are regular languages.

(ii) **Induction step for $r_1|r_2$:** given NFA$^\varepsilon$s M_1 and M_2, construct an NFA$^\varepsilon$ $Union(M_1, M_2)$ satisfying

$$\boxed{L(Union(M_1, M_2)) = \{u \mid u \in L(M_1) \vee u \in L(M_2)\}}$$

Thus if $L(r_1) = L(M_1)$ and $L(r_2) = L(M_2)$, then $L(r_1|r_2) = L(Union(M_1, M_2))$.

(iii) **Induction step for r_1r_2:** given NFA$^\varepsilon$s M_1 and M_2, construct an NFA$^\varepsilon$ $Concat(M_1, M_2)$ satisfying

$$\boxed{L(Concat(M_1, M_2)) = \{u_1u_2 \mid u_1 \in L(M_1) \;\&\; u_2 \in L(M_2)\}}$$

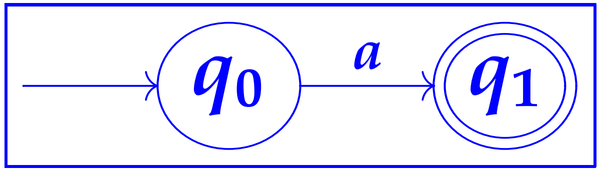
Thus $L(r_1r_2) = L(Concat(M_1, M_2))$ when $L(r_1) = L(M_1)$ and $L(r_2) = L(M_2)$.

(iv) **Induction step for r^*:** given NFA$^\varepsilon$ M, construct an NFA$^\varepsilon$ $Star(M)$ satisfying

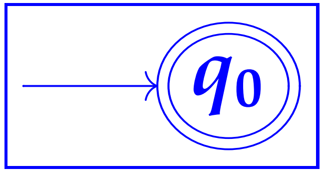
$$\boxed{L(Star(M)) = \{u_1u_2 \ldots u_n \mid n \geq 0 \text{ and each } u_i \in L(M)\}}$$

Thus $L(r^*) = L(Star(M))$ when $L(r) = L(M)$.

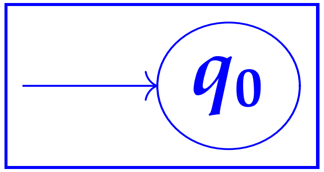
NFAs for regular expressions a, ϵ, $\emptyset$



just accepts the one-symbol string a

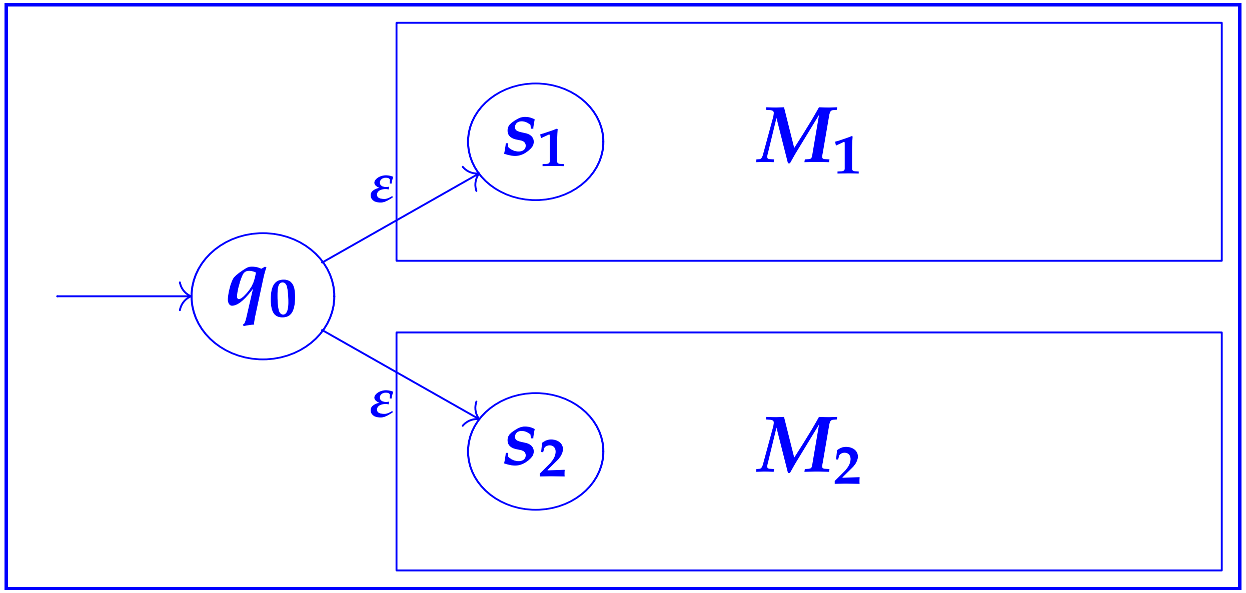


just accepts the null string, ε



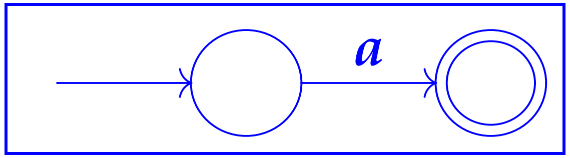
accepts no strings

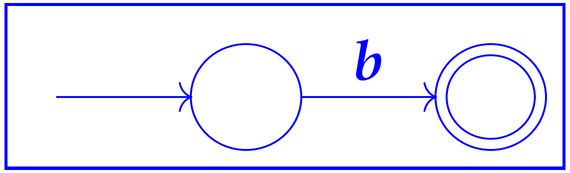
$Union(M_1, M_2)$

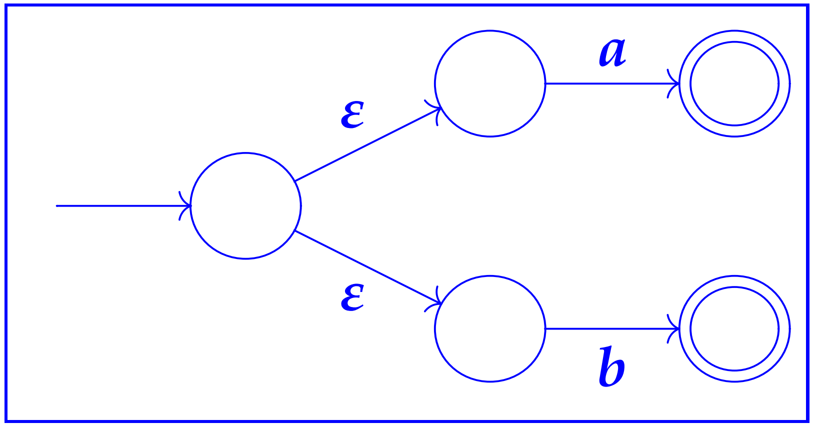


accepting states = union of accepting states of M_1 and M_2

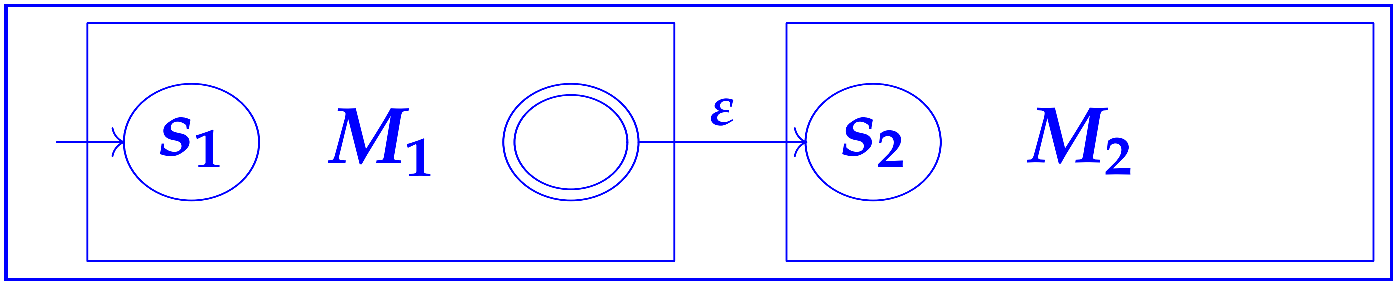
For example,

if $M_a =$ 

and $M_b =$ 

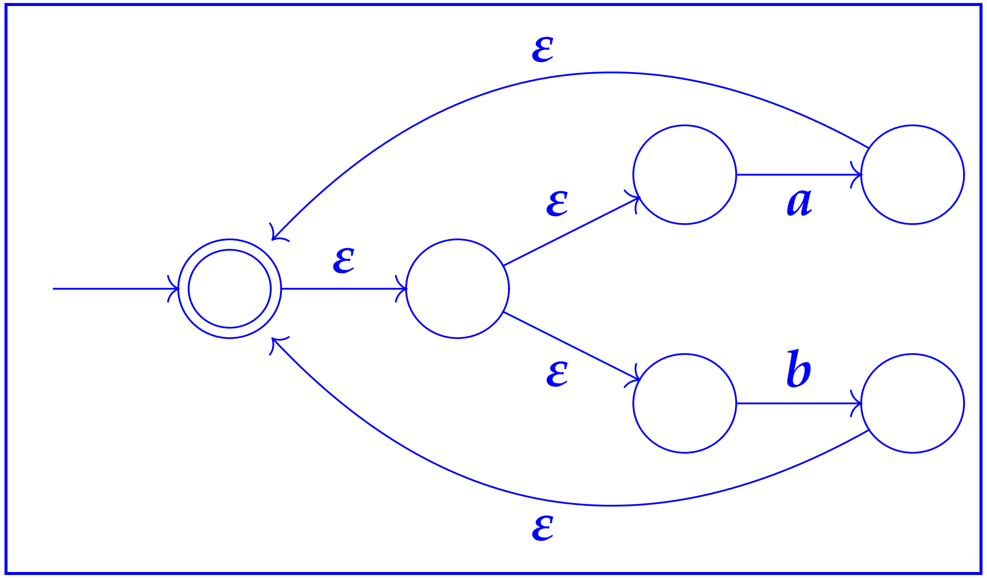
then $Union(M_a, M_b) =$ 

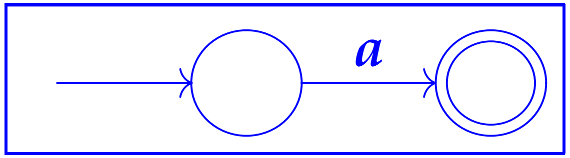
$Concat(M_1, M_2)$

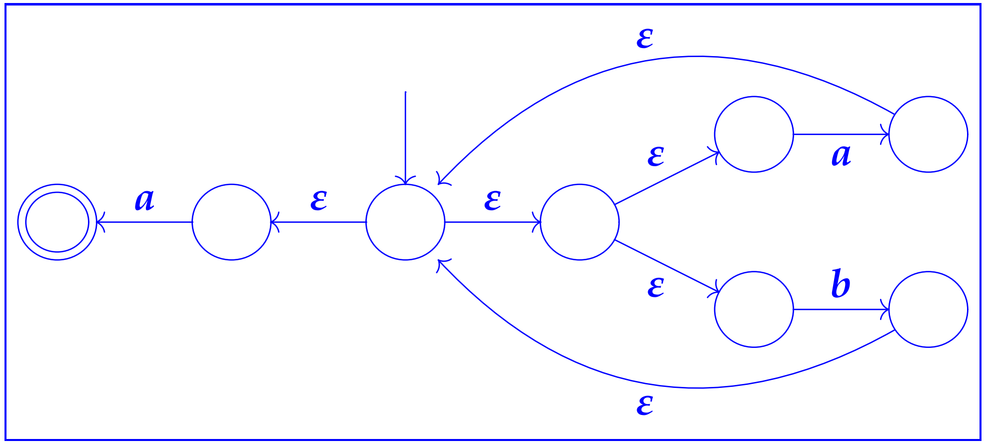


accepting states are those of M_2

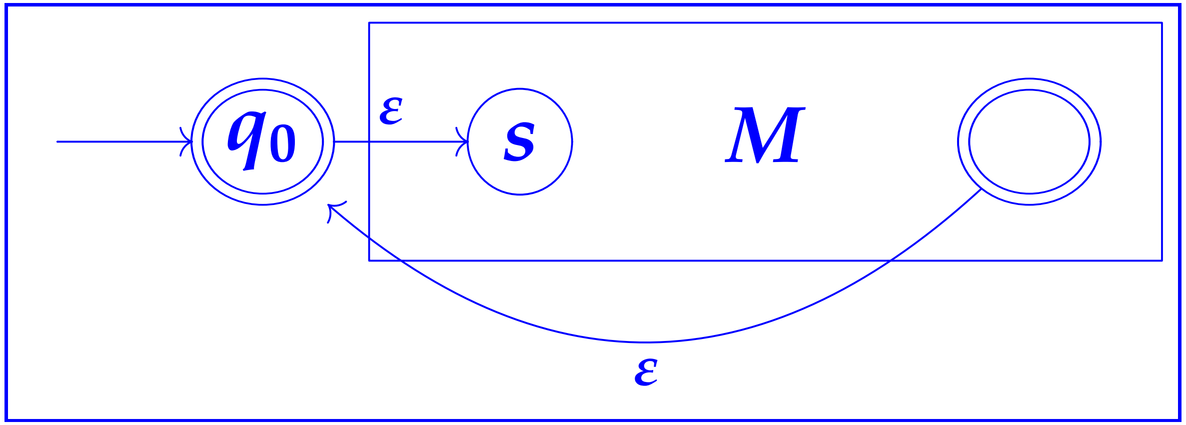
For example,

if $M_1 =$ 

and $M_2 =$ 

then $Concat(M_1, M_2) =$ 

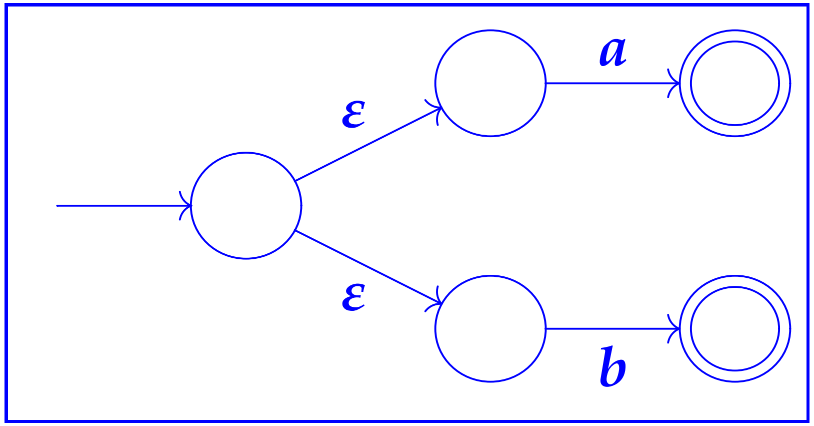
$Star(M)$

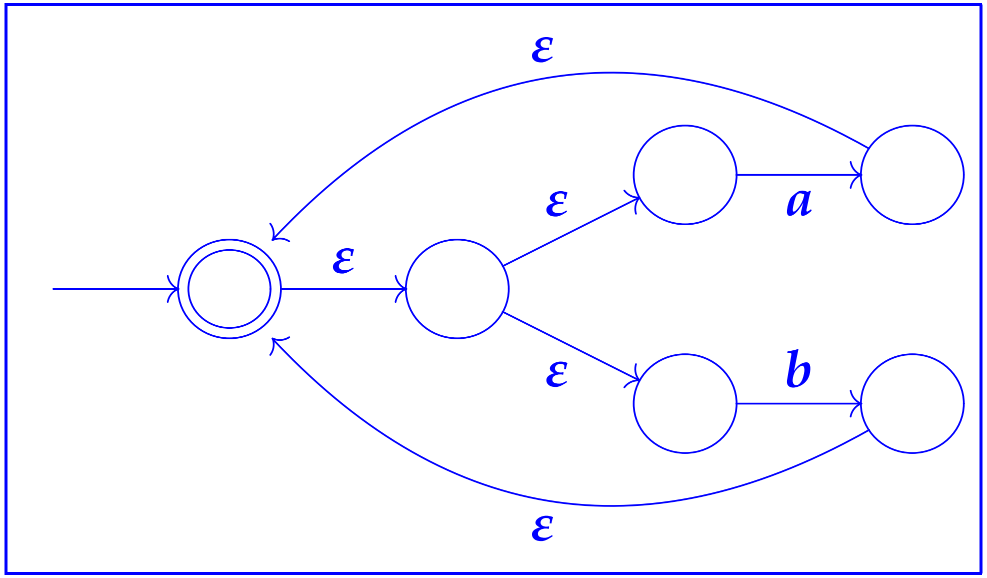


the only accepting state of $Star(M)$ is q_0

(N.B. doing without q_0 by just looping back to s
and making that accepting won't work – Exercise 4.1.)

For example,

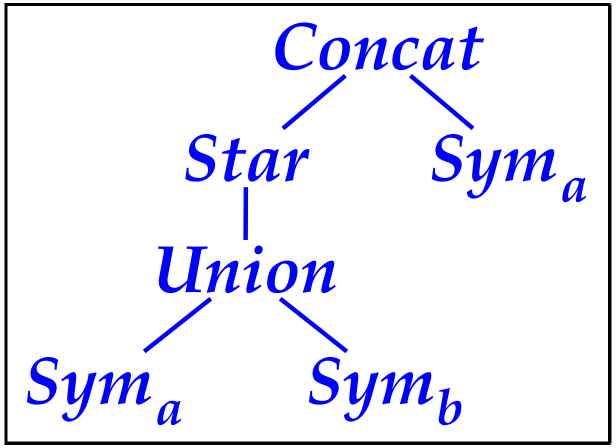
if $M =$ 

then $Star(M) =$ 

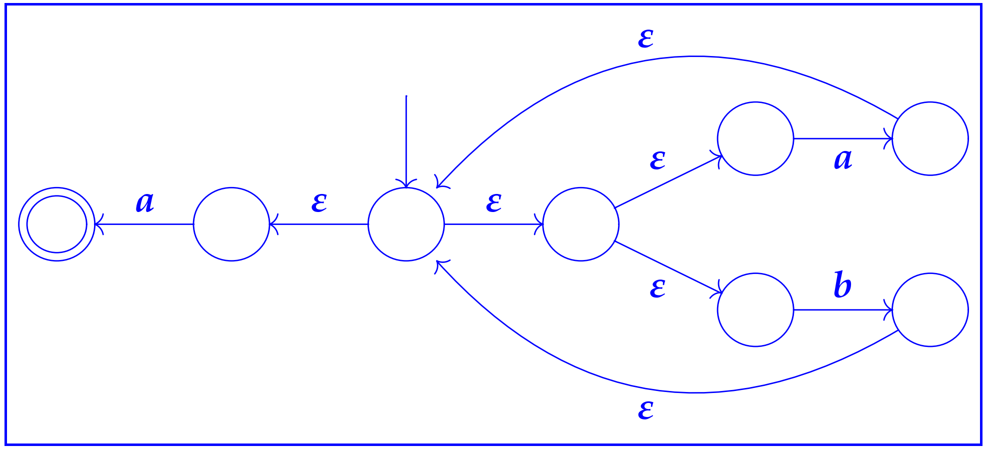
Example

Regular expression $(a|b)^*a$

whose abstract syntax tree is



is mapped to the NFA$^\varepsilon$ $Concat(Star(Union(M_a, M_b)), M_a) =$



(*cf.* Slides 68, 71 and 74).

Some questions

(a) Is there an algorithm which, given a string u and a regular expression r, computes whether or not u matches r?

(b) In formulating the definition of regular expressions, have we missed out some practically useful notions of pattern?

(c) Is there an algorithm which, given two regular expressions r and s, computes whether or not they are **equivalent**, in the sense that $L(r)$ and $L(s)$ are equal sets?

(d) Is every language (subset of Σ^*) of the form $L(r)$ for some r?

Decidability of matching

We now have a positive answer to question (a) on Slide 38.
Given string u and regular expression r:

- construct an NFA$^{\varepsilon}$ M satisfying $L(M) = L(r)$;
- in PM (the DFA obtained by the subset construction, Slide 59) carry out the sequence of transitions corresponding to u from the start state to some state q (because PM is deterministic, there is a unique such transition sequence);
- check whether q is accepting or not: if it is, then $u \in L(PM) = L(M) = L(r)$, so u matches r; otherwise $u \notin L(PM) = L(M) = L(r)$, so u does not match r.

(The subset construction produces an exponential blow-up of the number of states: PM has 2^n states if M has n. This makes the method described above potentially inefficient – more efficient algorithms exist that don't construct the whole of PM.)