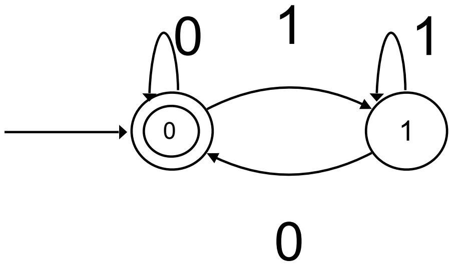
NFA and regex

cl

- ε-transitions
- regular expressions

Two examples

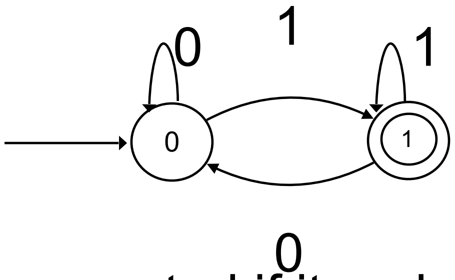
	×2	×2 + 1
	0	**1**
0	0	1
1	0	1



Even binary numbers

Input sequence is accepted if it ends with a zero.

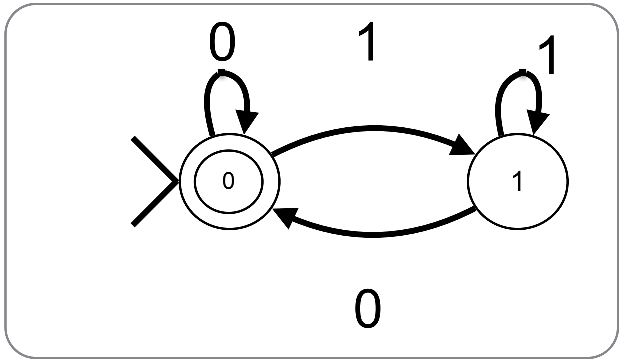
	×2	×2 + 1
	0	**1**
0	0	1
1	0	1



Odd binary numbers

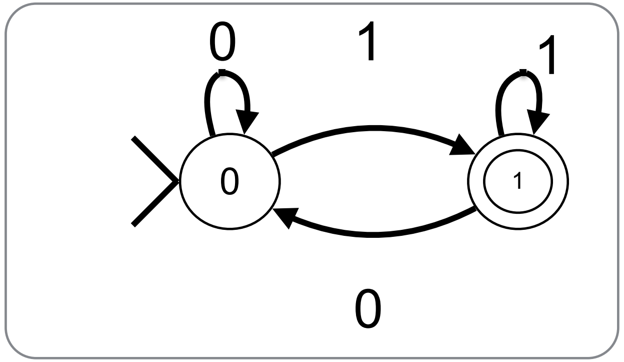
Input sequence is accepted if it ends with a one.

The complement of a DFA regular language is DFA regular



L_0 : even numbers

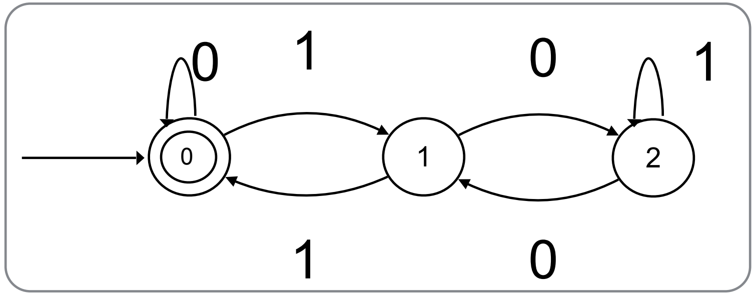
$= 0 \bmod 2$



L_1 : odd numbers

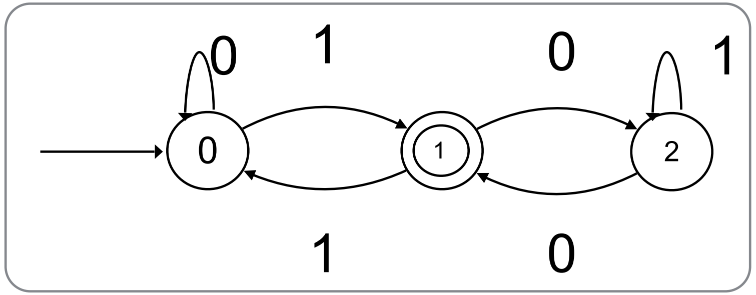
$= 1 \bmod 2$

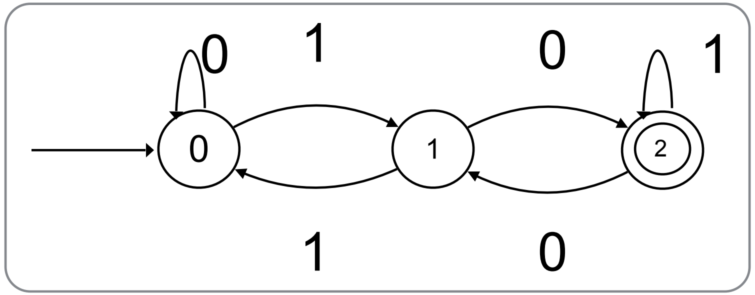
Three examples



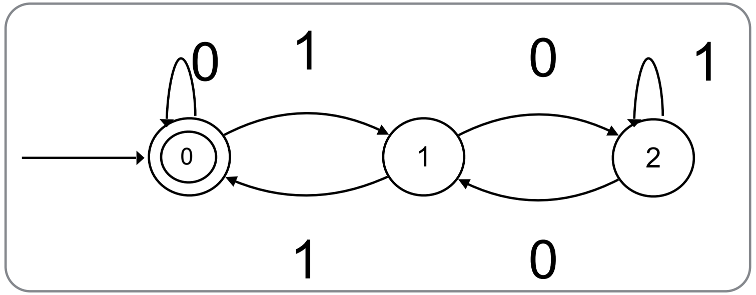
Which binary numbers are accepted?

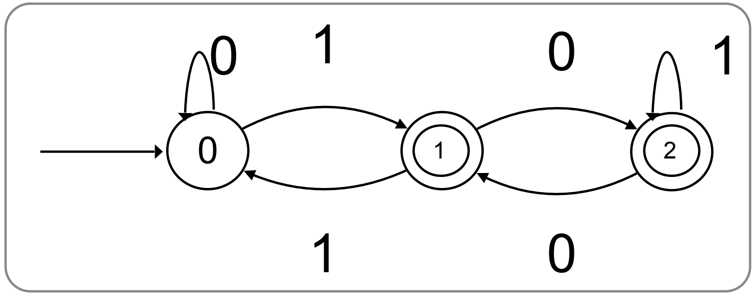
	$\times 2$	$\times 2 + 1$
mod 3	**0**	**1**
0	0	1
1	2	0
2	1	2





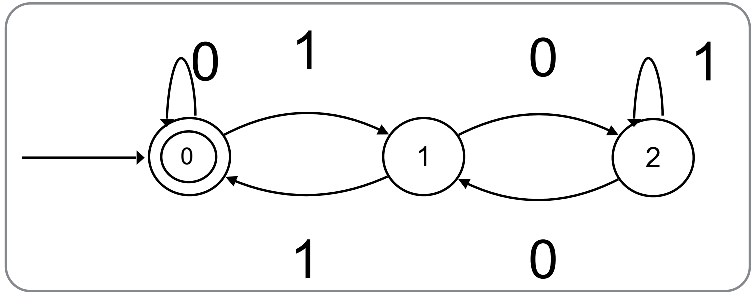
The complement of a DFA regular language is DFA regular



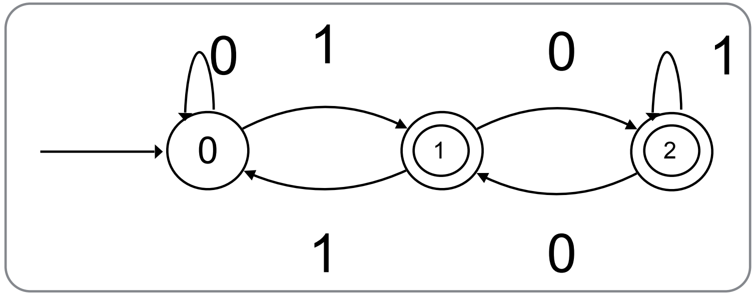


If $A \subseteq \Sigma^*$ is recognised by M then $\bar{\mathbf{A}} = \Sigma^* \setminus A$ is recognised by $\bar{\mathbf{M}}$ where $\bar{\mathbf{M}}$ and **M** are identical except that the accepting states of $\bar{\mathbf{M}}$ are the non-accepting states of **M** and vice-versa

By three or not by three?

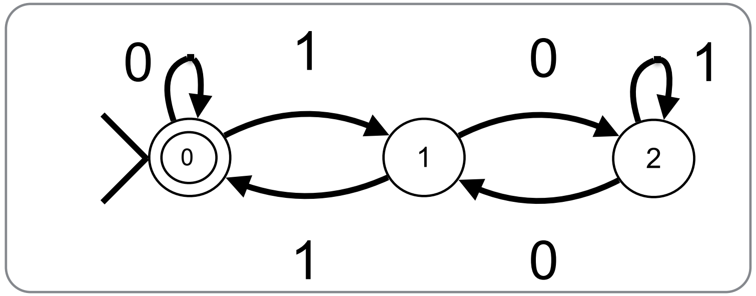


divisible by three



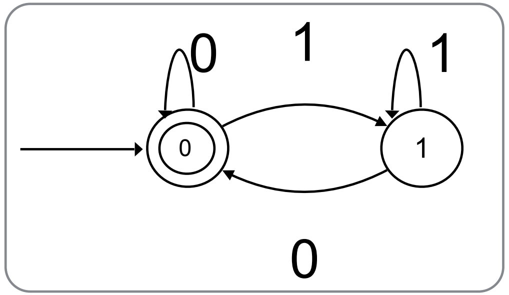
not
divisible by three

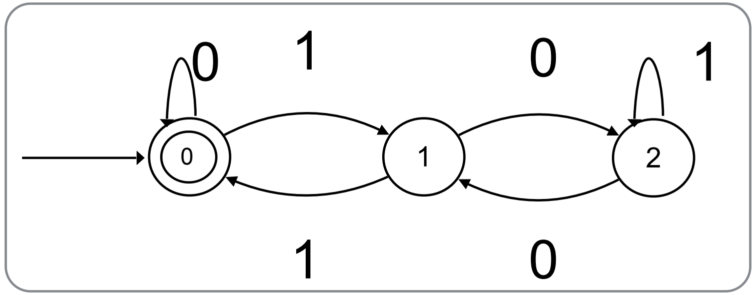
The intersection of two DFA regular languages is DFA regular



$L_0 = 0 \bmod 3$
$L_1 = 1 \bmod 3$
$L_2 = 2 \bmod 3$

The intersection of two DFA regular languages is DFA regular





divisible by 6

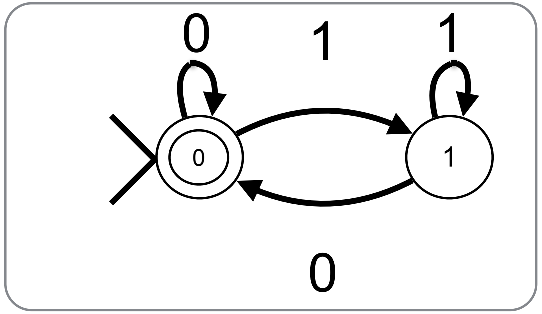
≡

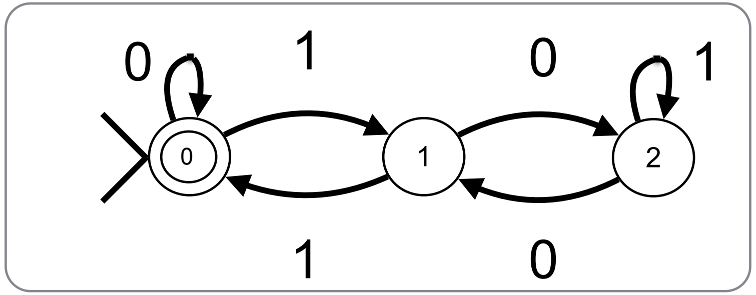
divisible by 2

and

divisible by 3

The intersection of two DFA-regular languages is DFA-regular



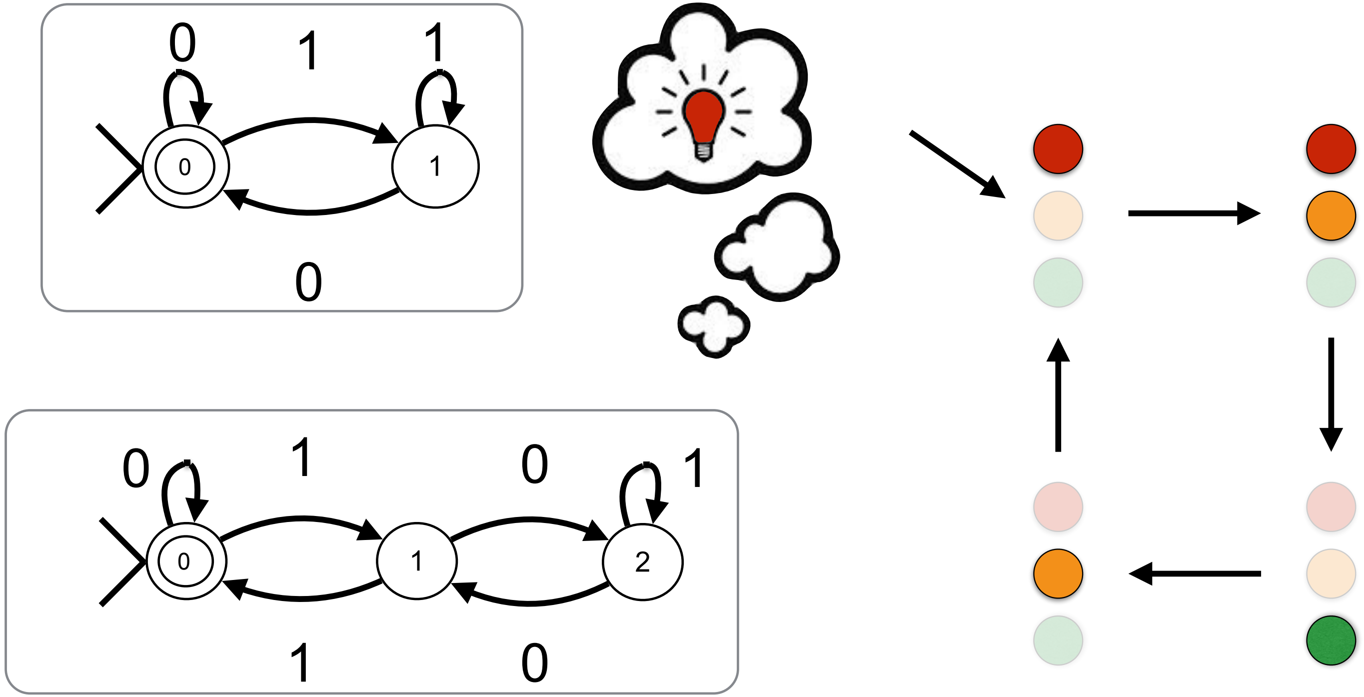


Run both machines in parallel?

Build one machine that simulates two machines running in parallel!

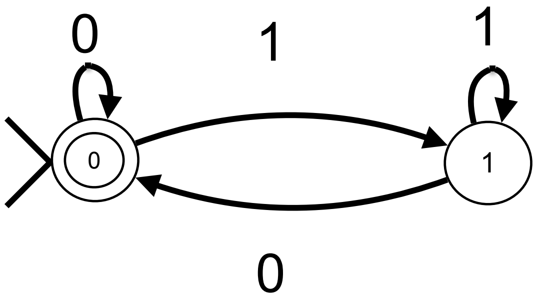
Keep track of the state of each machine.

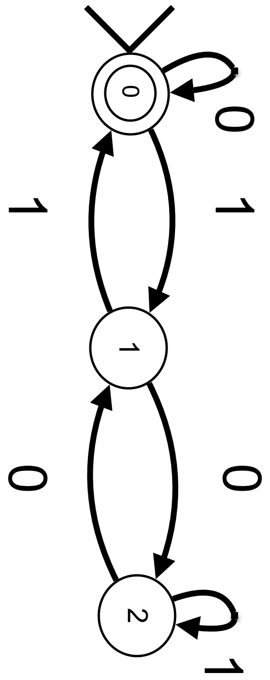
The intersection of two DFA-regular languages is DFA-regular

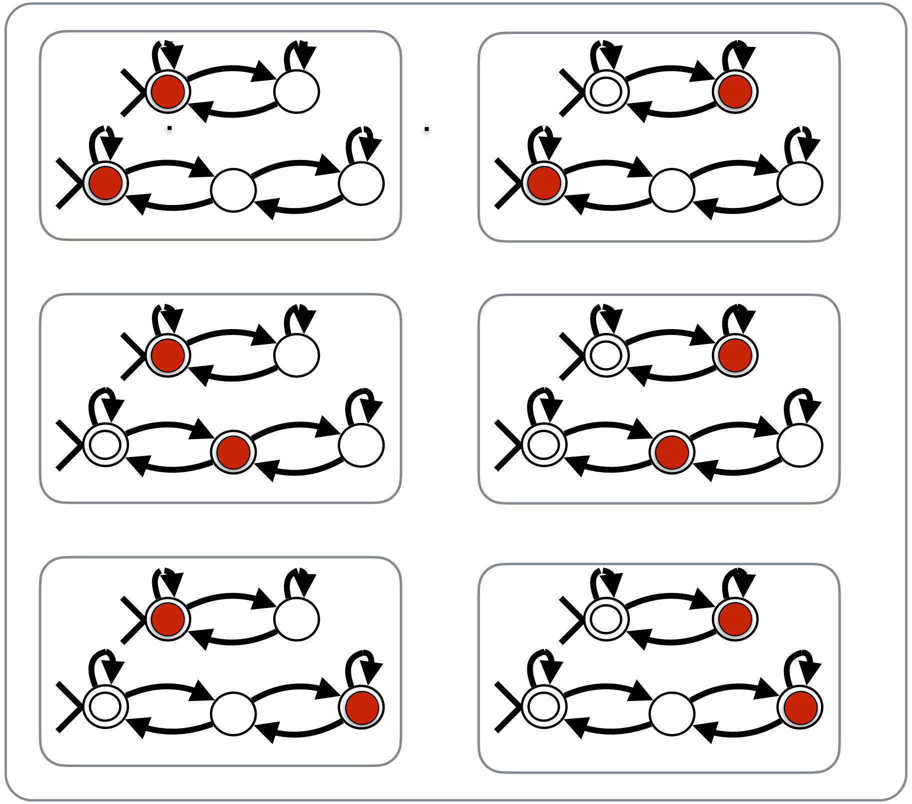


intersection of languages

run the two machines in parallel
when a string is in both languages,
both are in an accepting state

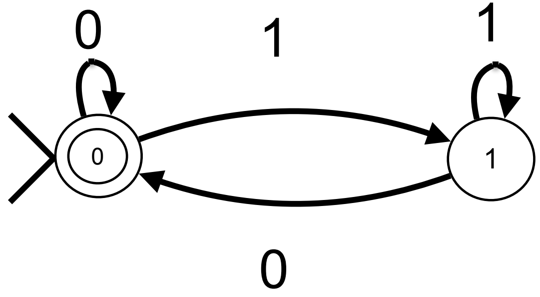


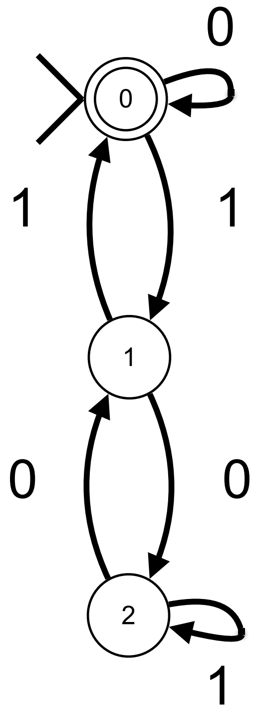


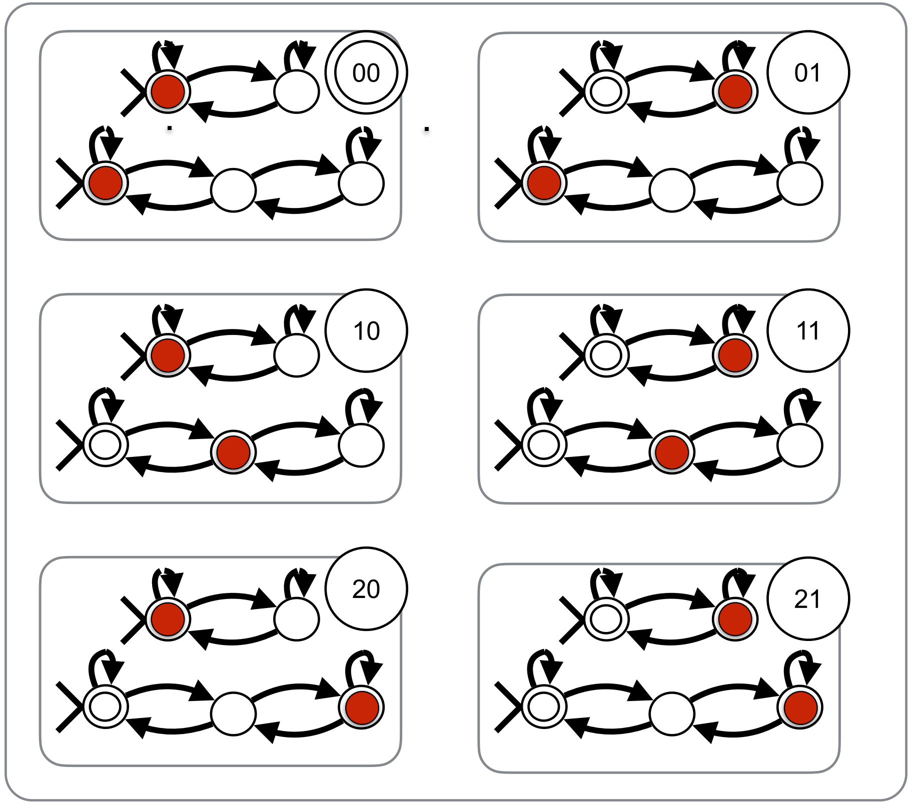


intersection of languages

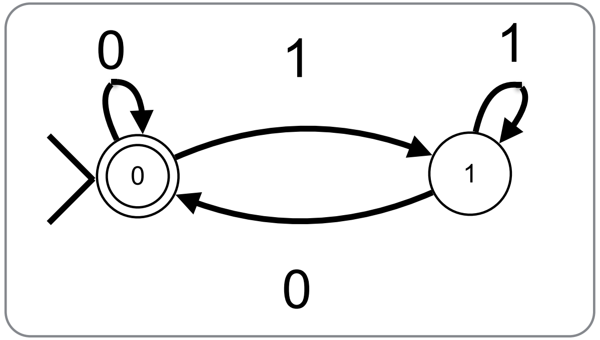
run the two machines in parallel
when a string is in both languages,
both are in an accepting state



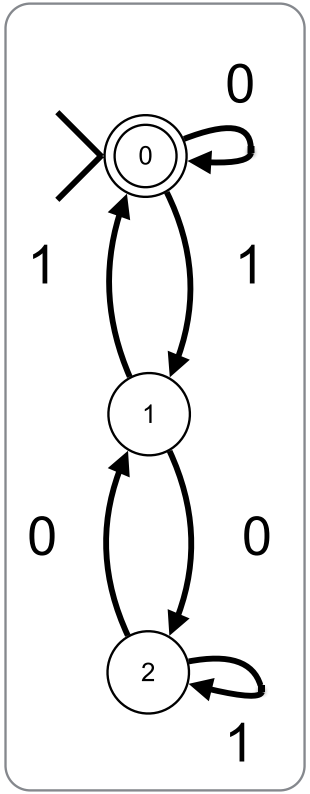


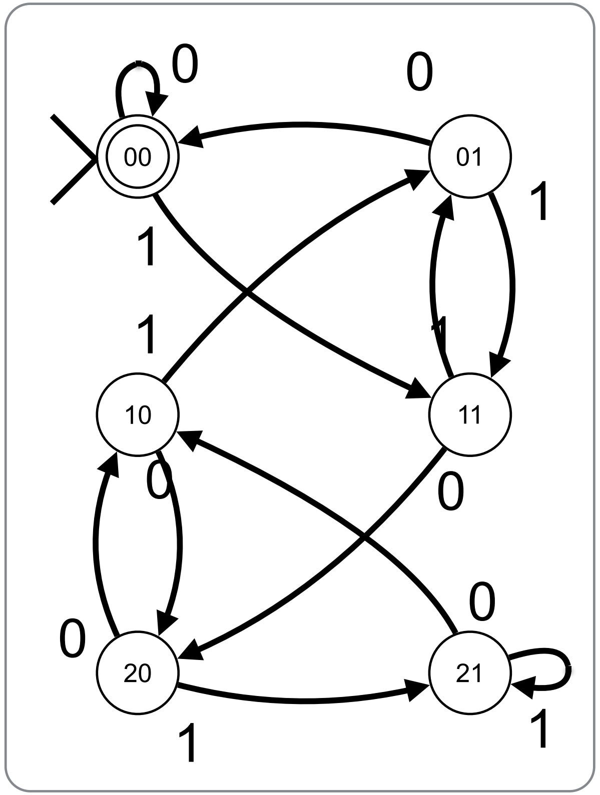


intersection of two regular languages is regular



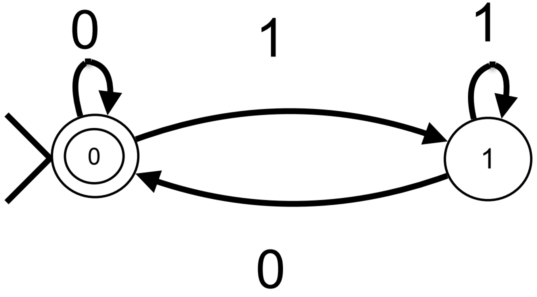
run two machines in synchrony

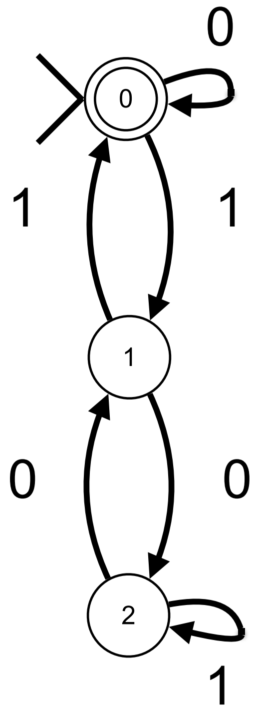


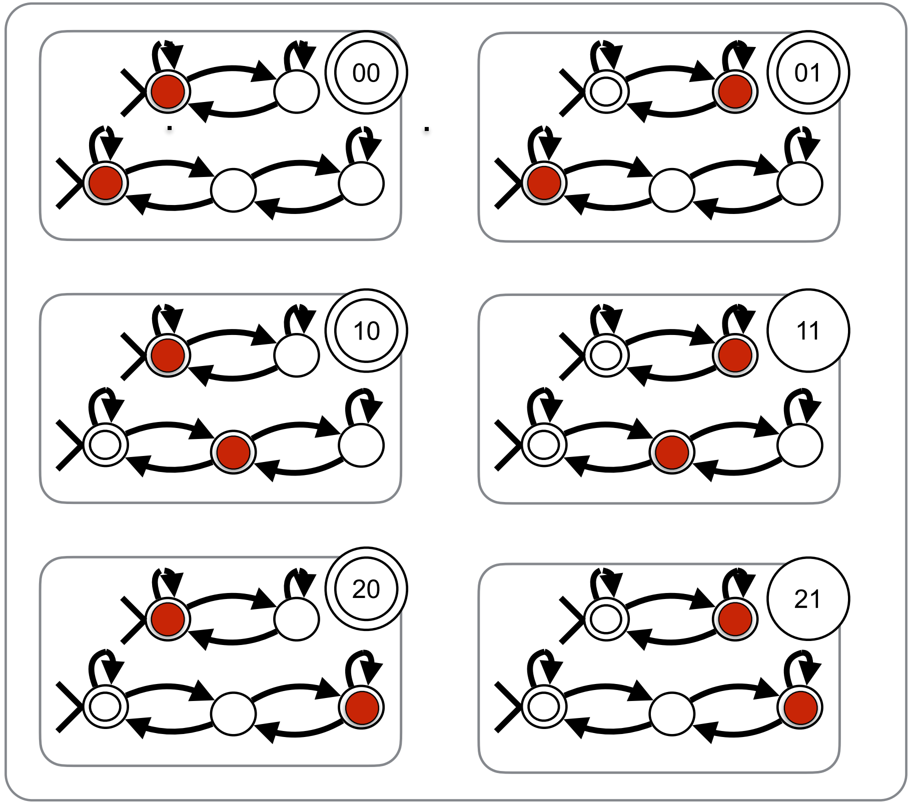


union of languages

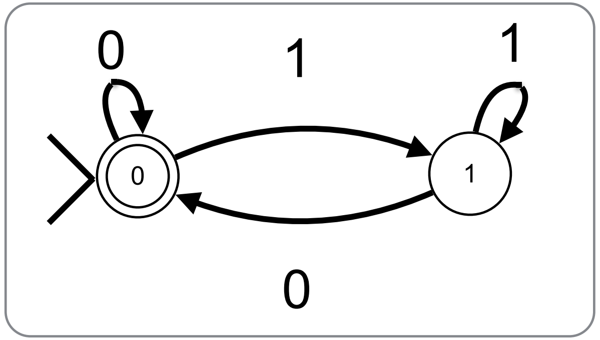
run the two machines in parallel
when a string is in the union of the
two languages, either or both are in
an accepting state



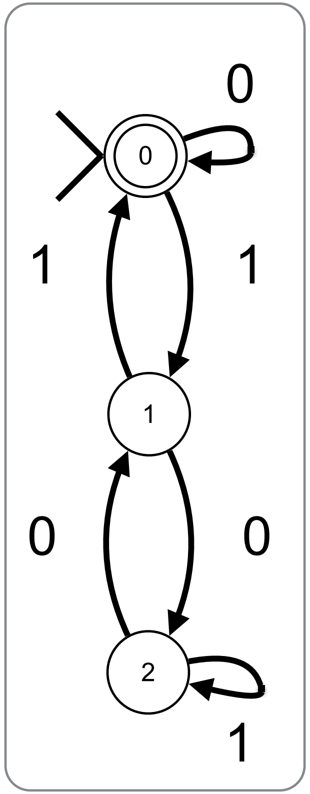


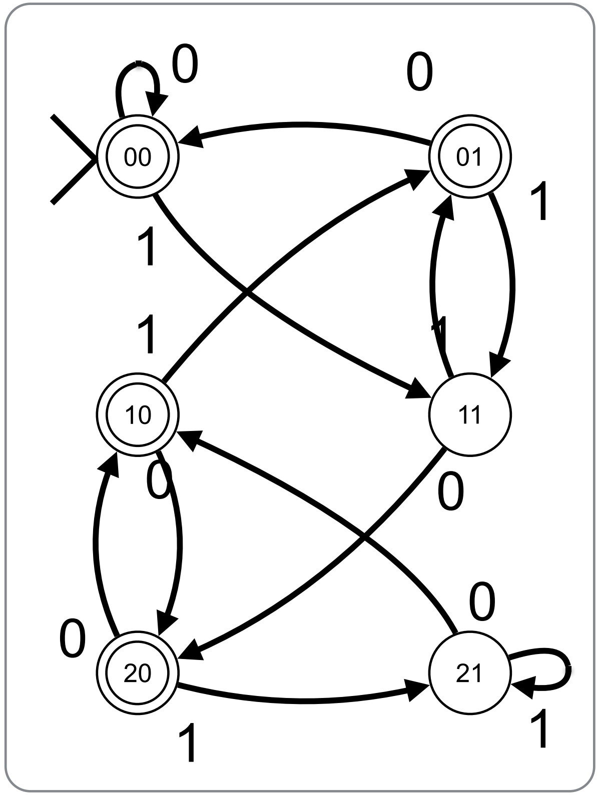


union of two
regular languages
is regular



run two
machines
in
synchrony





The DFA-regular languages $A \subseteq \Sigma^*$ form a Boolean Algebra

- Since they are closed under intersection and complement.

The DFA-regular languages $A \subseteq \Sigma^*$ form a Boolean Algebra

Are the DFA-regular languages closed under concatenation R S   and   iteration ( )* ?,

we define non-deterministic NFA

— FSM with ε-transitions —

and show that:

for each regex p there is an NFA that accepts exactly the strings matching p

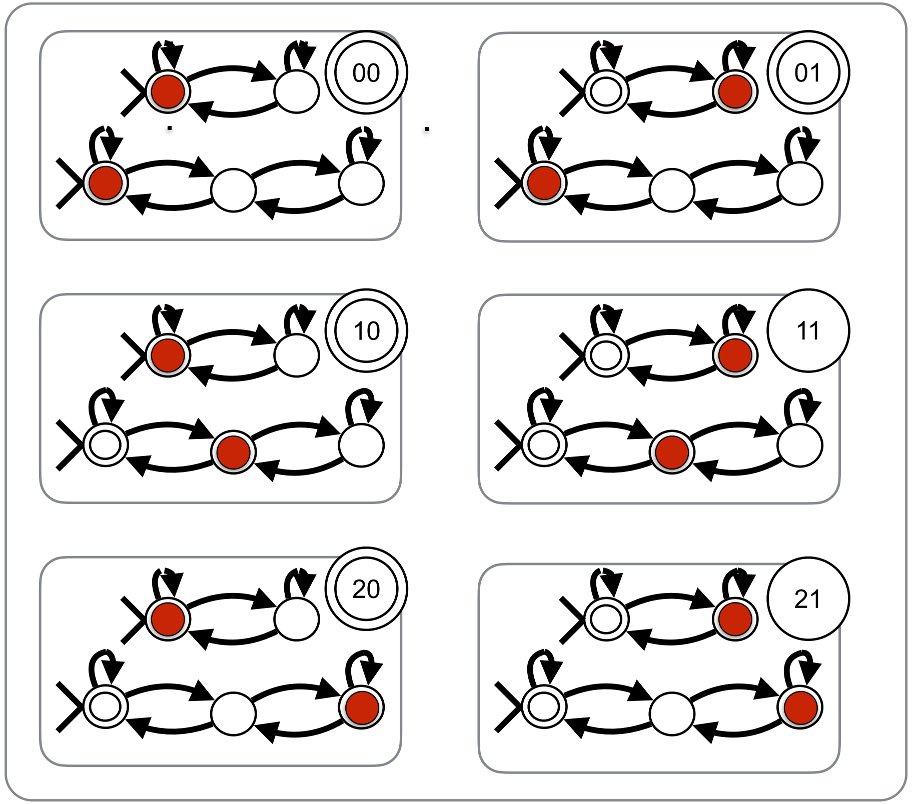
every NFA is equivalent to some FSM

every FSM is equivalent to some DFA

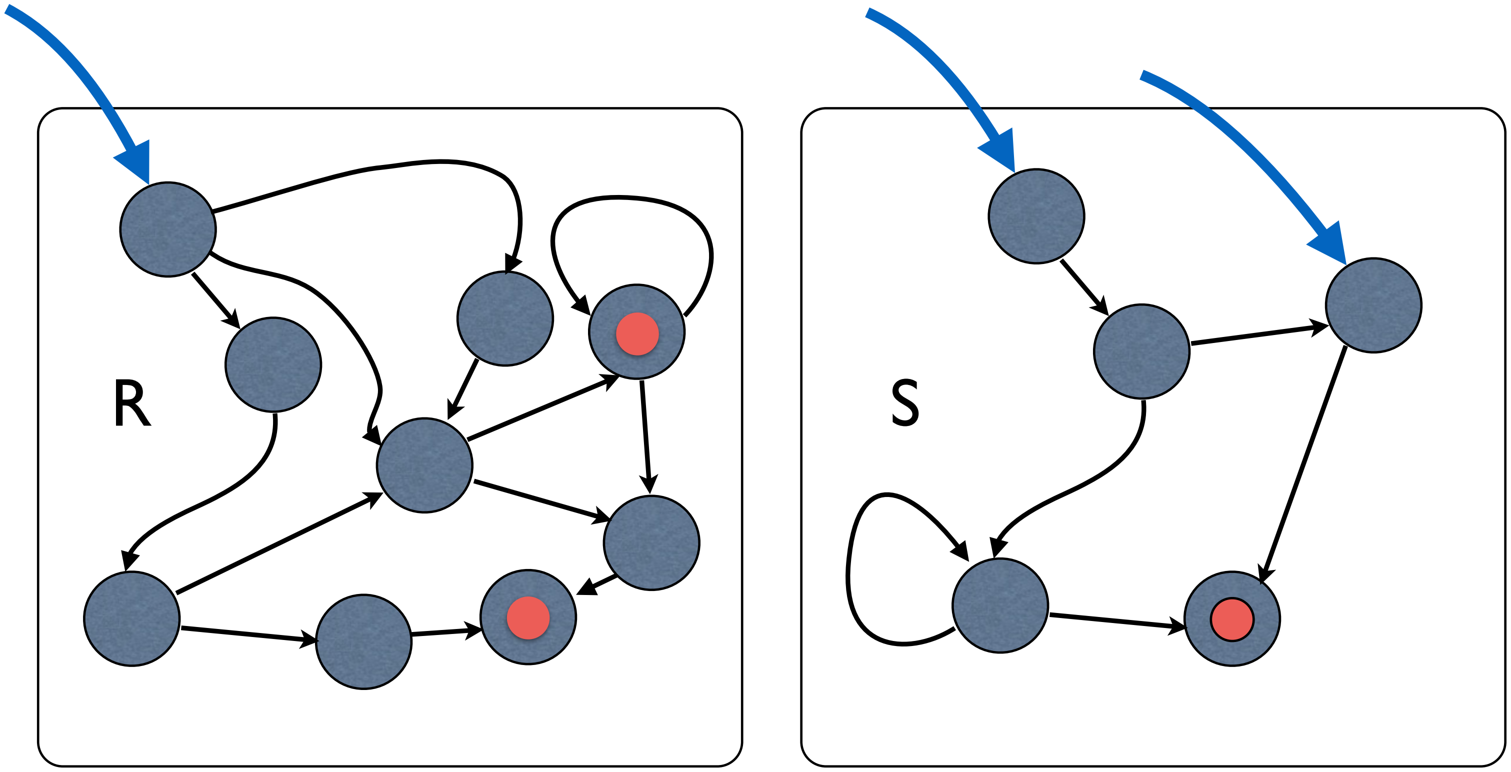
FSM model non-deterministic machines

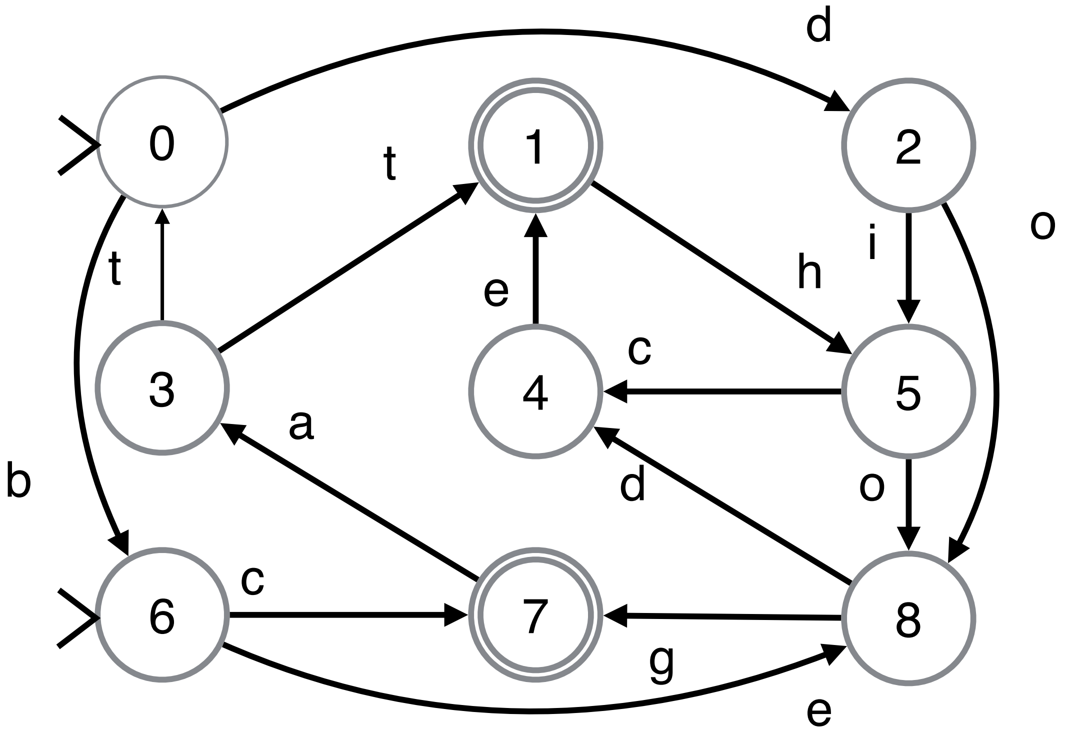
multiple threads of computation running in parallel

multiple start states

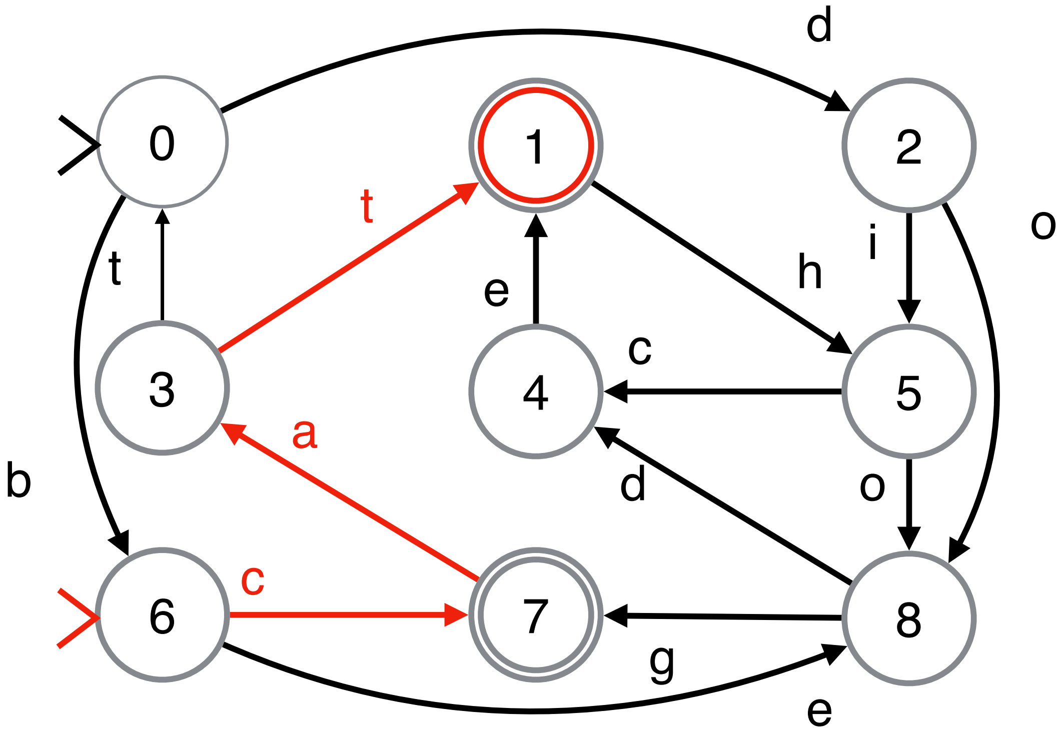


NFA any number of start states and accepting states

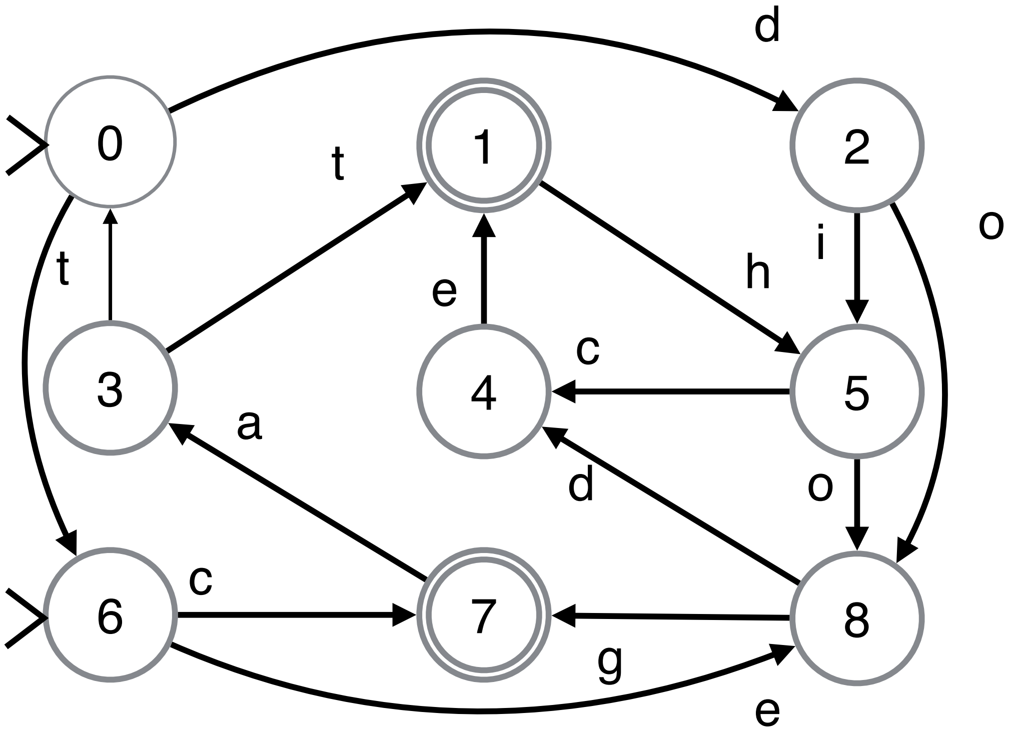




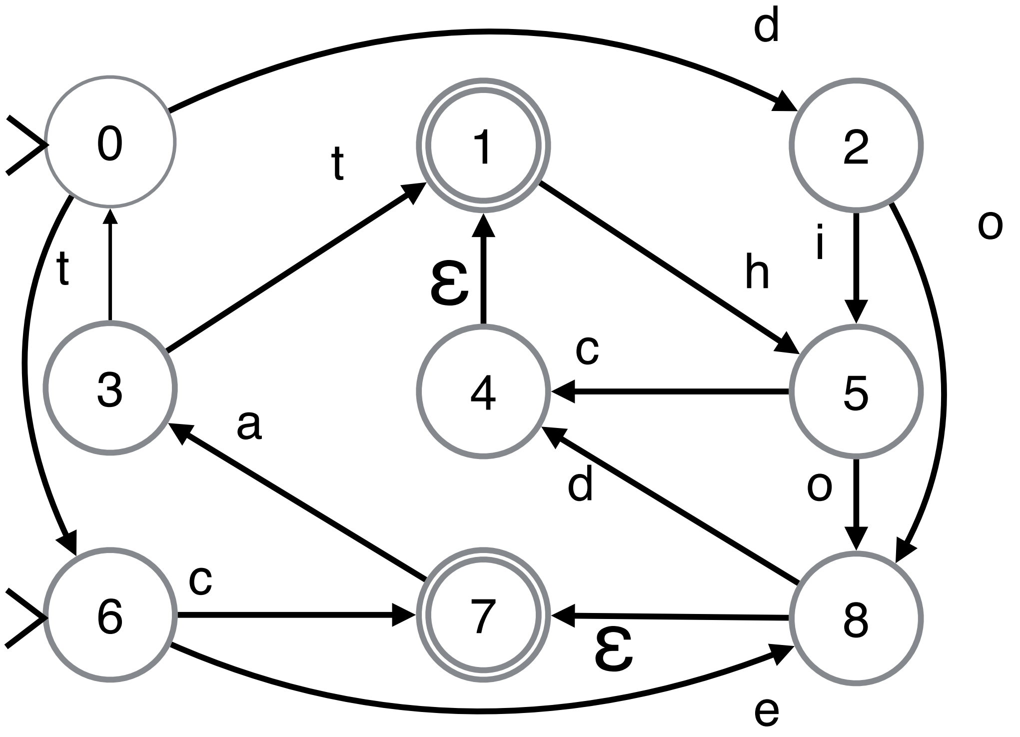
An FSM accepts a word iff there is a trace
from some start state q_0
to some finish state q_n
along transitions that spell out the word



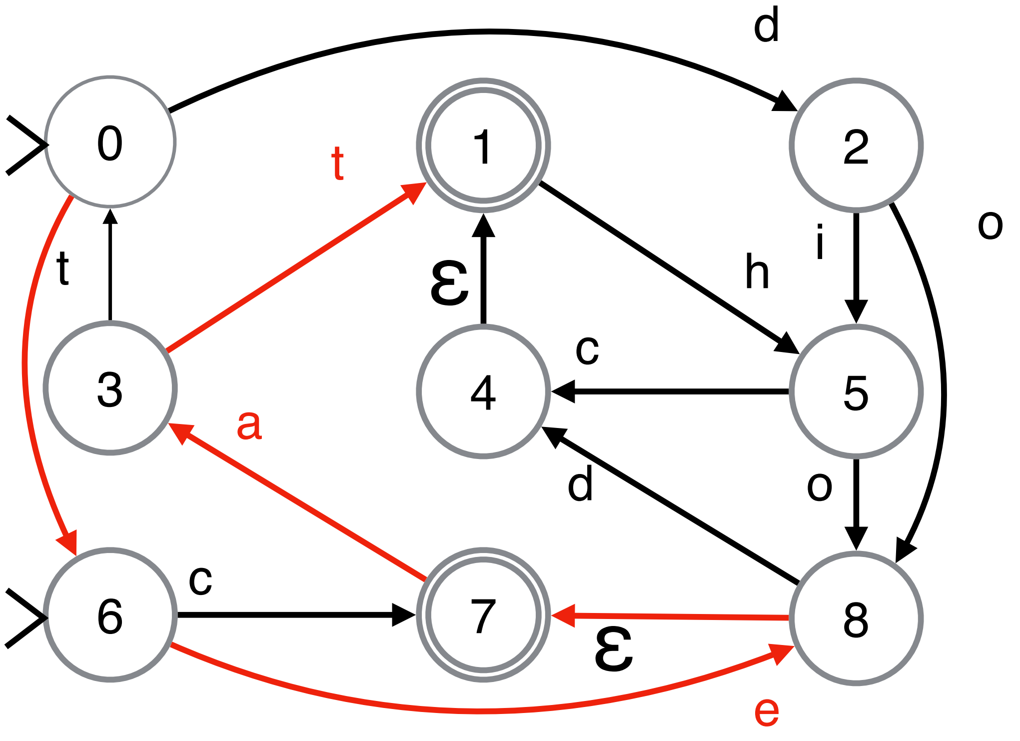
An FSM accepts a word iff there is a trace
from some start state q_0
to some finish state q_n
along transitions that spell out the word



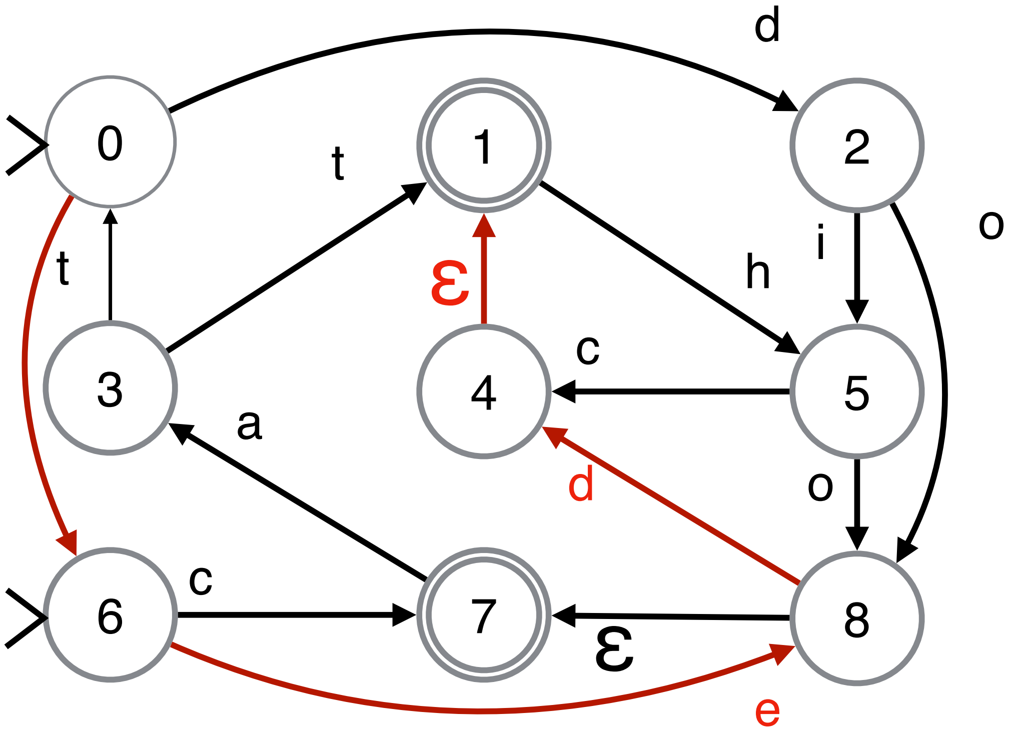
An FSM accepts a **string** iff there is a trace
from some start state q_0
to some finish state q_n
along transitions that spell out the **string**



An **ε**-FSM accepts a **string** iff there is a trace
from some start state q_0
to some finish state q_n
whose **non-ε** transitions spell out the **string**



An **ε**-FSM accepts a **string** iff there is a trace
from some start state q_0
to some finish state q_n
whose **non-ε** transitions spell out the **string**



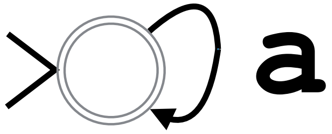
An **ε**-FSM accepts a **string** iff there is a trace
from some start state q_0
to some finish state q_n
whose **non-ε** transitions spell out the **string**

If $R \subseteq (\Sigma \cup \{\varepsilon\})^*$ is a regular language
with the alphabet $\Sigma \cup \{\varepsilon\}$ (where $\varepsilon \notin \Sigma$)
then R **//** $\varepsilon = \{ s \mathbin{//} \varepsilon \mid s \in R \}$ is regular
where $s \mathbin{//} \varepsilon$ is
the result of removing every ε from s

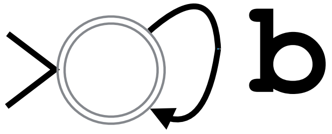
often 'explained' as
ε *stands for the empty string*

today we will use this theorem
tomorrow we will prove it

a*



b*



a*|b*

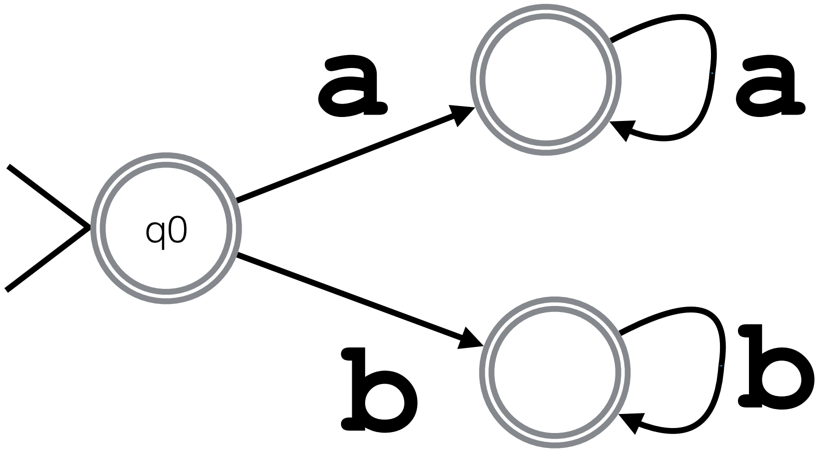
??

`(a|b)*`

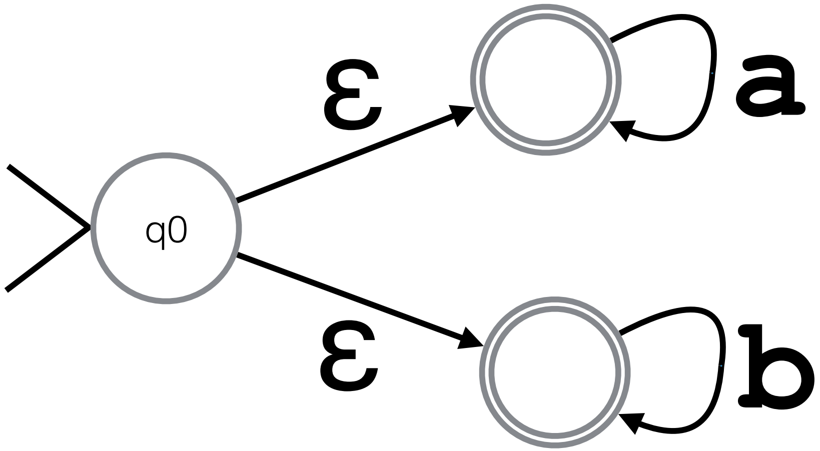
`a*|b*`

??

a*|b*



`a*|b*`

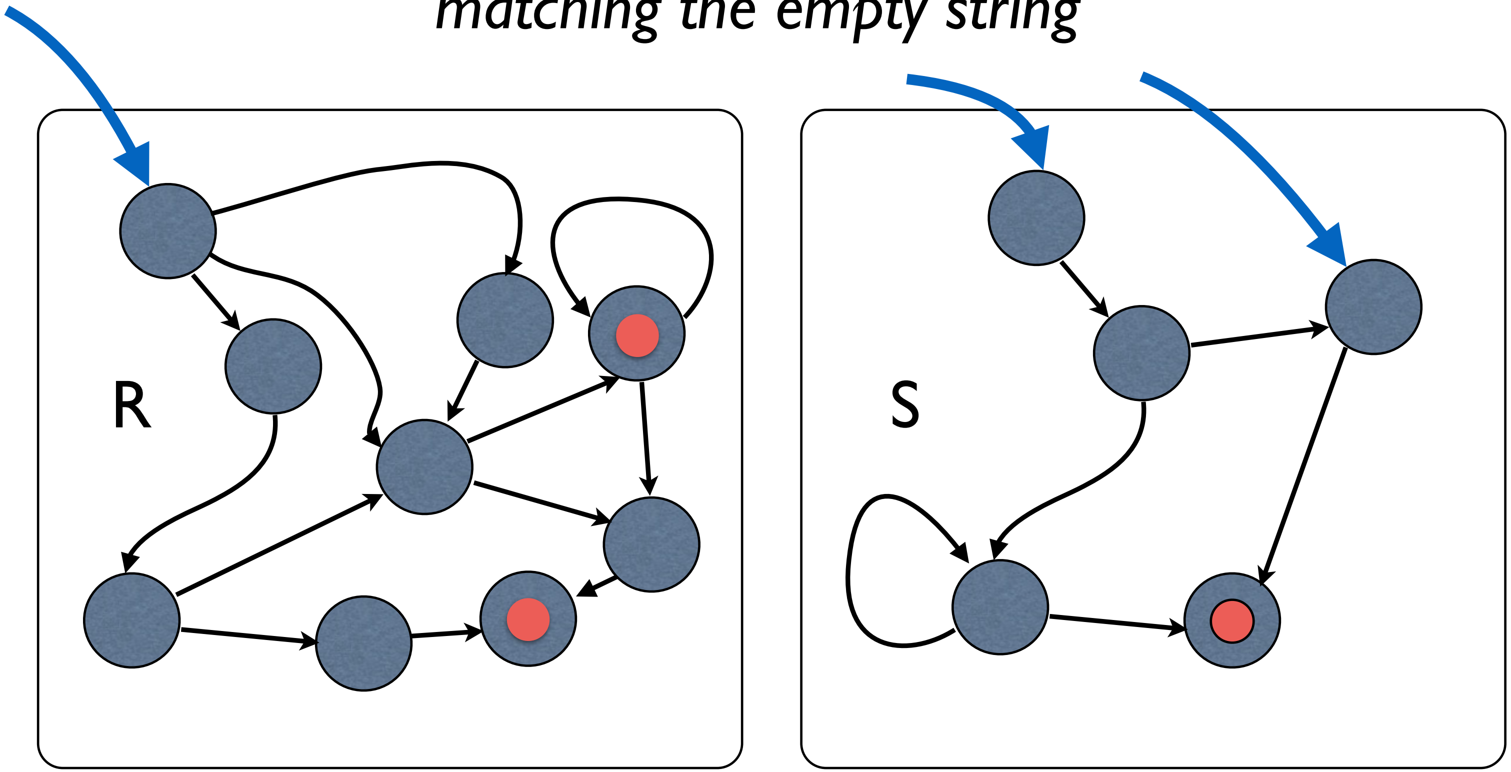


ε-NFA

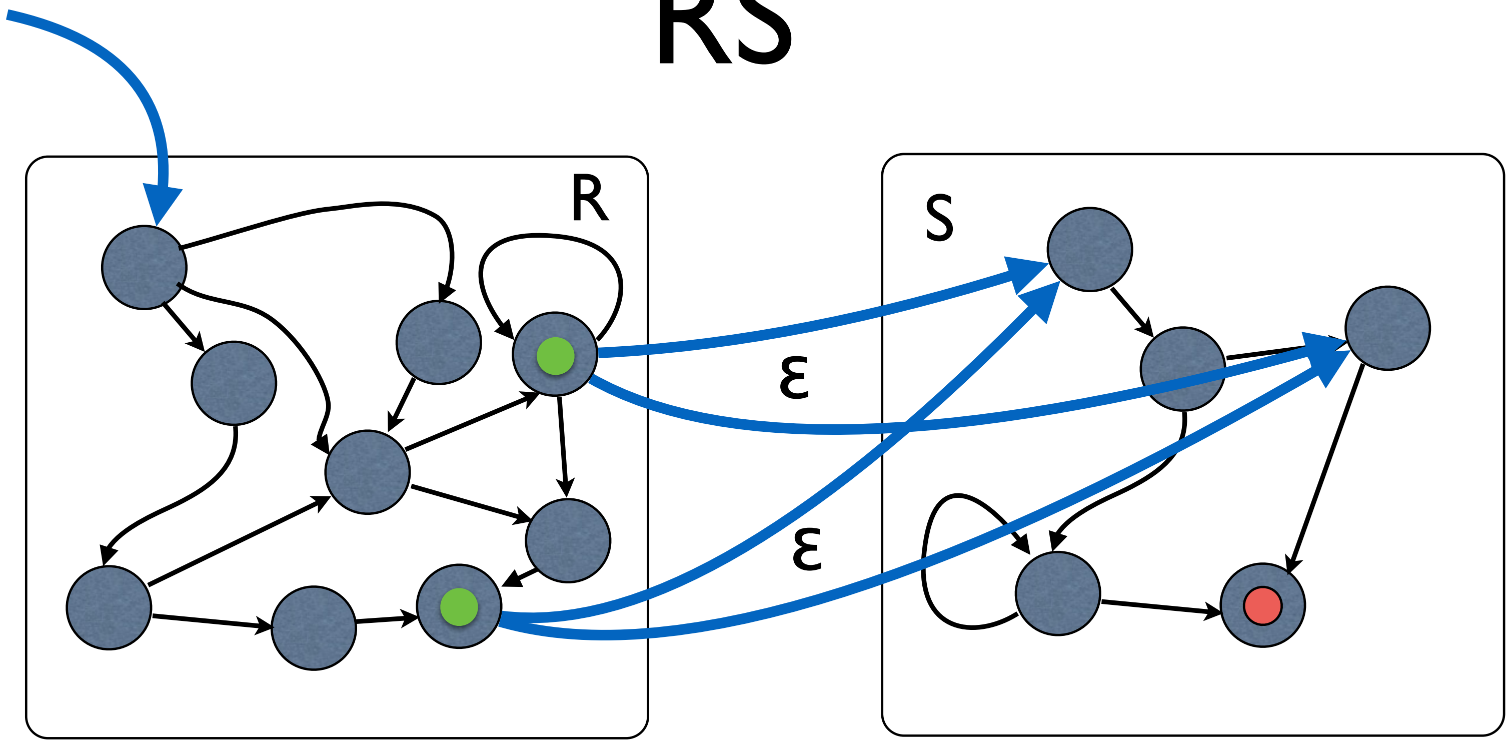
any number of start and finish states

ε - transitions ‘hidden actions’

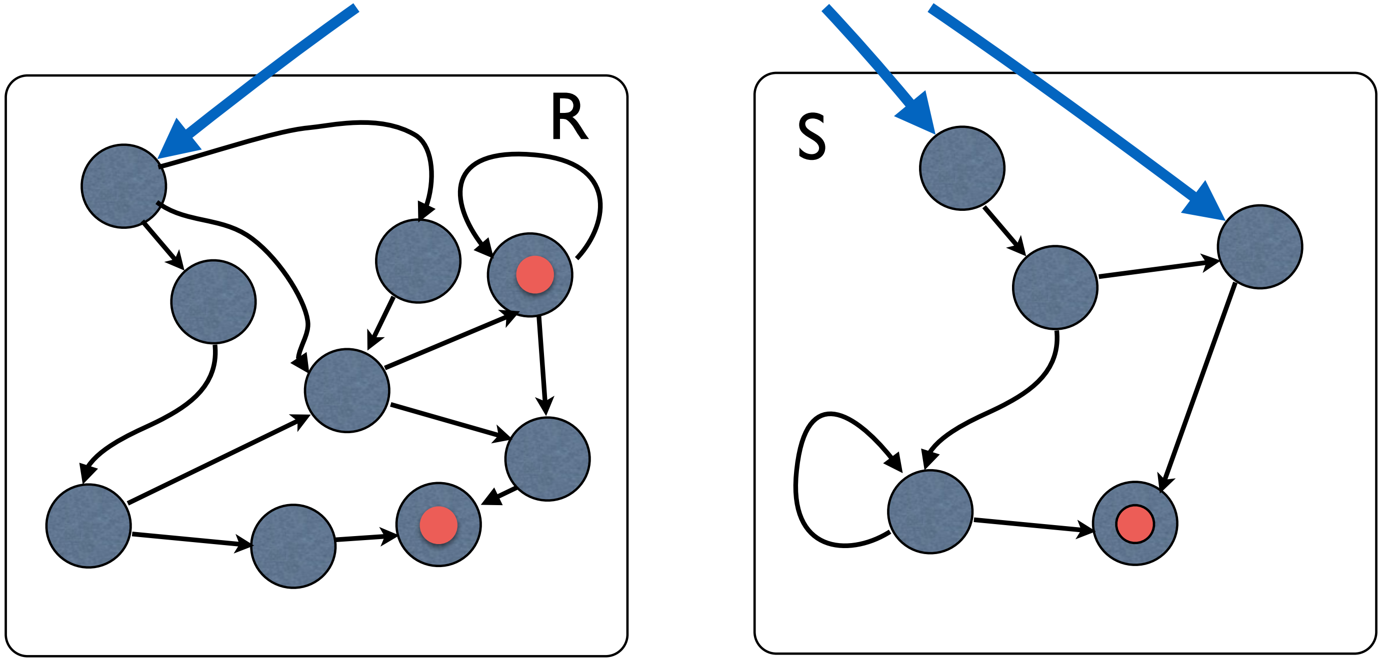
‘matching the empty string’



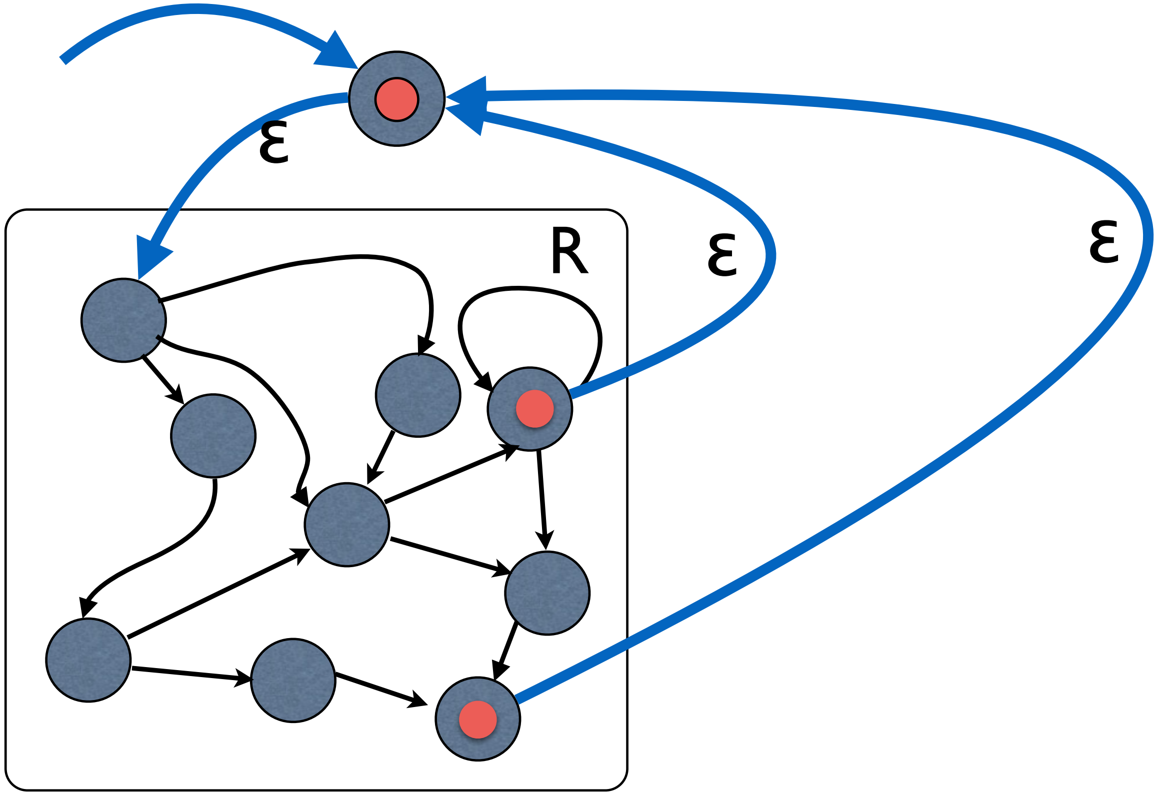
sequence
RS



alternation R|S



iteration R*



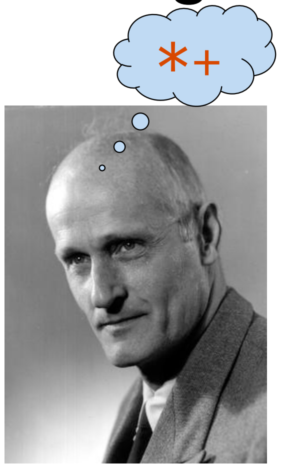
regular expressions

each regex is a pattern that matches a set of strings

- any character is a regex
  - matches itself
- if `R` and `S` are regex, so is `RS`
  - matches a match for `R` followed by a match for `S`
- if `R` and `S` are regex, so is `R|S`
  - matches any match for `R` or `S` (or both)
- if `R` is a regex, so is `R*`

  matches any sequence of 0 or more matches for `R`

Kleene *



Stephen Cole Kleene

1909-1994

- The algebra of regular expressions also includes elements `0` and `1`
  - `0 = ∅` matches nothing; `1 = Σ*` matches everything
  - `ε = ∅*` matches the empty string

```
0|R = R|0 = R      1|R = R|1 = 1
  0R = R0 = 0      εR = Rε = R
      ε = 0*       A* = ε|AA* = ε|A*A
```

the language of strings that match a regex, R, is recognised by some ε-FSM regular language ≡ recognised by some FSM

DFA regular languages — closed under Boolean operation

ε-FSM-regular languages — closed under regex operations

regex languages — strings matching some regex

FSM DFA ε-FSM
all recognise the same languages

every regular language is defined by some regex

regex languages are closed under Boolean operations