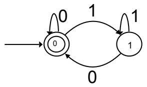
NFA and regex

- ε-transitions
- regular expressions

Two examples

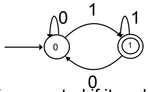
	×2	×2 + 1
	0	1
0	0	1
1	0	1



Even binary numbers

Input sequence is accepted if it ends with a zero.

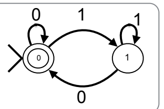
	×2	×2 + 1
	0	1
0	0	1
1	0	1



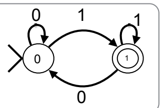
Odd binary numbers

Input sequence is accepted if it ends with a one.

The complement of a DFA regular language is DFA regular

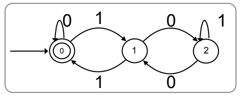


L_0 : even numbers
= 0 mod 2



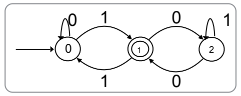
L_1 : odd numbers
= 1 mod 2

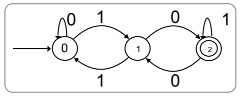
Three examples



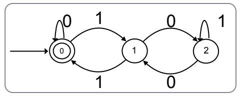
Which binary numbers are accepted?

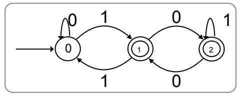
	×2	×2 + 1
mod 3	**0**	**1**
0	0	1
1	2	0
2	1	2





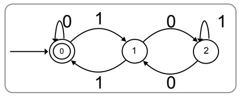
The complement of a DFA regular language is DFA regular



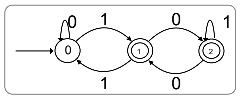


If $A \subseteq \Sigma^*$ is recognised by M then $\overline{A} = \Sigma^* \setminus A$ is recognised by $\overline{\mathbf{M}}$ where $\overline{\mathbf{M}}$ and **M** are identical except that the accepting states of $\overline{\mathbf{M}}$ are the non-accepting states of **M** and vice-versa

By three or not by three?

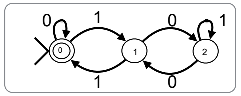


divisible by three



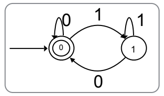
not
divisible by three

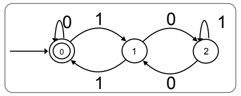
The intersection of two DFA regular languages is DFA regular



$L_0 = 0 \text{ mod } 3$
$L_1 = 1 \text{ mod } 3$
$L_2 = 2 \text{ mod } 3$

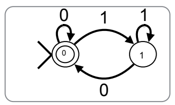
The intersection of two DFA regular languages is DFA regular

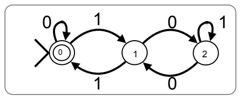




divisible by 6
≡
divisible by 2
and
divisible by 3

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



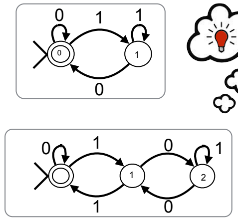


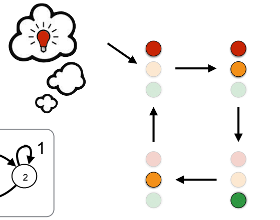
Run both machines in parallel?

Build one machine that simulates two machines running in parallel!

Keep track of the state of each machine.

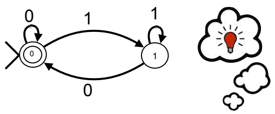
The intersection of two DFA-regular languages is DFA-regular

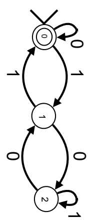


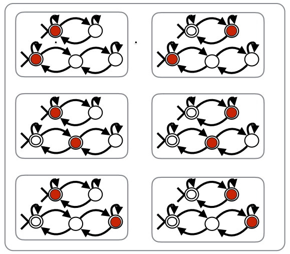


intersection of languages

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

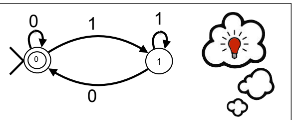


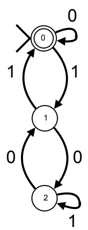


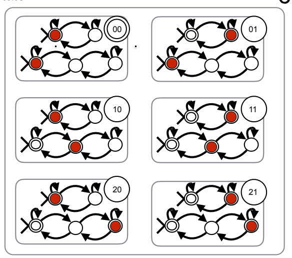


intersection of languages

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

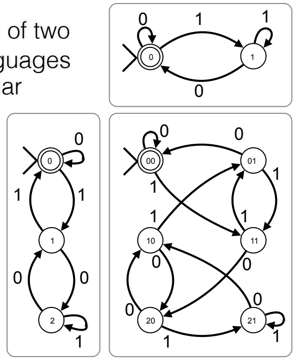






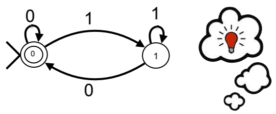
intersection of two regular languages is regular

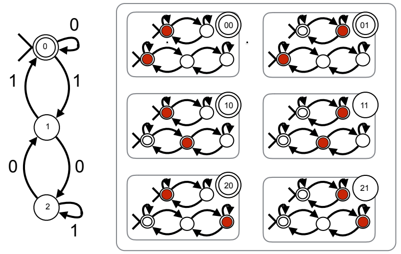
run two machines in synchrony



union of languages

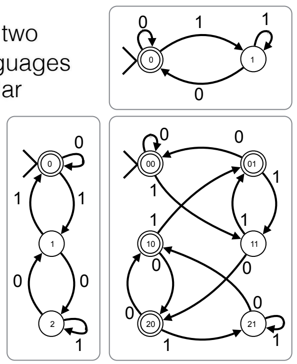
run the two machines in parallel when a string is in the union of the two languages, either or both are in an accepting state





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 regular languages
closed under concatenation R S ?
closed under iteration ( )* ?

They are ! To show this we add more non-deterministic NFA are FSM with ε-transitions

We will do three things!

Show that
every regular language is accepted by some NFA

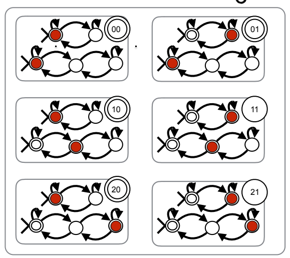
Show that every NFA is equivalent to some FSM

Show that every FSM is equivalent to some DFA

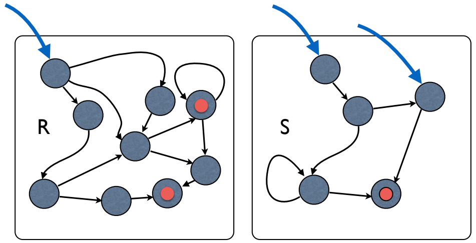
FSM model non-deterministic machines

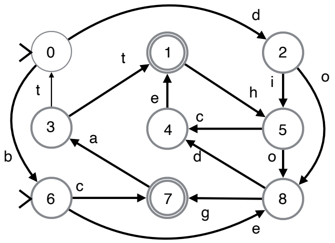
multiple threads of computation running in parallel

multiple start states

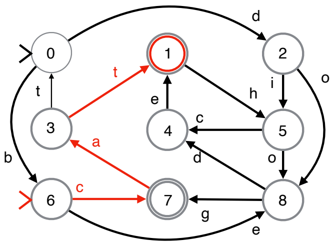


NFA any number of start states and accepting states

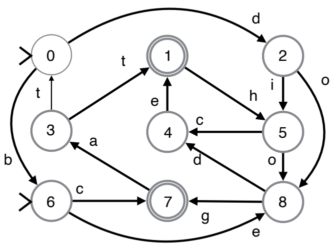




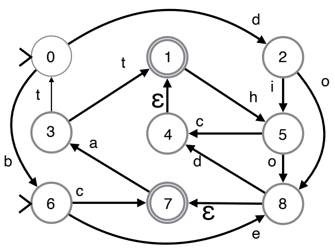
An FSM accepts a word iff there is a trace
from some start state q_0
to some finish state q_n
along transitions that spell out the word



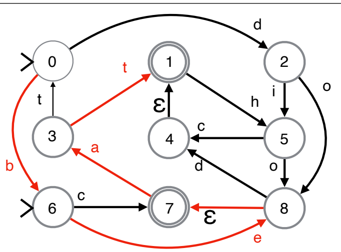
An FSM accepts a word iff there is a trace
from some start state q_0
to some finish state q_n
along transitions that spell out the word



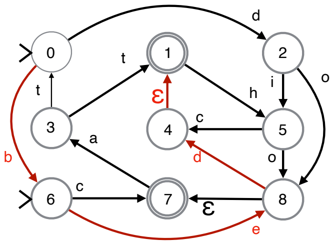
An FSM accepts a **string** iff there is a trace
from some start state q_0
to some finish state q_n
along transitions that spell out the **string**



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**

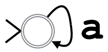


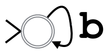
An ε-FSM accepts a **string** iff there is a trace
from some start state q_0
to some finish state q_n
whose **non-ε** transitions spell out the **string**

If R ⊆ (Σ∪{ε})* is a regular language
with the alphabet Σ∪{ε} (where ε∉Σ)
then R **//** ε = { s // ε | s ∈ R } is **regular**
where **s** // **ε** 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*` ??

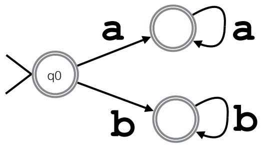
(a|b)*

a*|b*

??

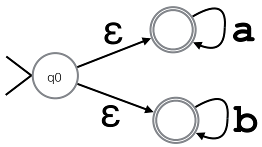
---

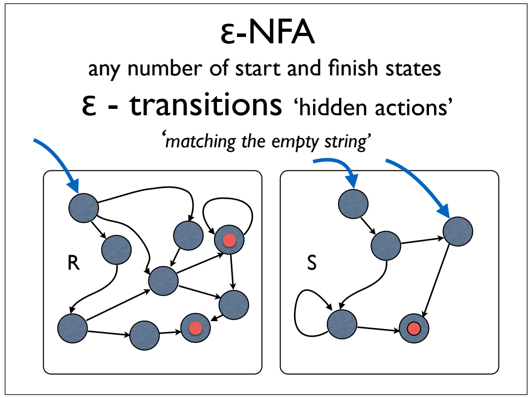
a*|b*

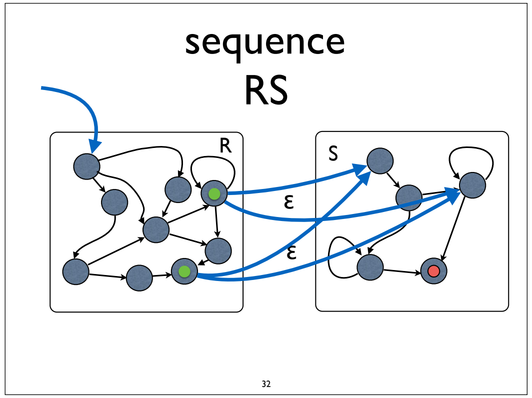


---

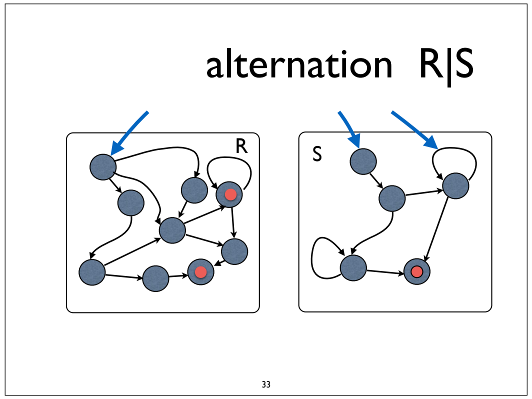
a*|b*





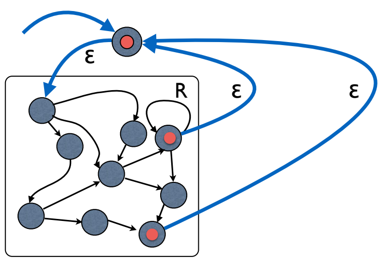


The red lines are automatic transitions that can always happen, without any input.
They are normally labelled ε



The red lines are automatic transitions that can always happen, without any input.
They are normally labelled ε

iteration R*



The red lines are automatic transitions that can always happen, without any input.
They are normally labelled ε

regular expressions

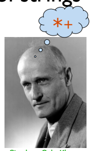
each regex is a pattern that matches a set of strings

- 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`
- 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
```

Kleene *



Stephen Cole Kleene

1909-1994

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