https://app-ca.tophat.com/e/835603

NFA and regex

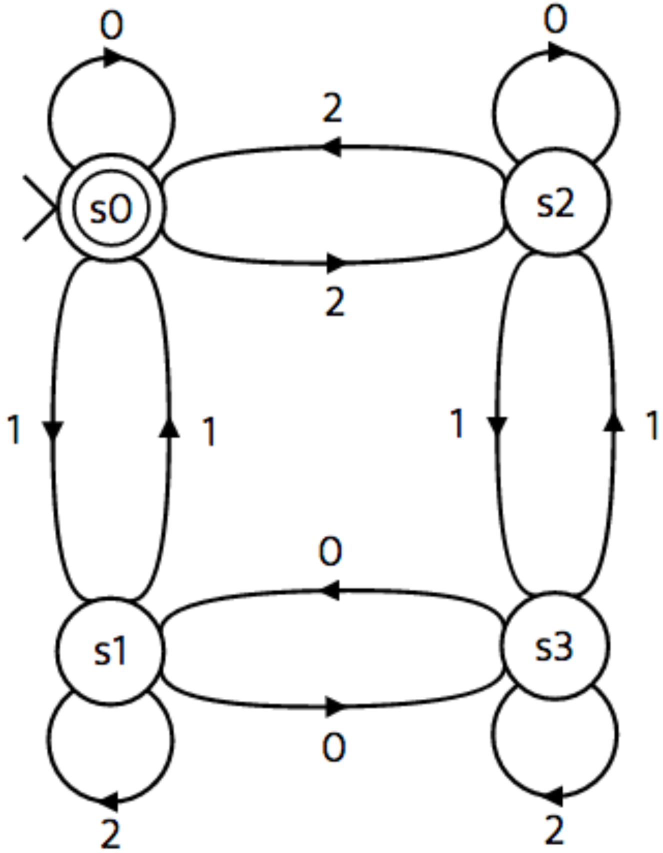
cl

- the Boolean algebra of languages
- regular expressions

KISS – DFA

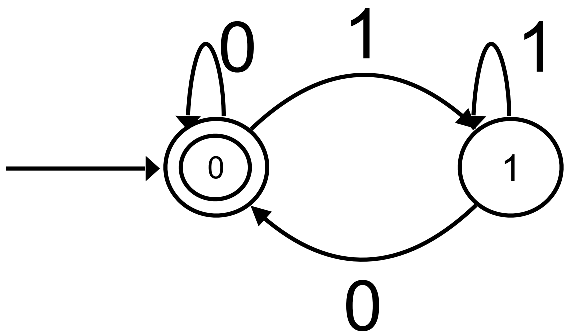
Deterministic **F**inite **A**utomaton

Exactly one start state, and
from each state, **q**,
for each token, **t**,
there is
exactly one transition
from **s** with label **t**



Two examples

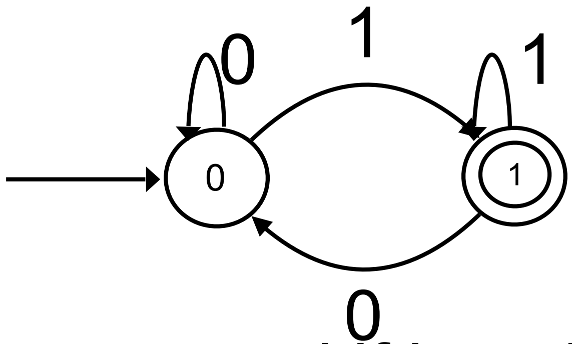
	×2	×2 + 1
	0	**1**
0	0	1
1	0	1



Even binary numbers

Input sequence is accepted if it ends with a zero.

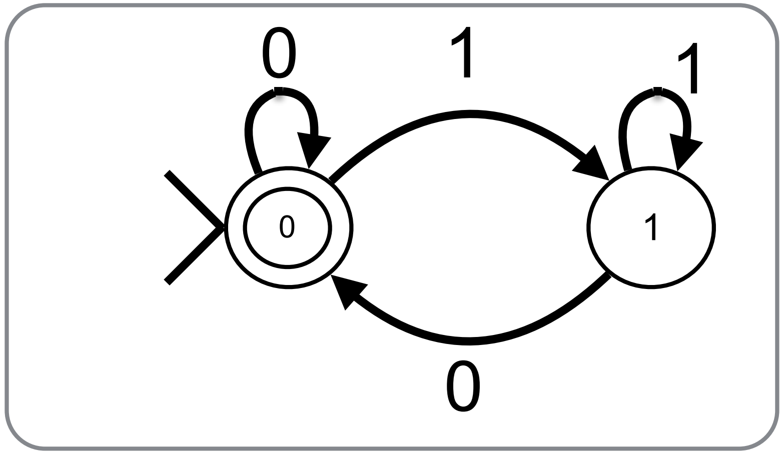
	×2	×2 + 1
	0	**1**
0	0	1
1	0	1



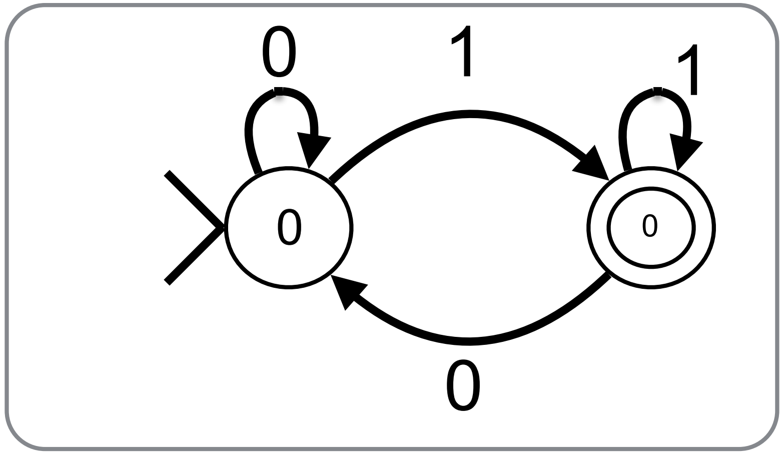
Odd binary numbers

Input sequence is accepted if it ends with a one.

The complement of a regular language is regular

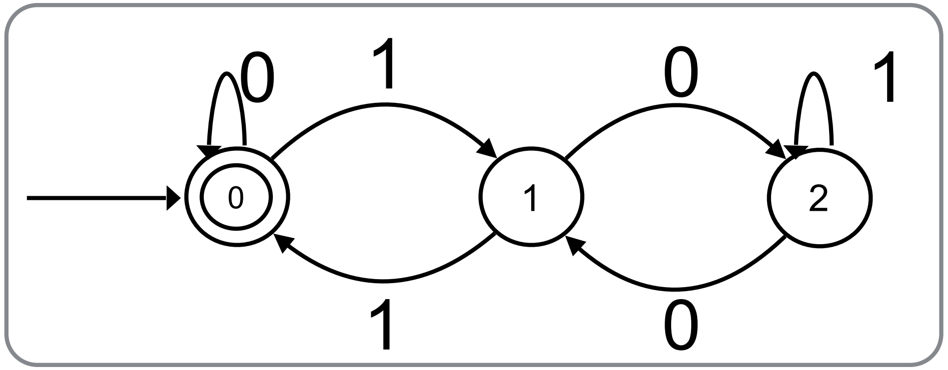


L_0 : even numbers
$= 0 \bmod 2$



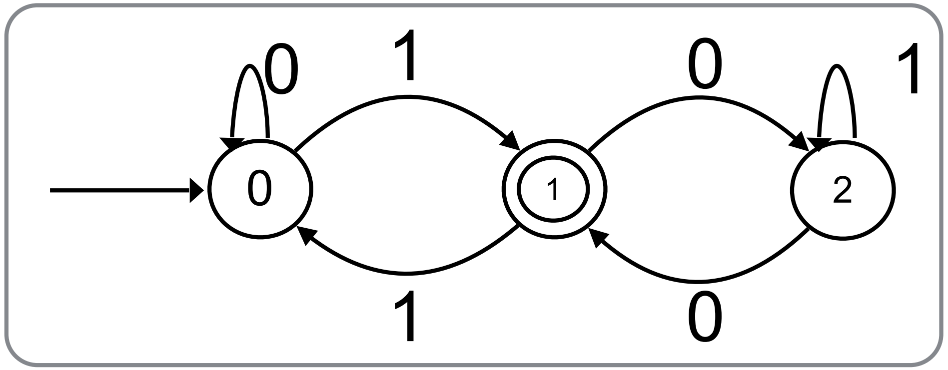
L_1 : odd numbers
$= 1 \bmod 2$

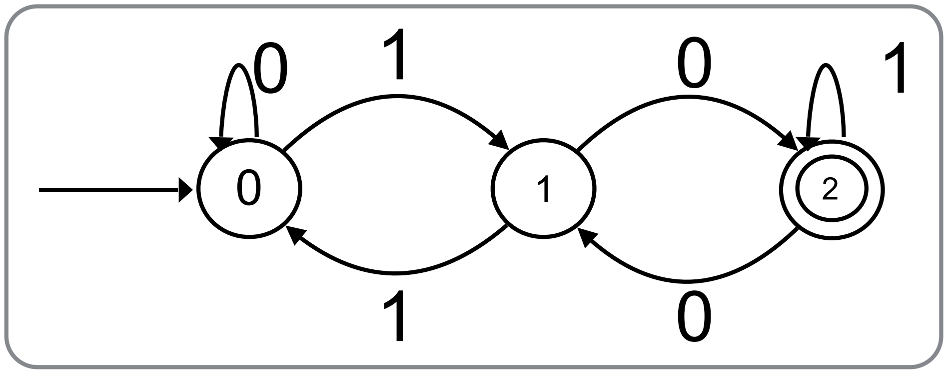
Three examples



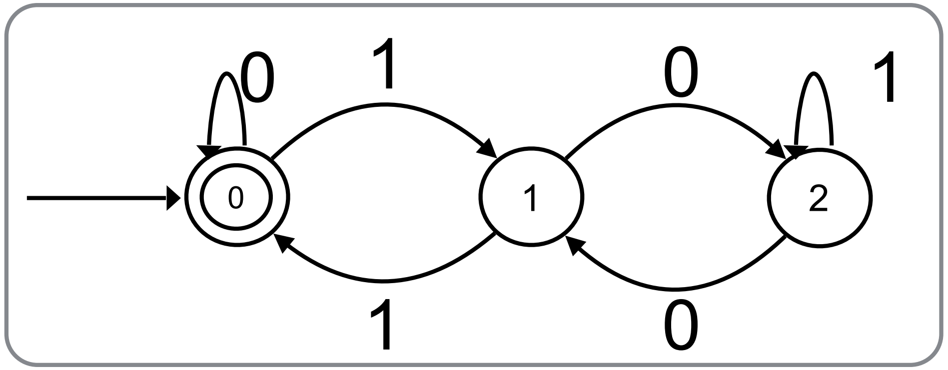
Which binary numbers are accepted?

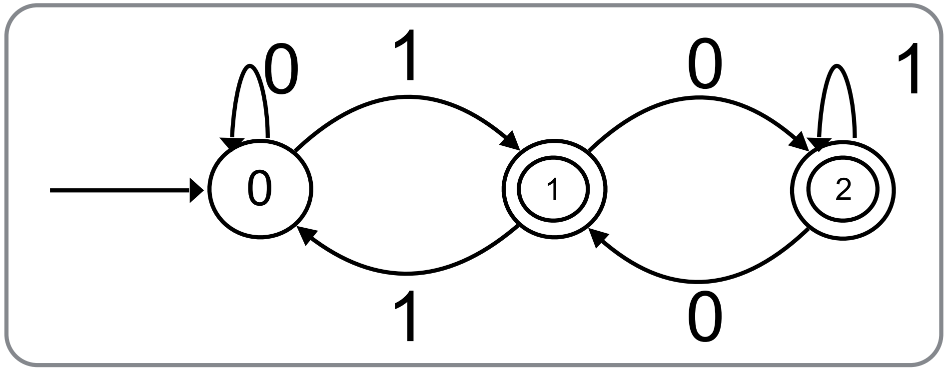
	×2	×2 + 1
mod 3	**0**	**1**
0	0	1
1	2	0
2	1	2





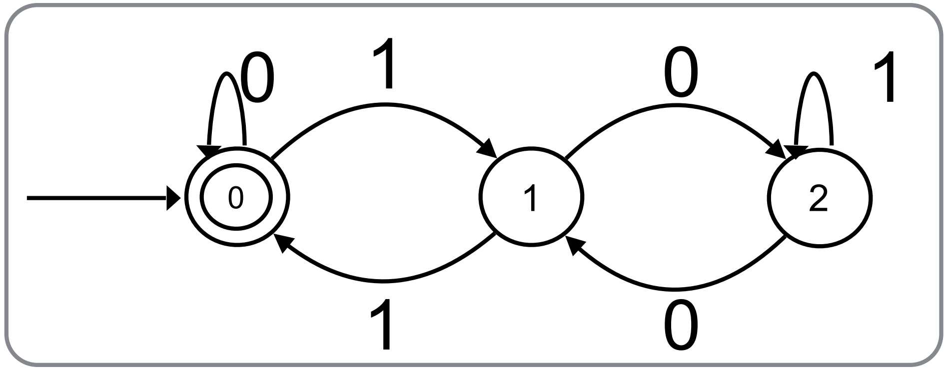
The complement of a regular language is regular



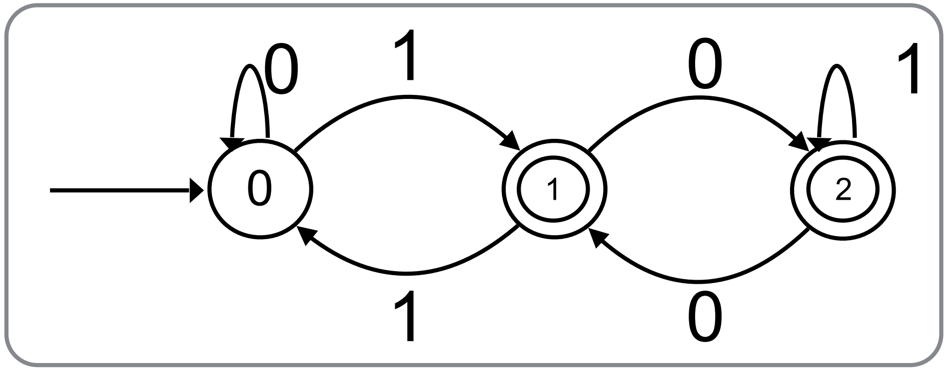


If $A \subseteq \Sigma^*$ is recognised by M then $\overline{\mathbf{A}} = \Sigma^* \setminus A$ is recognised by $\overline{\mathbf{M}}$ where $\overline{\mathbf{M}}$ and **M** are identical except that the accepting states of $\overline{\mathbf{M}}$ are the non-accepting states of **M** and vice-versa

By three or not by three?

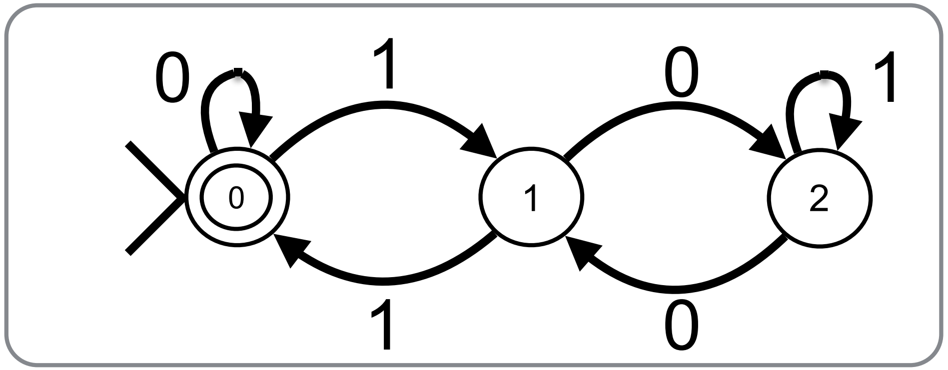


divisible by three



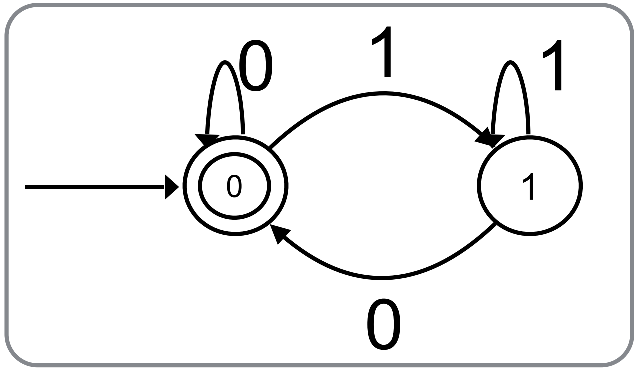
not
divisible by three

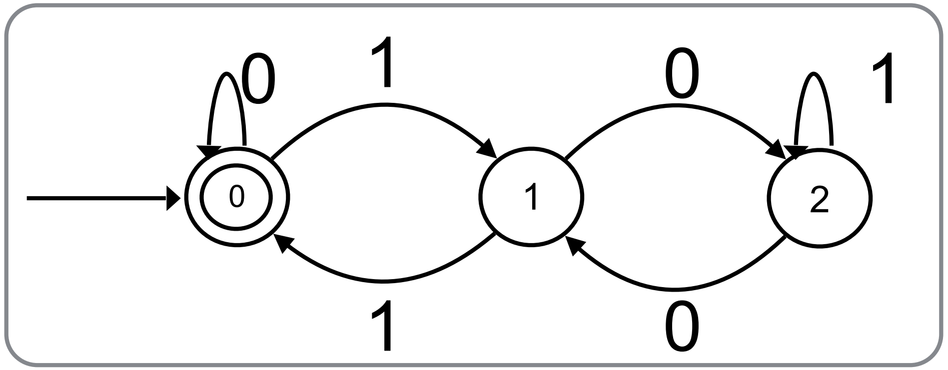
The intersection of two regular languages is regular



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

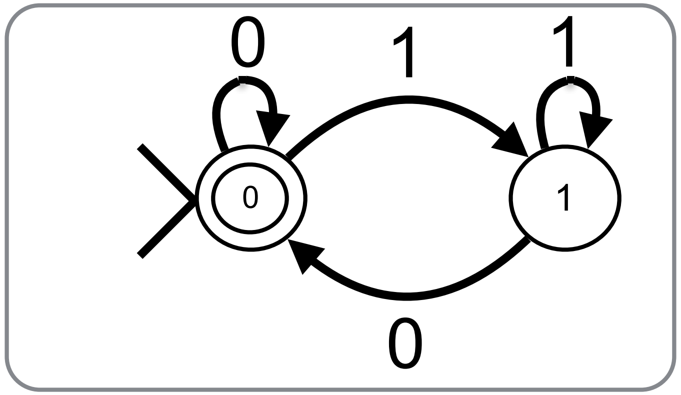
The intersection of two regular languages is regular

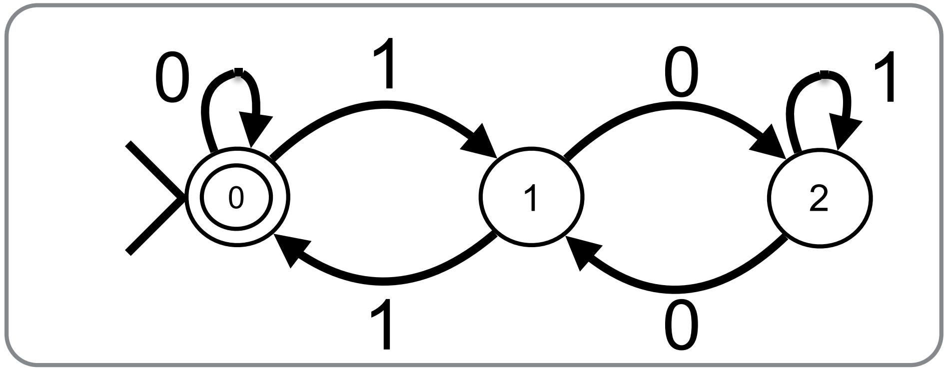




divisible by 6
≣
divisible by 2
and
divisible by 3

The intersection of two regular languages is regular



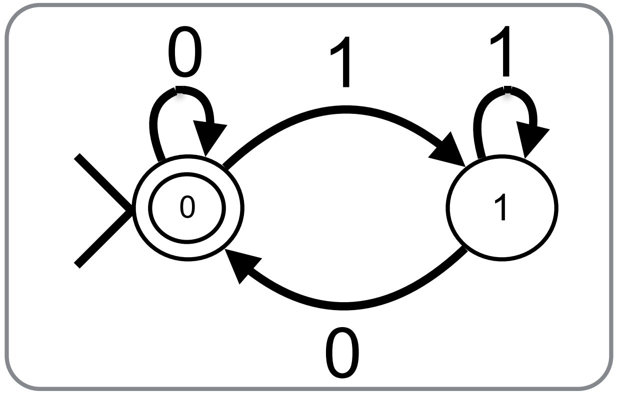


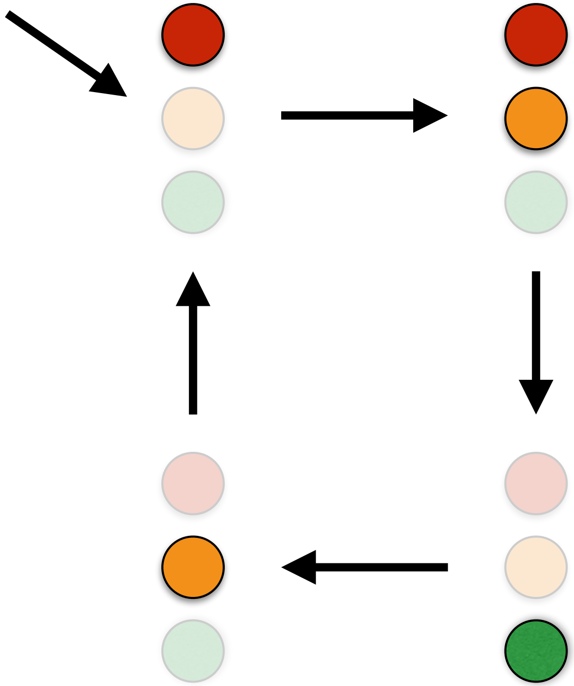
Run both machines in parallel?

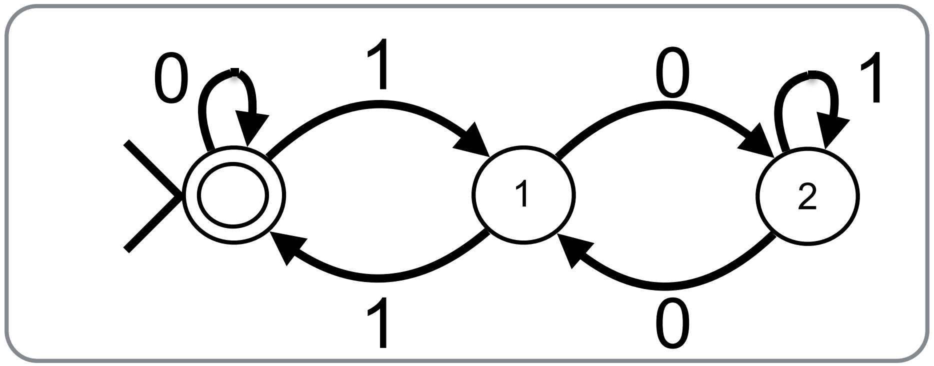
Build one machine that simulates two machines running in parallel!

Keep track of the state of each machine.

The intersection of two regular languages is regular

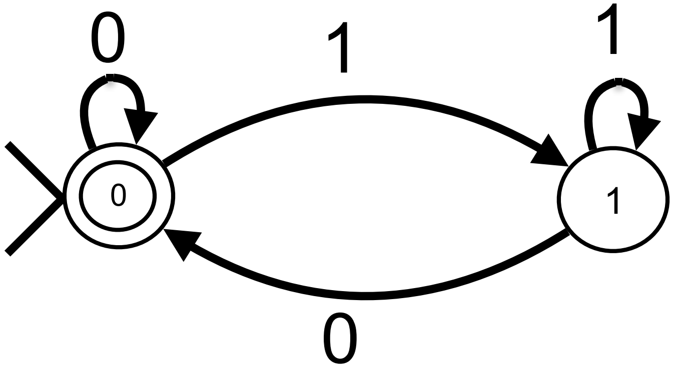


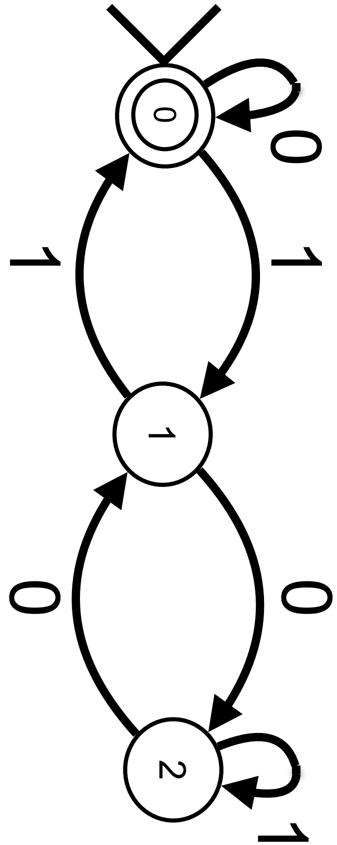


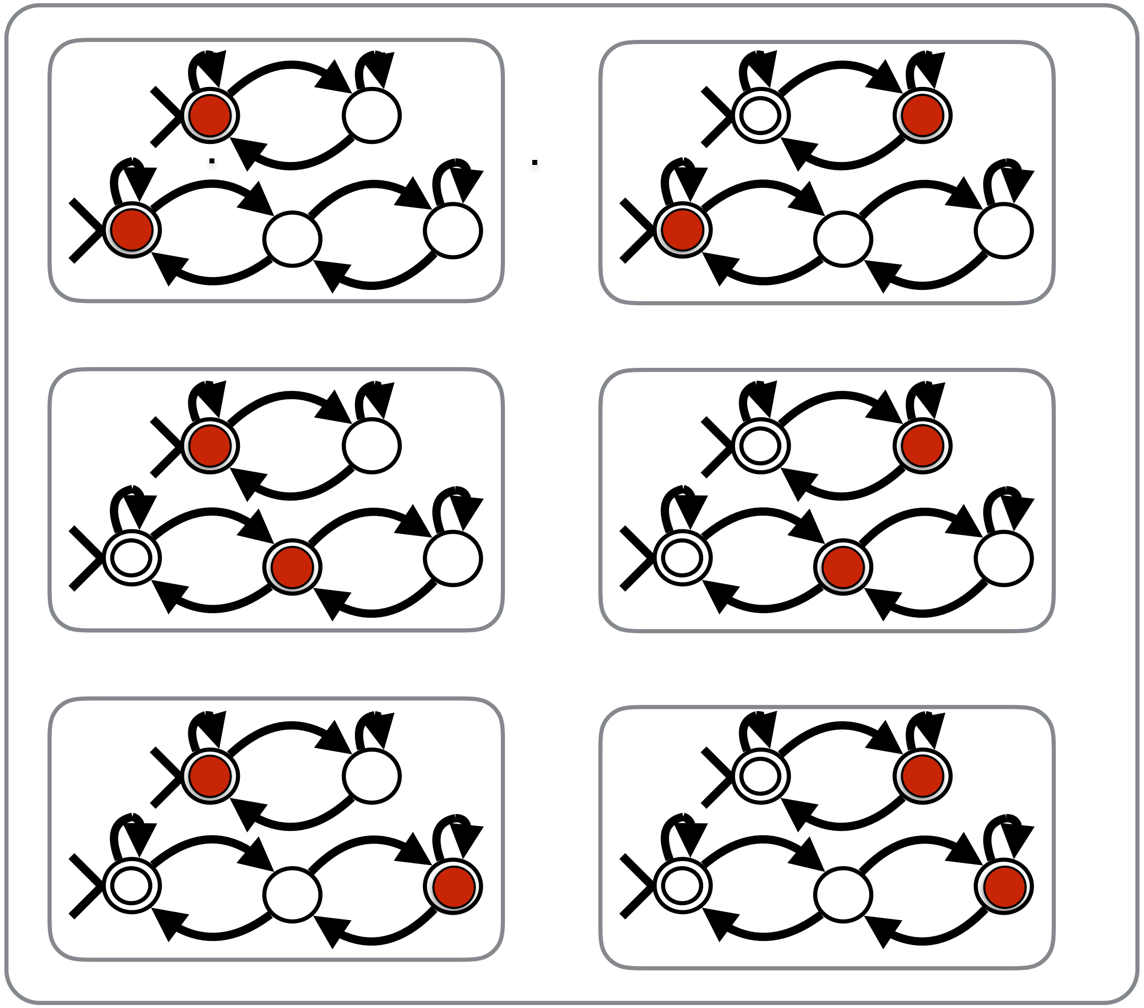


intersection of languages

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

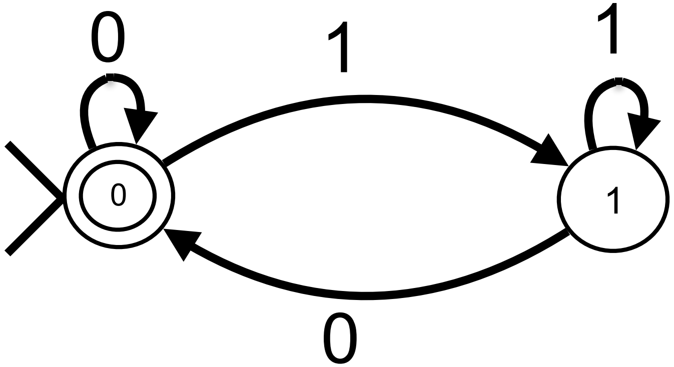


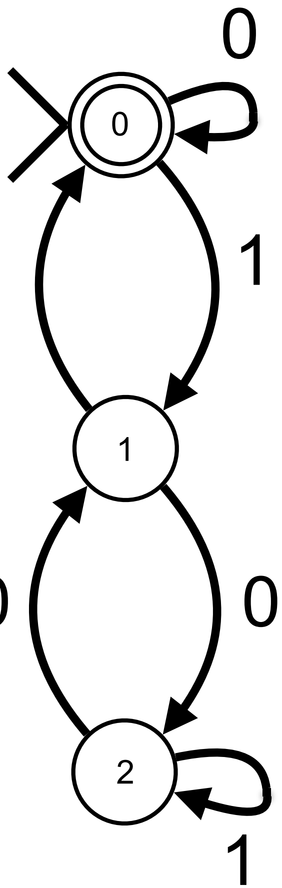


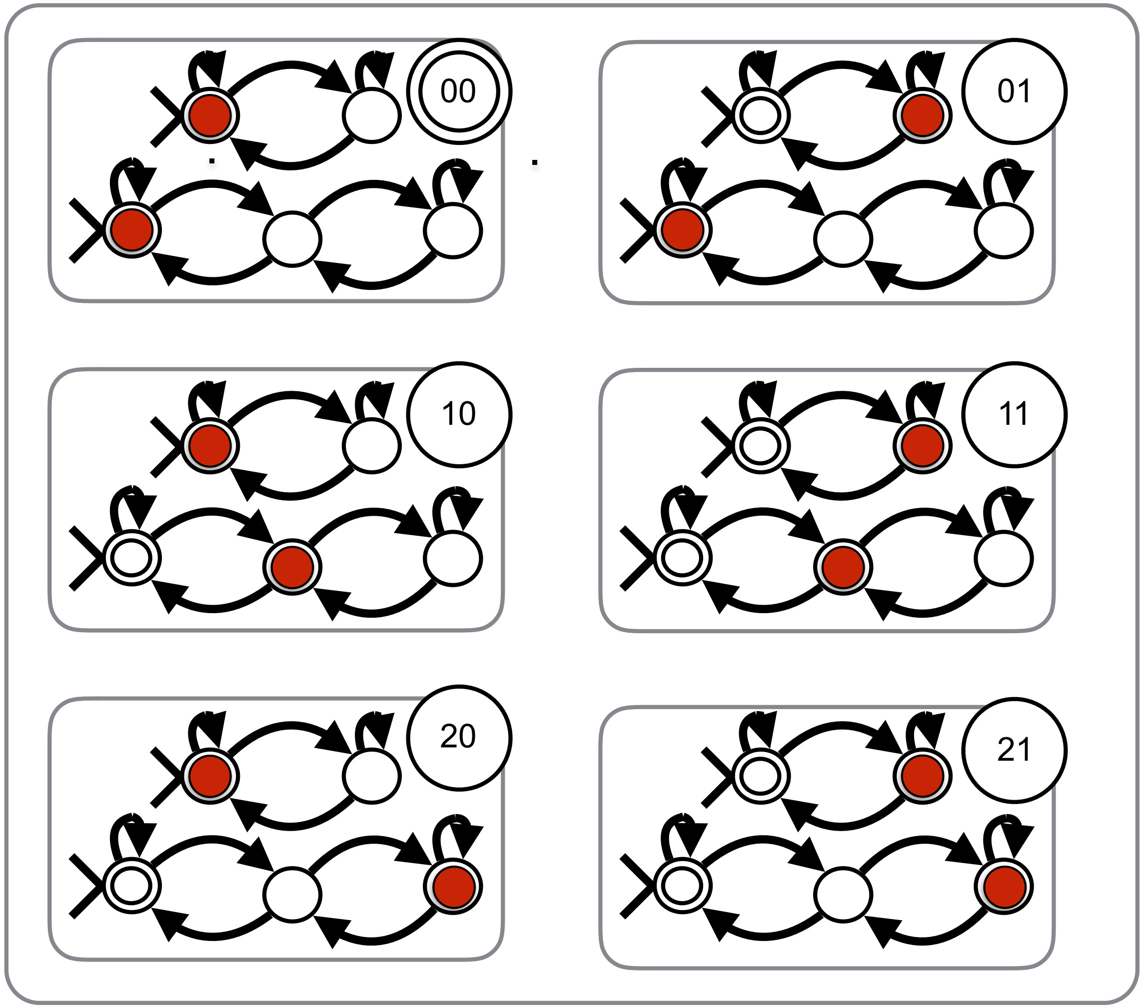


intersection of languages

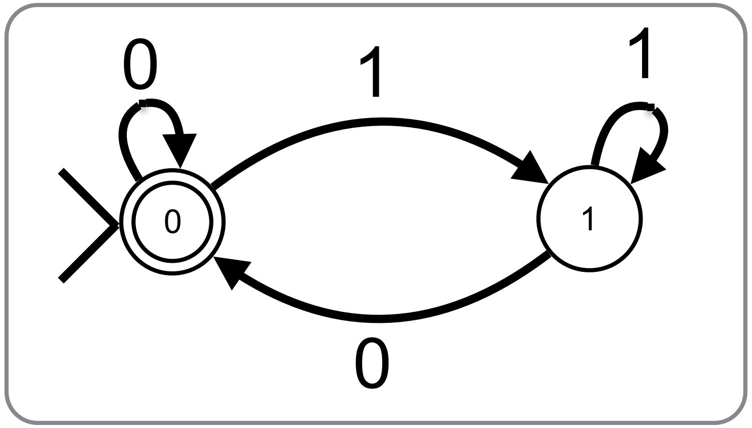
run the two machines in parallel
when a string is in both languages,
both are in an accepting state



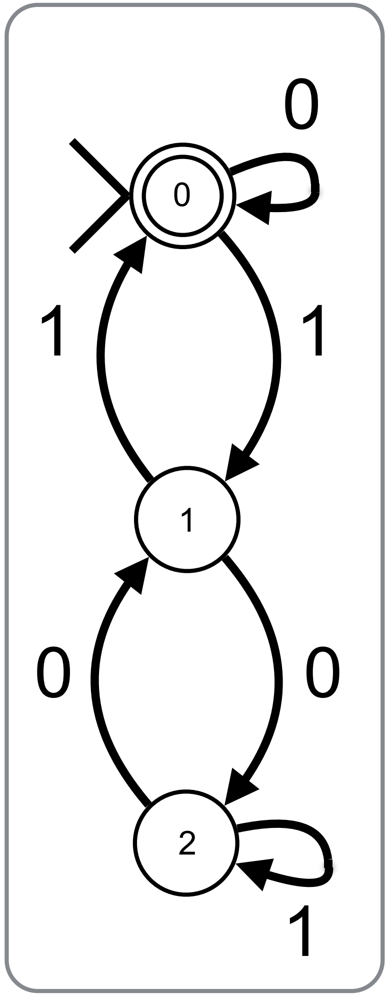


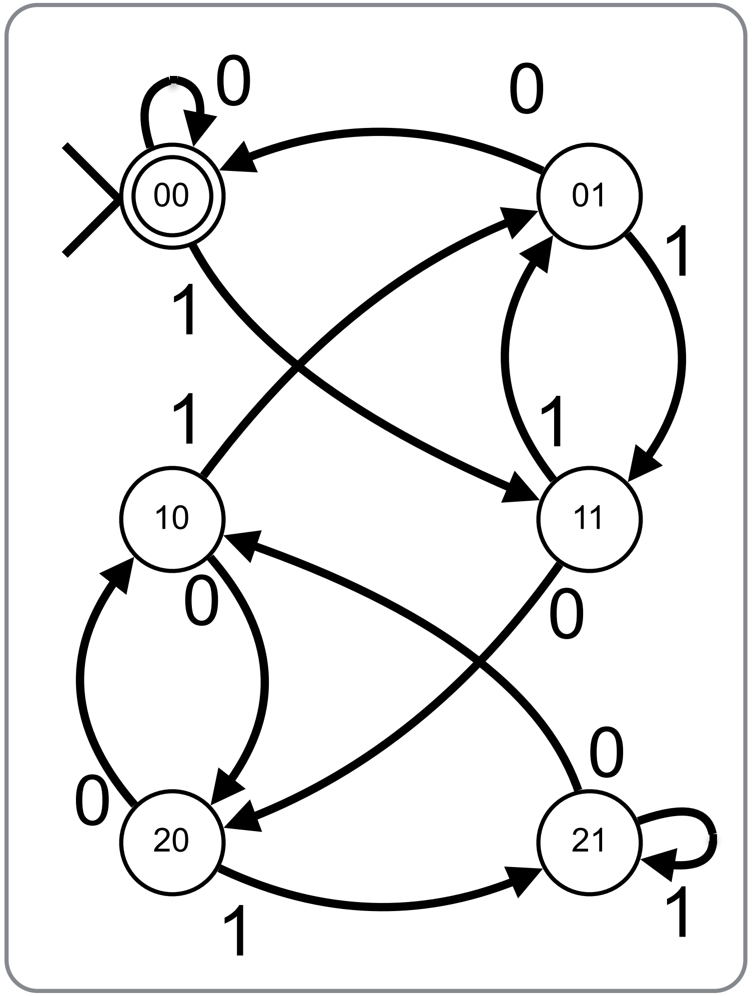


intersection of two regular languages is regular



run two machines in synchrony





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

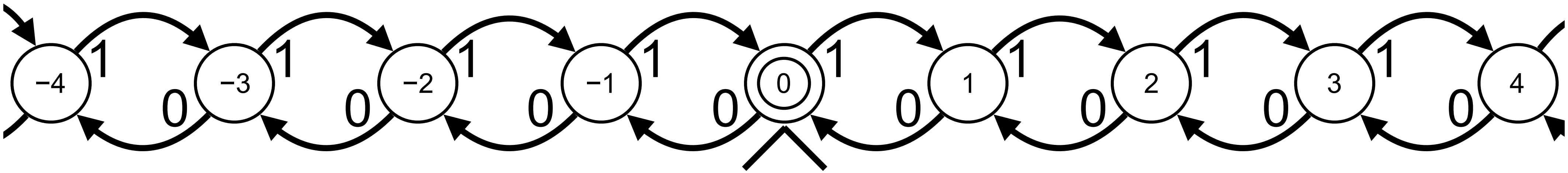
- Since they are closed under intersection and complement.

Are all languages $A \subseteq \Sigma^*$ regular?

- A finite machine can only do so much
  or rather, so little.

Consider the language with two symbols $\Sigma = \{0,1\}$, consisting of the all strings that contain equal numbers of zeros and ones.

How can we show that this language is not regular?



Are all languages $A \subseteq \Sigma^*$ regular?

Consider the language with two symbols $\Sigma = \{0,1\}$,
consisting of the all strings
that contain equal numbers of zeros and ones.

Suppose we have a DFA that recognises this language.

Let s_n be the state the machine reaches after an input of n zeros.

If the machine is in state s_n and we give it an input of n ones it will be in an accepting state.

Are all languages $A \subseteq \Sigma^*$ regular?

A = strings that contain equal numbers of zeros and ones.

We have a DFA that recognises this language.

s_n is the state the machine after an input of n zeros.

If the machine is in state s_n and we give it an input of n ones it will be in an accepting state.

If $n \neq m$ what can we say about s_n and s_m ?

Are all languages $A \subseteq \Sigma^*$ regular?

A = strings that contain equal numbers of zeros and ones.
We have a DFA that recognises this language.

s_n is the state the machine after an input of n zeros.

If the machine is in state s_n (after an input of n zeros),
and we give it an input of m ones
it will be in an accepting state iff $m = n$.

Therefore, if $n \neq m$ then $s_n \neq s_m$ (why?)

So our machine must have infinitely many states!!

An FSM with n states cannot count beyond n-1.

Which languages $A \subseteq \Sigma^*$ are regular?

What kind of answer can we give to a question like this?

If we have a machine
then the language it recognises is regular.

For some languages, **L**, e.g. our #0s = #1s example,
we can argue that no FSM can recognise **L**.
But that is not a general argument.

Instead of finding a property,
to characterise the regular languages,
we will find **a set of rules that generate**
the set of regular languages.

Which languages $A \subseteq \Sigma^*$ are regular?

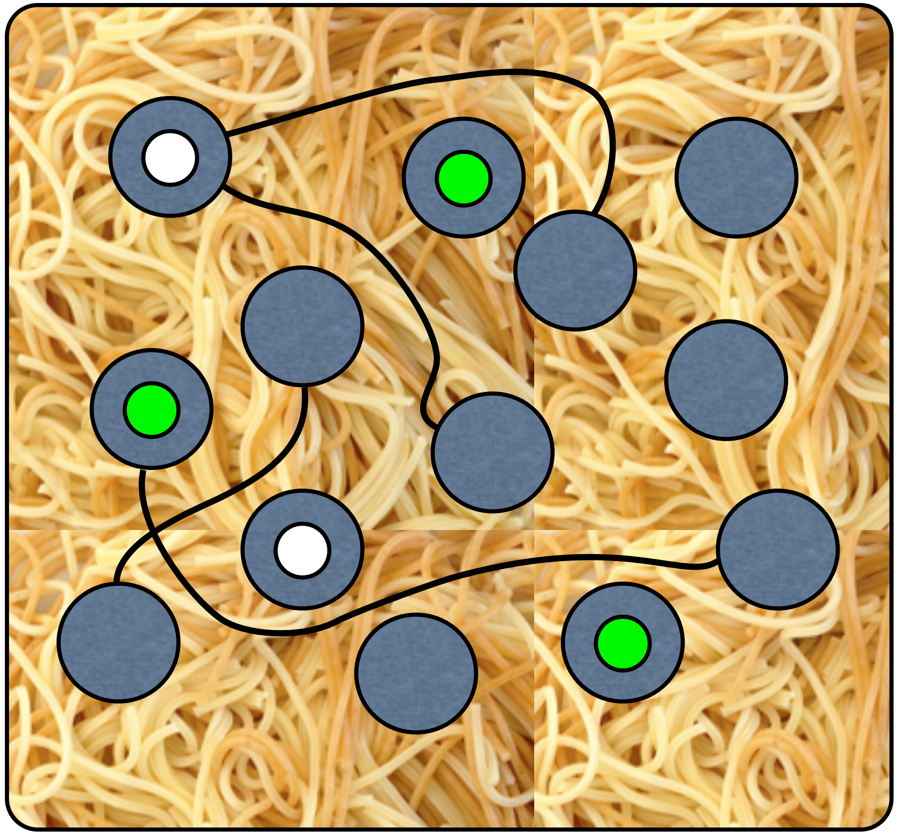
We give a set of rules,
and show that they generate all regular languages.

First we work with general NFA to show that the rules are **sound** – any language generated by the rules is regular.

Then we show that for any NFA, **M**, there is a DFA, $\mathcal{P}$(M), that recognises the same language. So, any language generated by the rules is recognised by some DFA.

Then we show that the rules are **complete** – any regular language is generated by the rules.

finite state spaghetti



A natural language is a set of finite sequences of words.

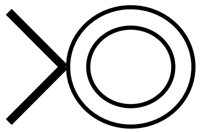
A formal language is a set of finite sequences of symbols.

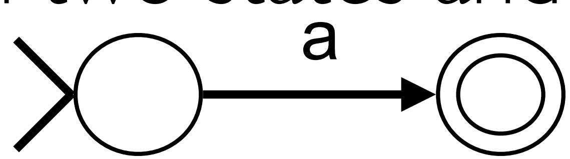
A formal language is **regular** iff it is the language recognised by some Finite State Machine.

KISS – start simple

NFA any number of start states and accepting states

Any NFA with no accepting states,
e.g. the NFA with no states,
recognises the empty language $\varnothing = \{\}$.

The NFA with one state - starting and accepting,
and no transitions  recognises the empty string $\{\varepsilon\}$.

The NFA with two states and one transition,
start to stop 

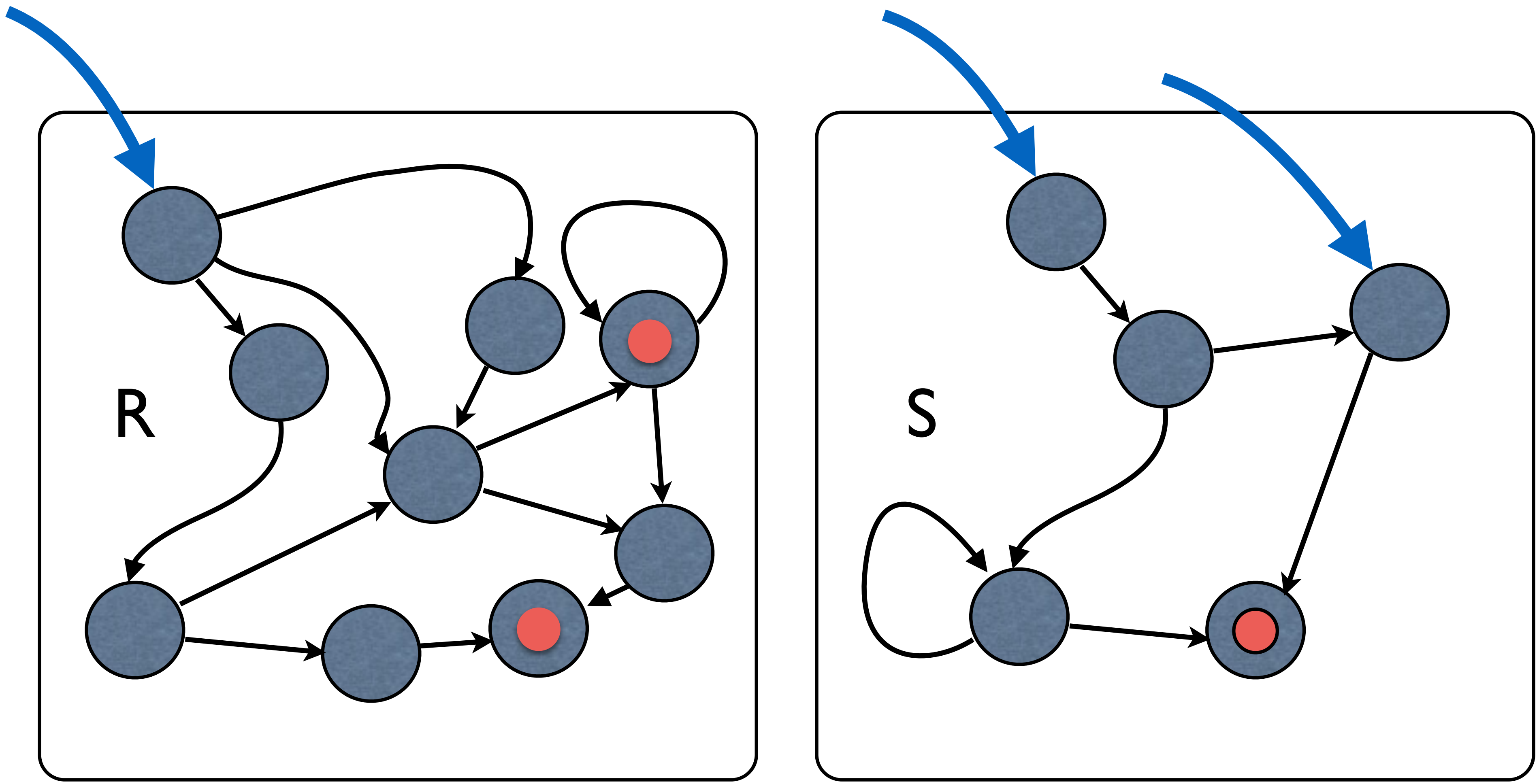
recognises {"a"}

KISS – the basic regular languages

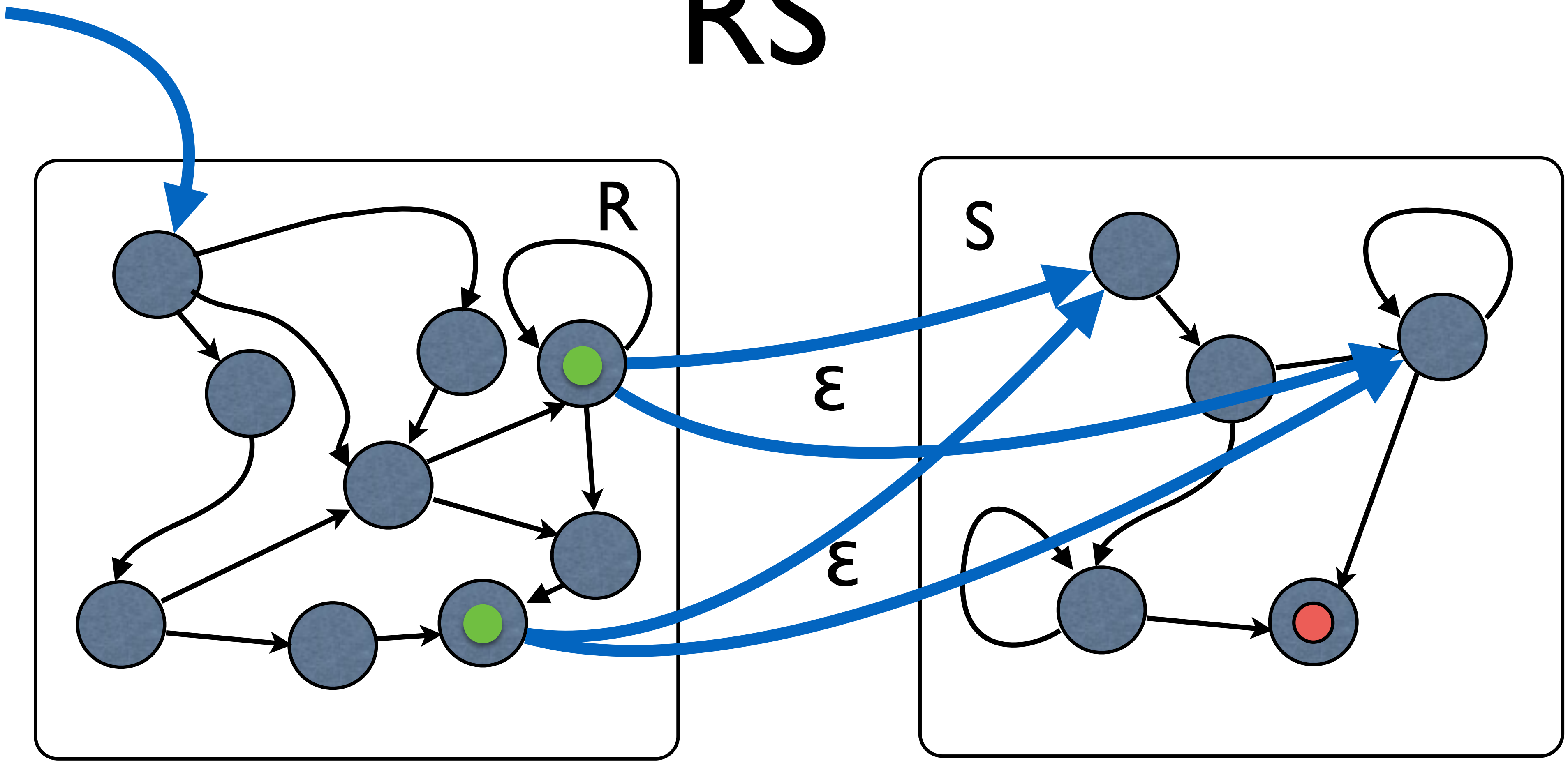
The following languages are regular

- The empty language $\varnothing = \{\}$
- The language that includes only the empty string $\{\varepsilon\}$.
- The language that includes only the one-letter string {"a"}

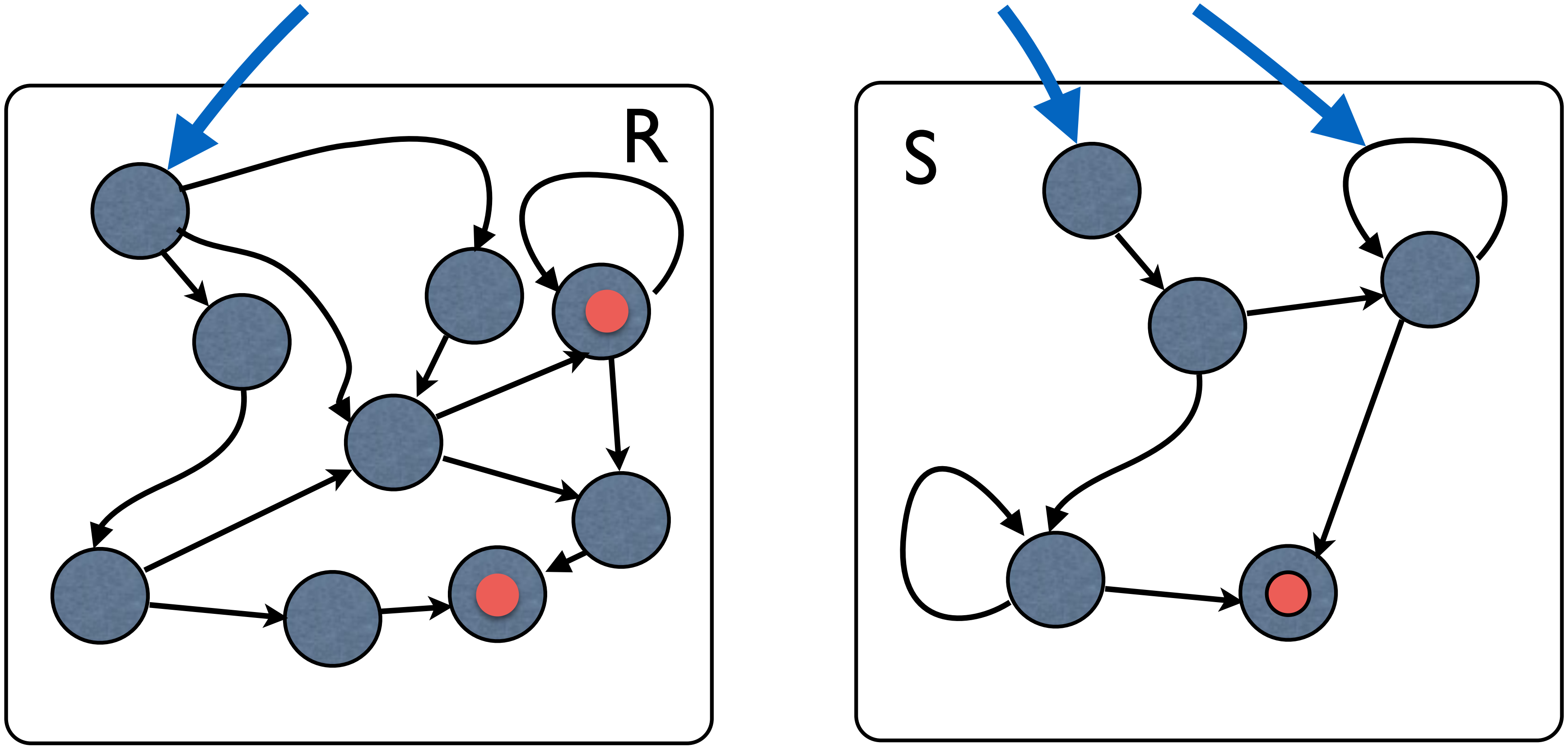
NFA any number of start states and accepting states



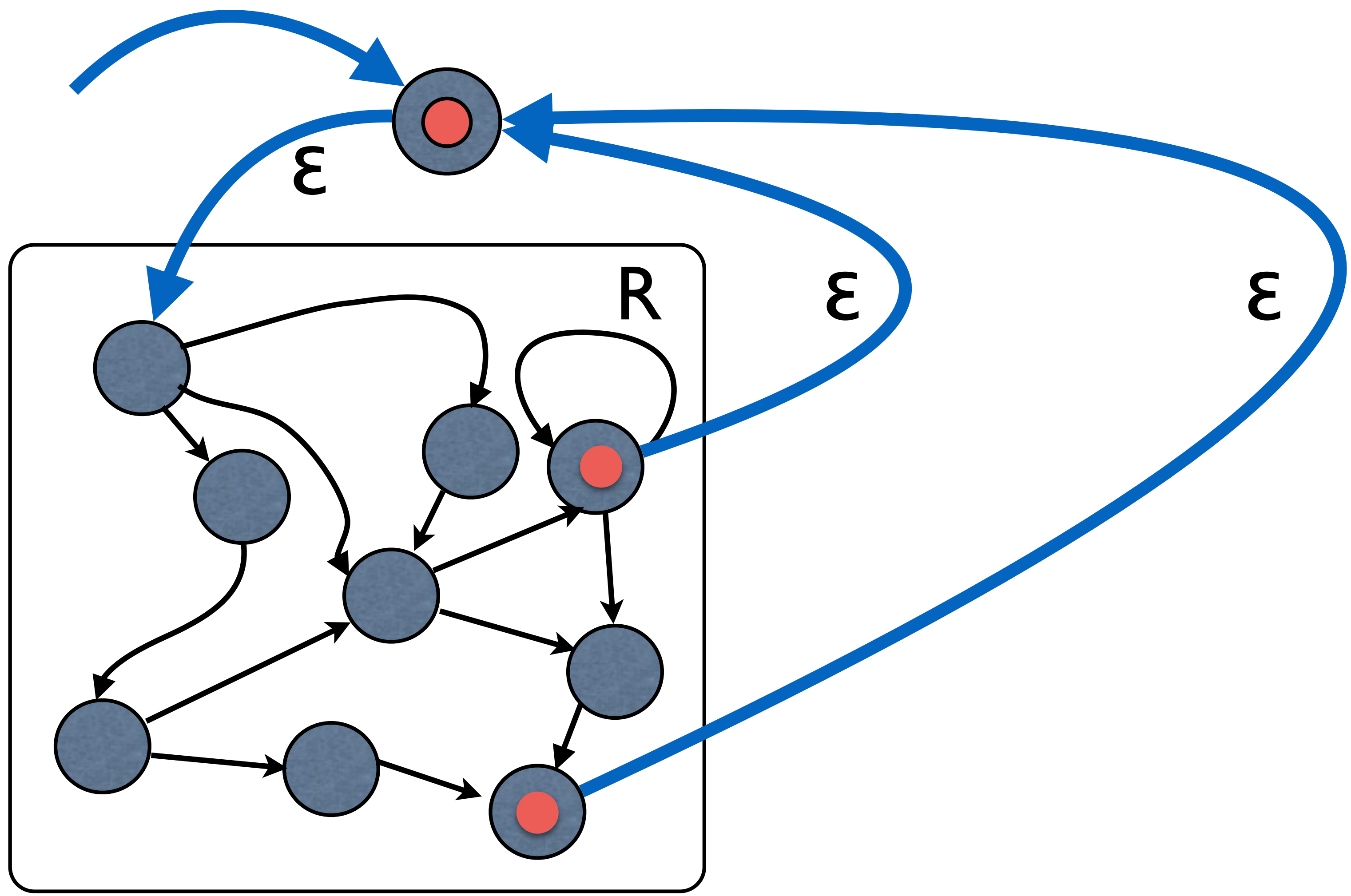
sequence
RS



alternation R|S



iteration R*



rules for regular languages

The following languages are regular

- The empty language $\varnothing = \{\}$
- The language that includes only the empty string $\{\boldsymbol{\varepsilon}\}$.
- The language that includes only the one-letter string {"a"}

If R and S are regular so are

- $R \mid S = R \cup S$
- $RS = \{ rs \mid r \in R \text{ and } s \in S \}$
- R^* generated by rules
  - $\boldsymbol{\varepsilon} \in R^*$
  - if $x \in R^*$ and $r \in R$ then $xr \in R^*$

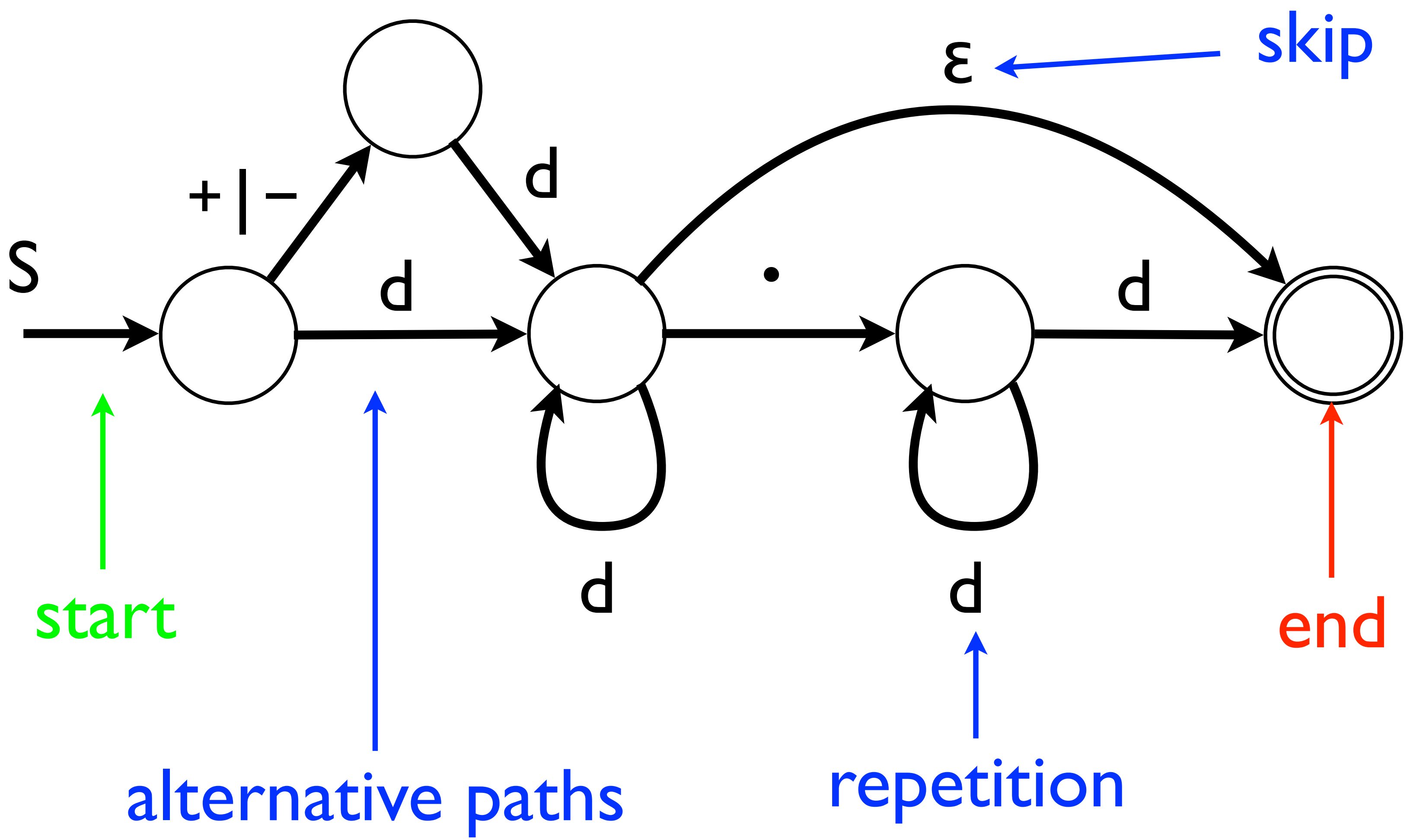
patterns for regular languages

regular expressions for sets of strings

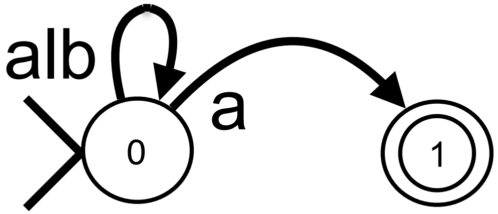
- **""** empty language $\varnothing = \{\}$
- **ε** empty string {ε}.
- **a** one-letter string {"a"}
- **R | S** union R ∪ S
- **RS** concatenation { rs | r ∈ R and s ∈ S }
- **R*** iteration

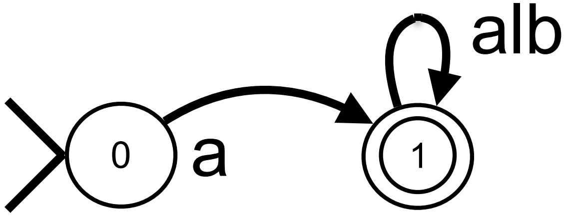
precedence: **R|ST* = R|(S(T*))**

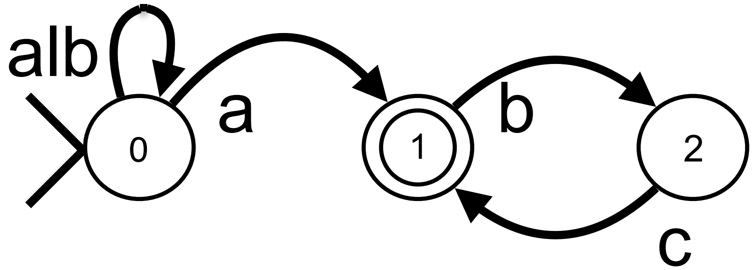
A Decimal Number

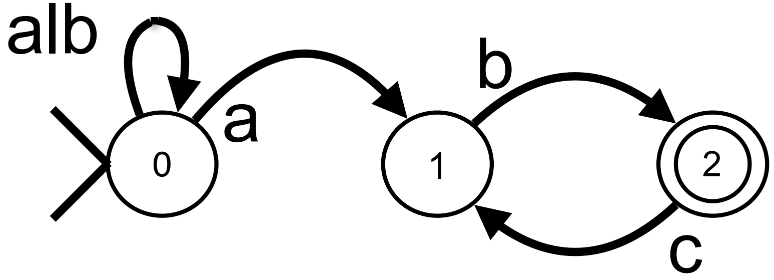


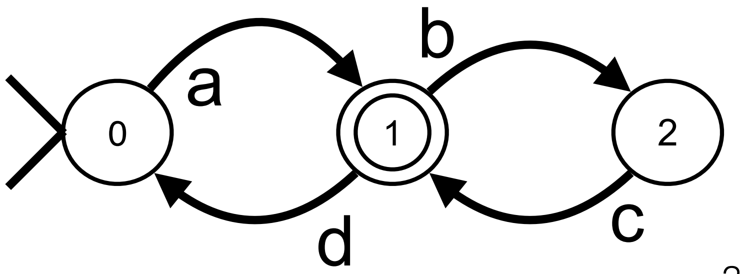
((+|-)\d|\d)\d*(ε|.\d*\d)   where \d is (0|1|2|3|4|5|6|7|8|9)

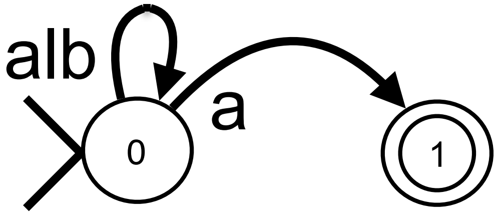

a|b
a
0
1


a|b
a
0
1

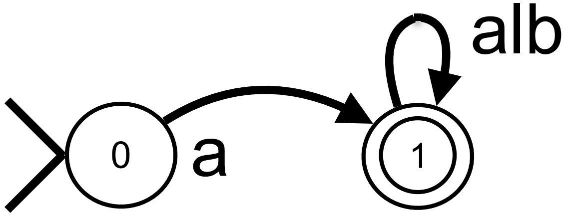

a|b
a
b
0
1
2
c


a|b
b
a
0
1
2
c

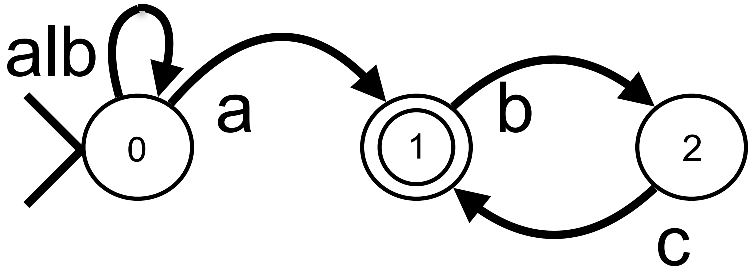

b
a
0
1
2
d
c



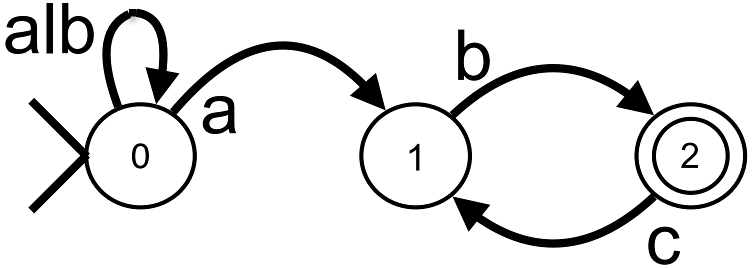
(a|b)*a



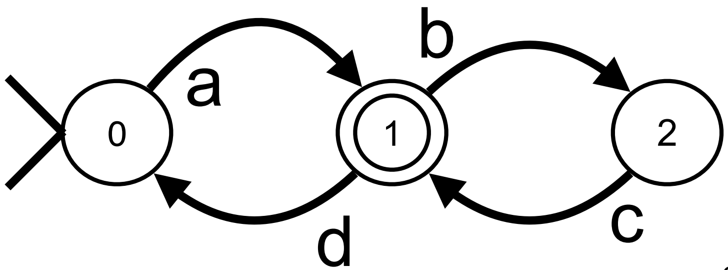
a(a|b)*



(a|b)*a(bc)*



(a|b)*ab(cb)*



a(da|bc)*

rules for regular languages

The following equations hold for any sets of strings **R,S,T**

- {} | S =
- {} S =
- ε S =
- ε* =
- {}* =
- R (S | T) =
- S R | T R =
- S* S | ε =

rules for regular languages

The following equations hold for any sets of strings **R,S,T**

- $\{\} \mid S = \{\} | S = S$
- $\{\}\ S = S\ \{\} = \{\}$
- $\varepsilon\ S = S\ \varepsilon = S$
- $\varepsilon^* = \varepsilon$
- $\{\}^* = \{\}$
- $R\ (S \mid T) = R\ S \mid R\ T$
- $(S \mid T)\ R = S\ R \mid T\ R$
- $S^* = S^*\ S \mid \varepsilon = S\ S^* \mid \varepsilon$