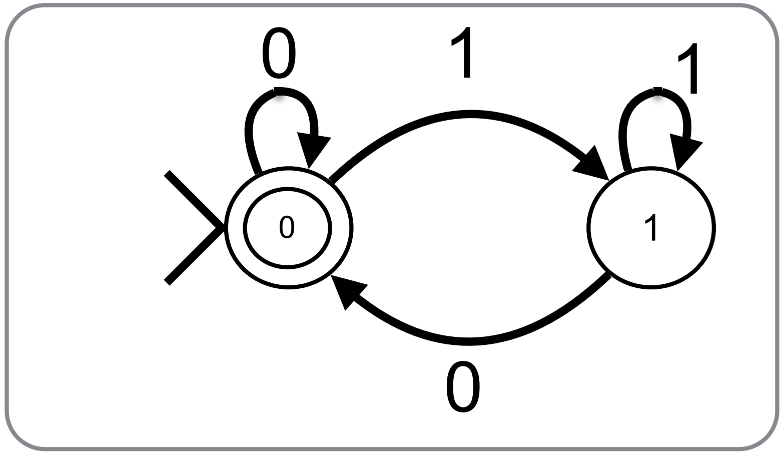
NFA and regex

cl

- the Boolean algebra of languages
- regular expressions

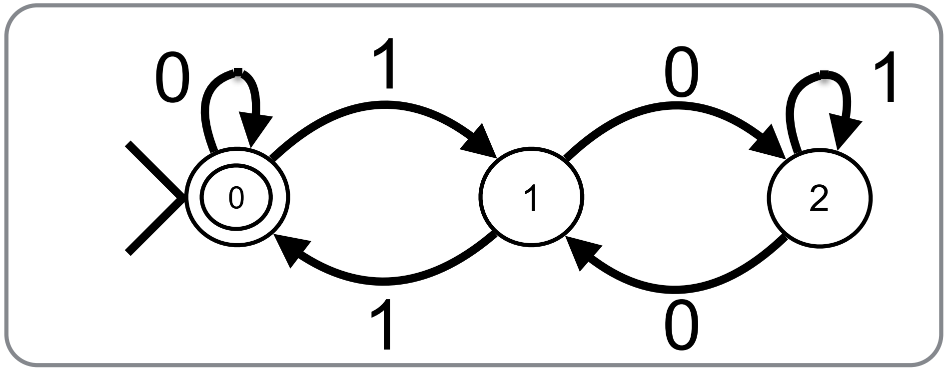
The intersection of two regular languages is regular



L_0 = even numbers
L_1 = odd numbers

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

The intersection of two regular languages is regular



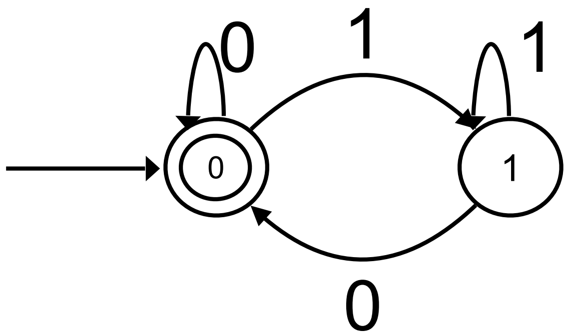
$L_0 = 0 \bmod 3$

$L_1 = 1 \bmod 3$

$L_2 = 2 \bmod 3$

Two examples

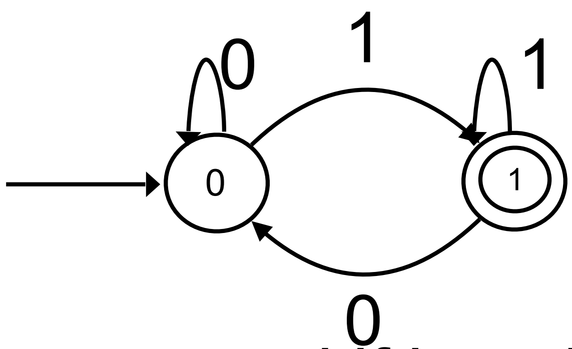
	×2	×2 + 1
	0	**1**
0	0	1
1	0	1



Even binary numbers

Input sequence is accepted if it ends with a zero.

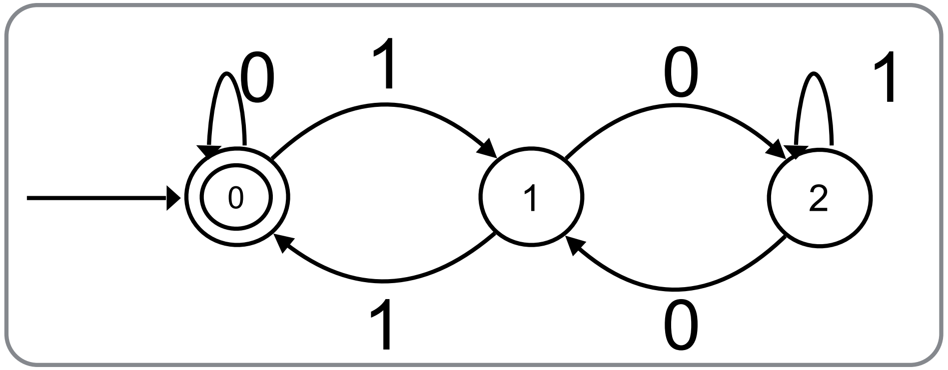
	×2	×2 + 1
	0	**1**
0	0	1
1	0	1



Odd binary numbers

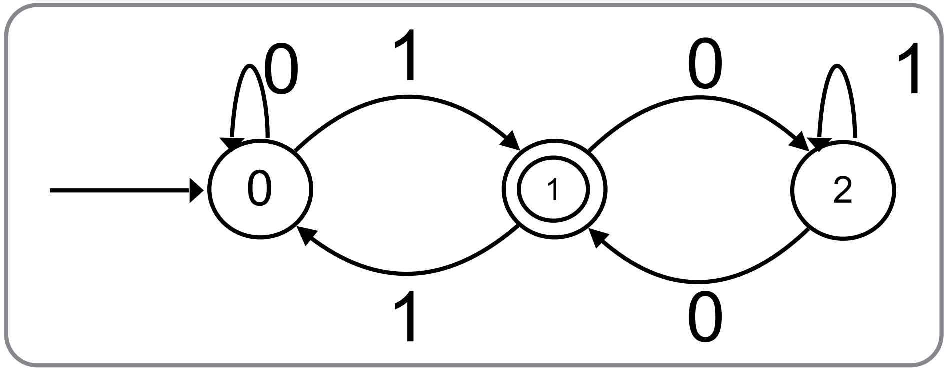
Input sequence is accepted if it ends with a one.

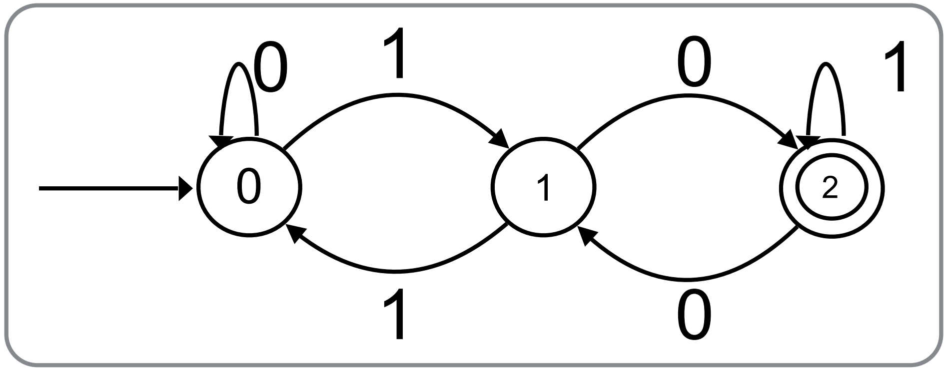
Three examples



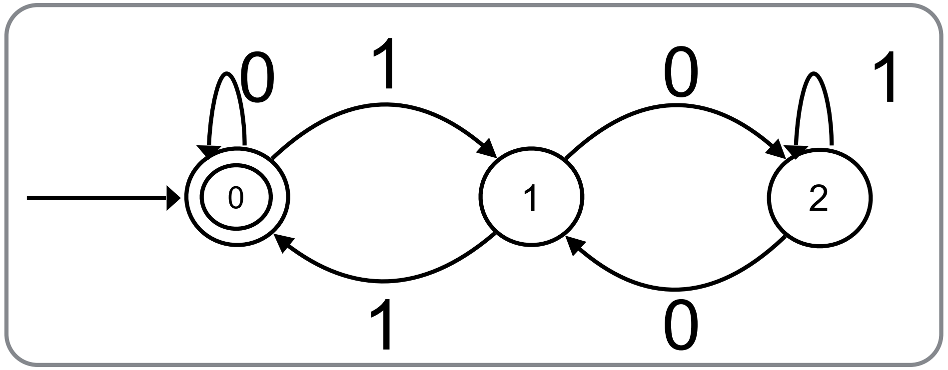
Which binary numbers are accepted?

mod 3	×2 0	×2 + 1 1
0	0	1
1	2	0
2	1	2

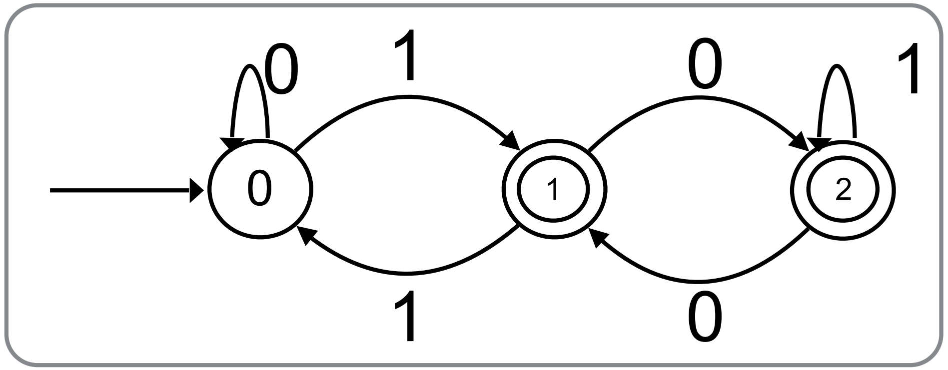




By three or not by three?

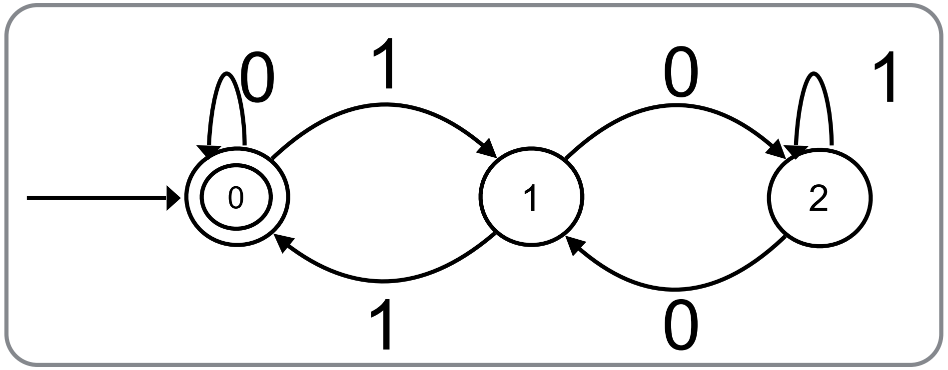


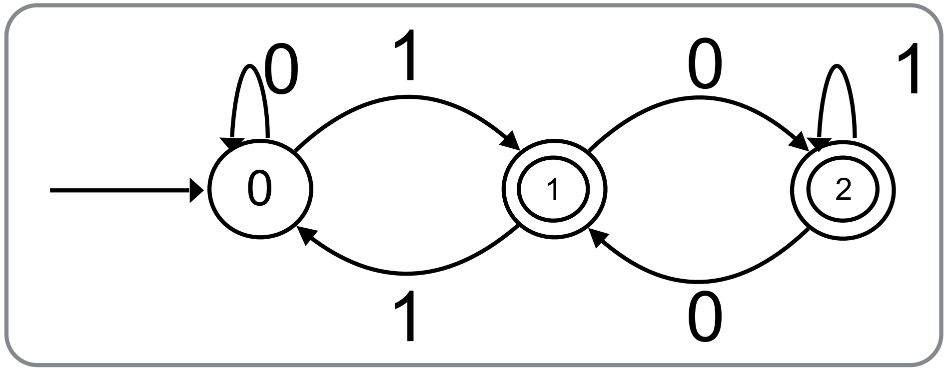
divisible by three



not
divisible by three

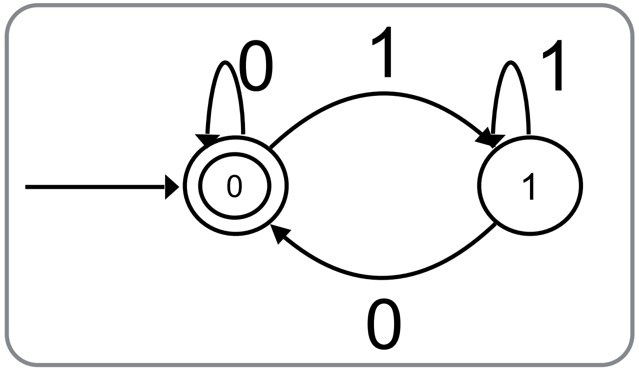
The complement of a regular language is regular

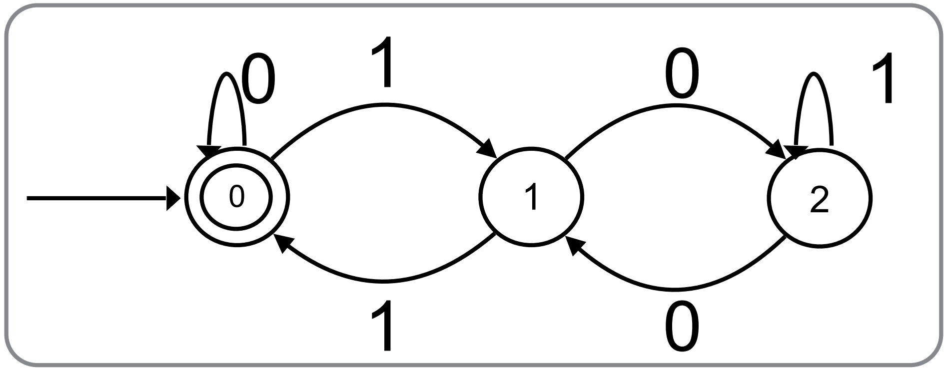




If $A \subseteq \Sigma^*$ is recognised by M then $\overline{\mathbf{A}} = \Sigma^* \setminus A$ is recognised by $\overline{\mathbf{M}}$ where $\overline{\mathbf{M}}$ and **M** are identical except that the accepting states of $\overline{\mathbf{M}}$ are the non-accepting states of **M** and vice-versa

The intersection of two regular languages is regular





divisible by 6

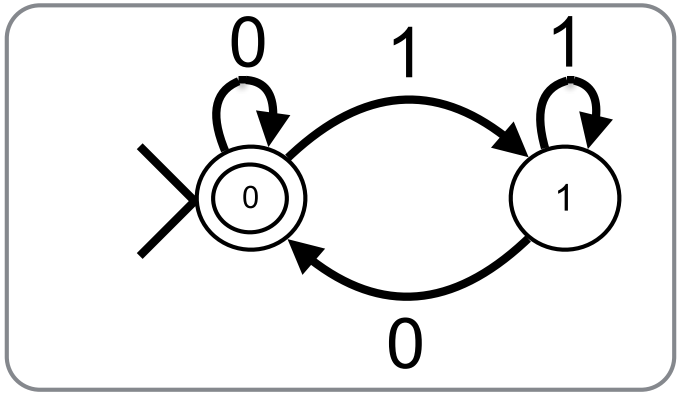
≣

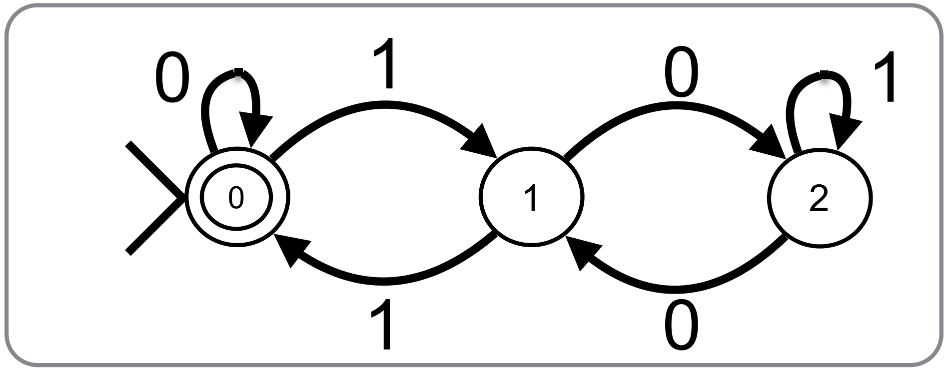
divisible by 2

and

divisible by 3

The intersection of two regular languages is regular



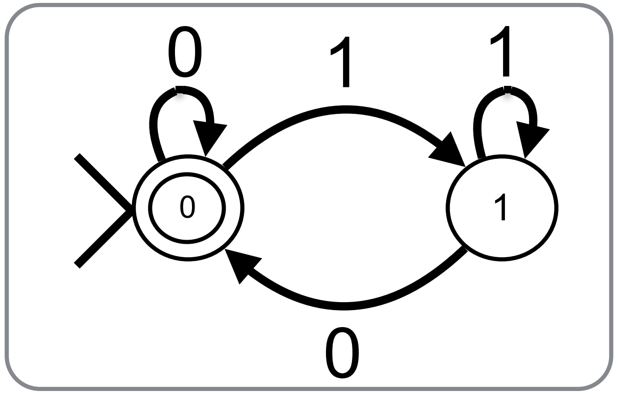


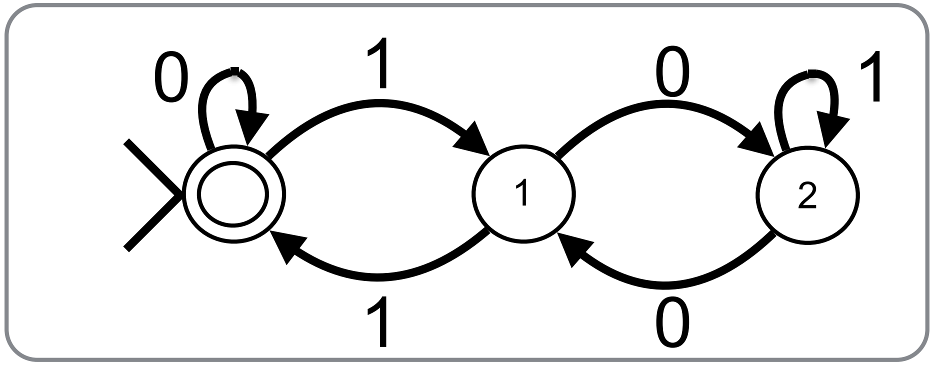
Run both machines in parallel?

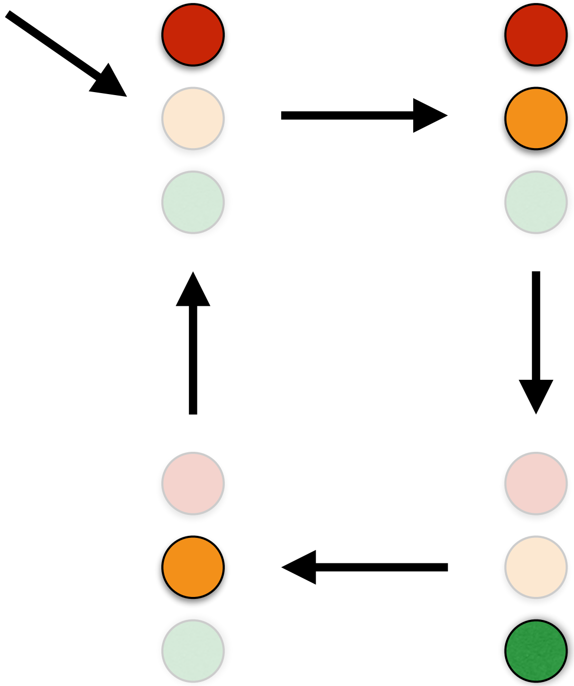
Build one machine that simulates two machines running in parallel!

Keep track of the state of each machine.

The intersection of two regular languages is regular

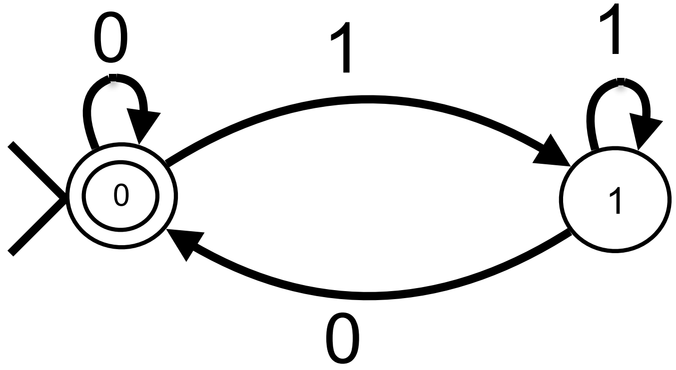


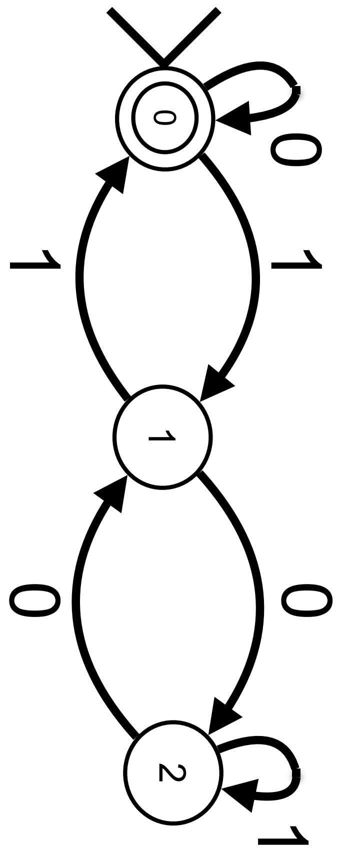


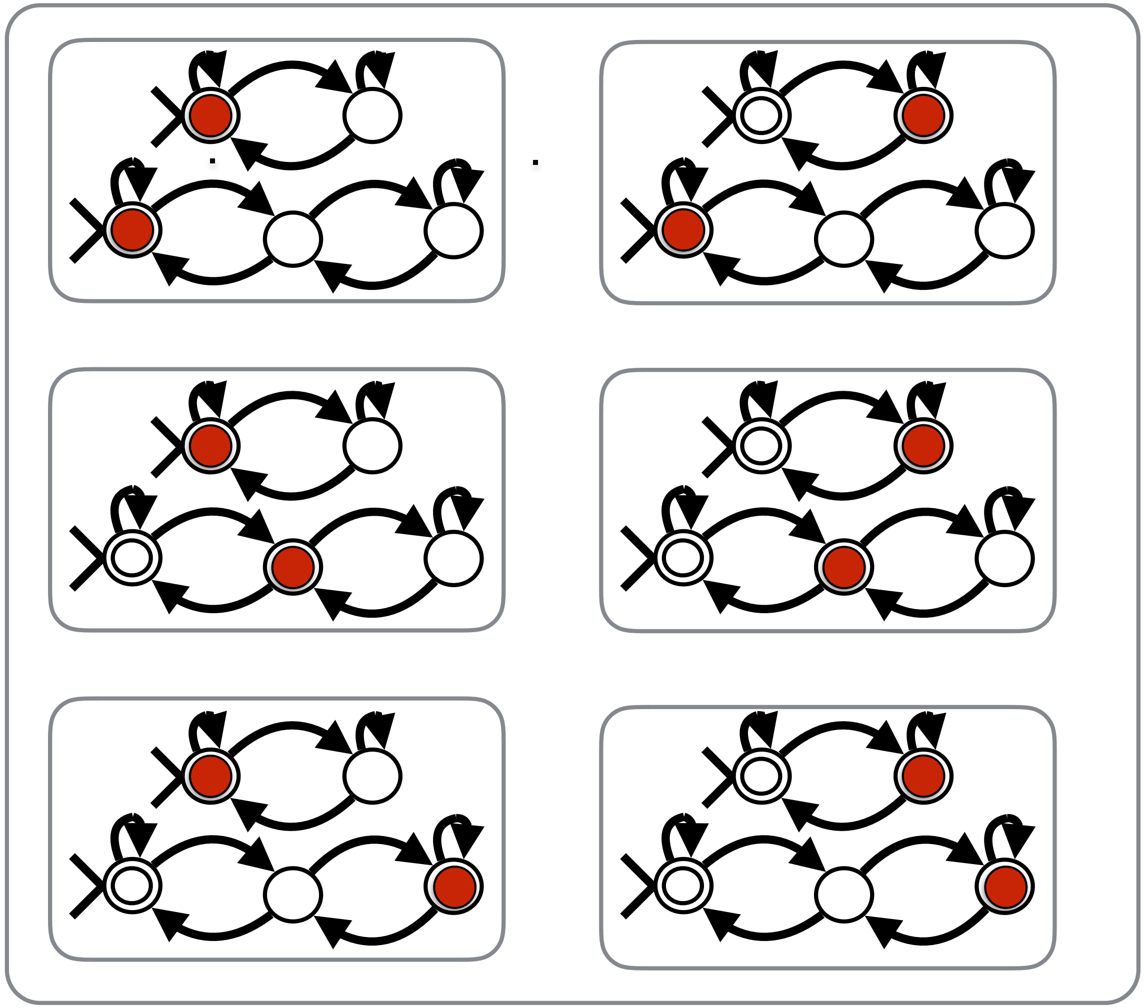


intersection of languages

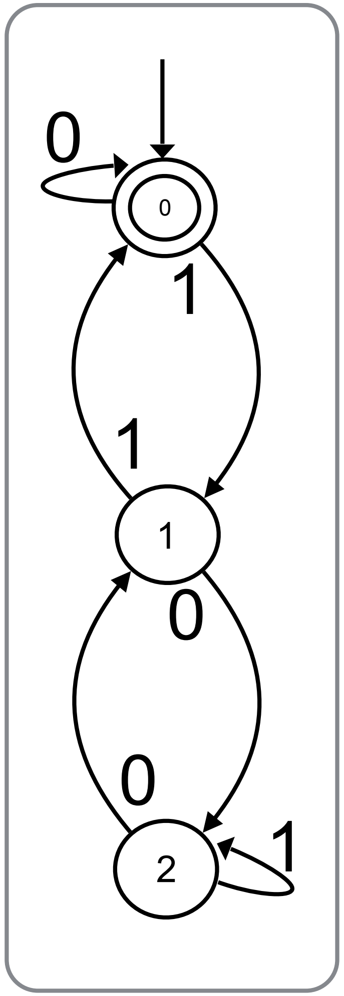
run the two machines in parallel
when a string is in both languages,
both are in an accepting state



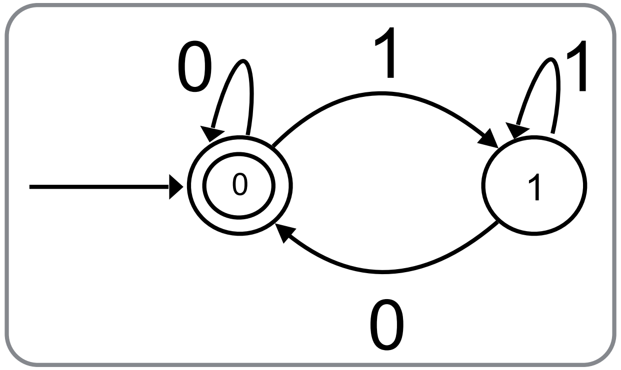




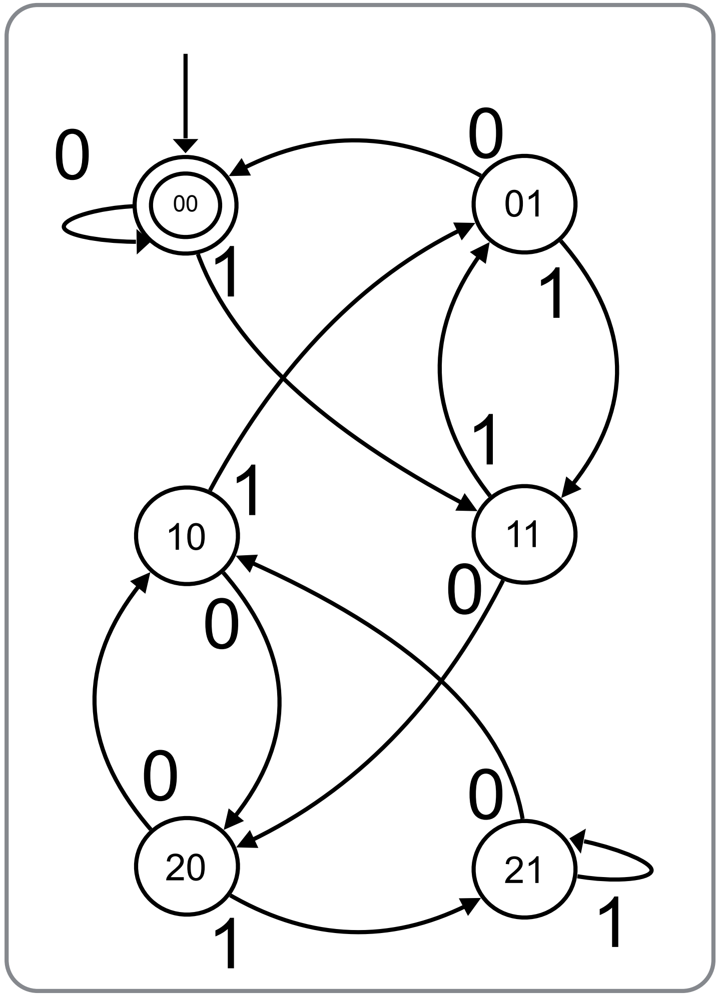
The intersection of two regular languages is regular



||

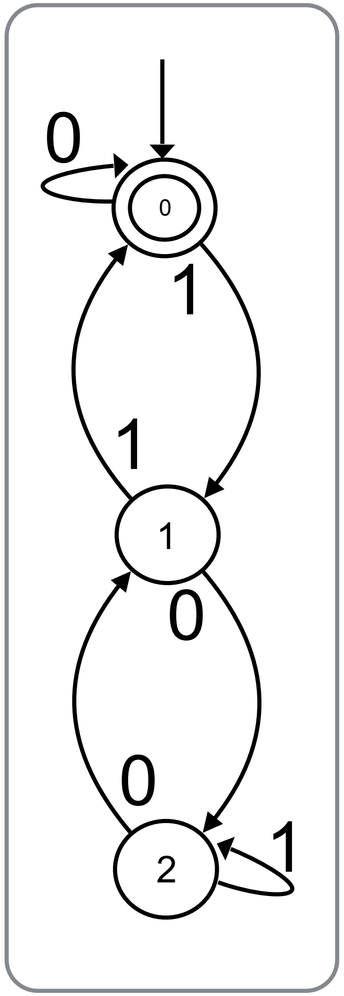


=

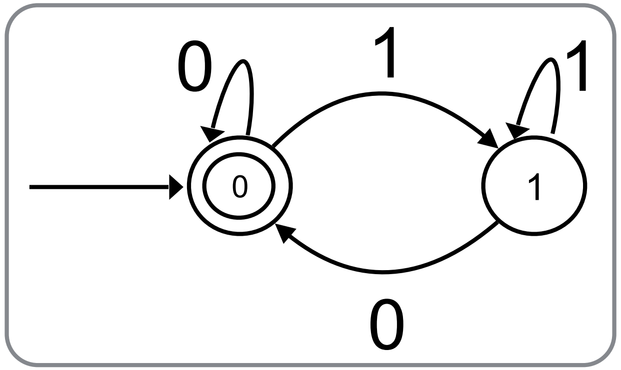


mod 3	mod 2	0		1	
		3	2	3	2
0	0	0	0	1	1
1	0	2	0	0	1
2	0	1	0	2	1
0	1	0	0	1	1
1	1	2	0	0	1
2	1	1	0	2	1

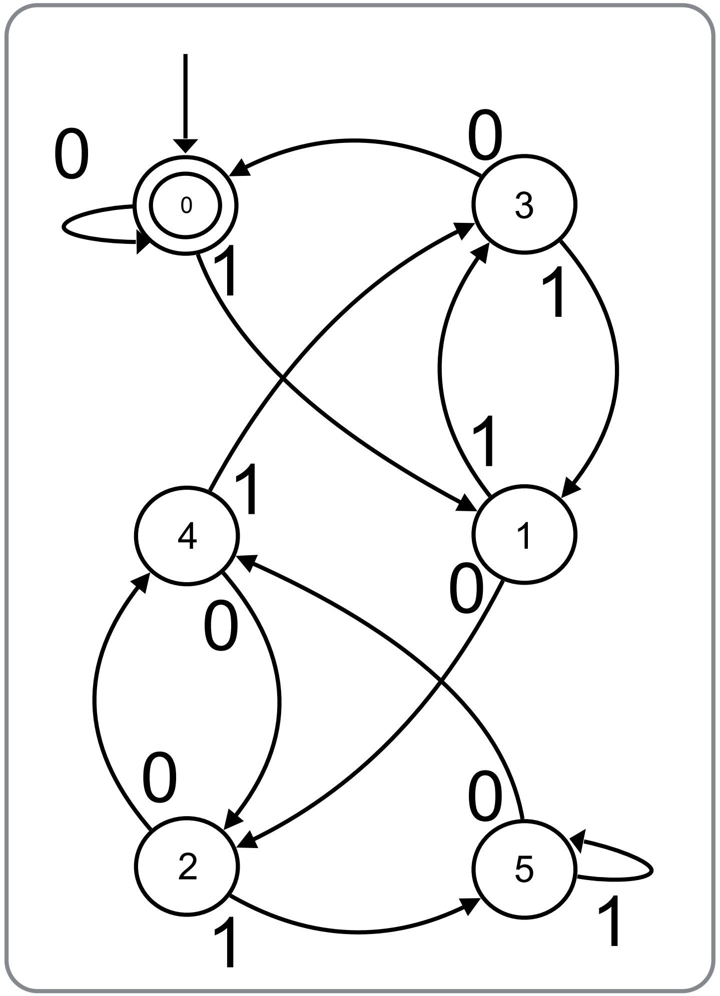
The intersection of two regular languages is regular



||

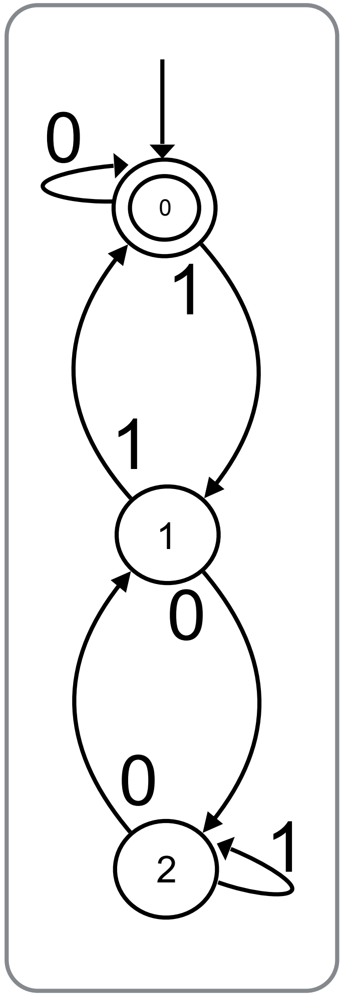


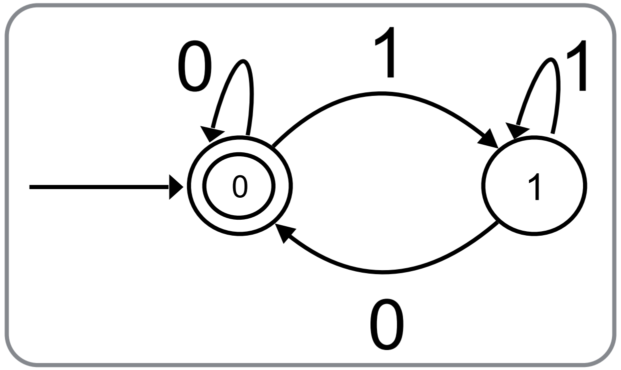
=

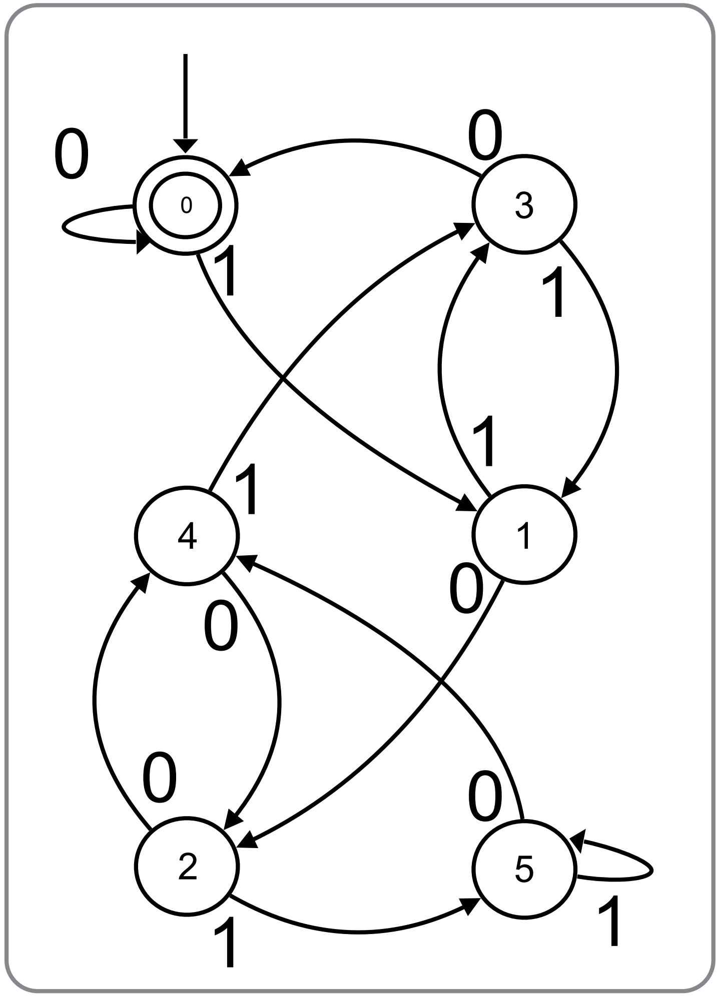


mod 3	mod 2	mod 6	0			1		
			3	2	6	3	2	6
0	0	0	0	0	0	1	1	1
1	0	4	2	0	2	0	1	3
2	0	2	1	0	4	2	1	5
0	1	3	0	0	0	1	1	1
1	1	1	2	0	2	0	1	3
2	1	5	1	0	4	2	1	5

The intersection of two regular languages is regular







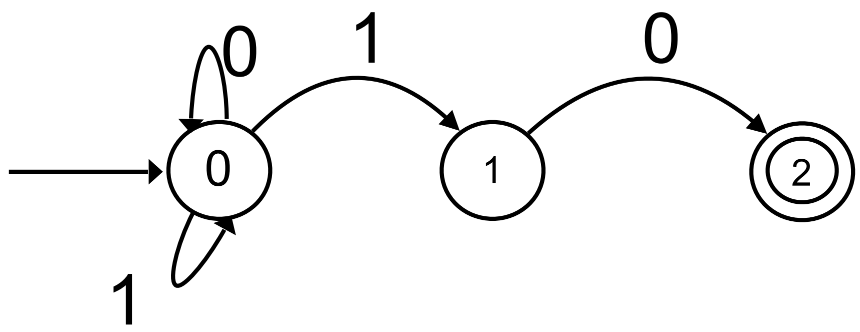
mod 3	mod 2	mod 6	0: 3	0: 2	0: 6	1: 3	1: 2	1: 6
0	**0**	0	0	0	0	1	1	1
1	**1**	1	2	0	2	0	1	3
2	**0**	2	1	0	4	2	1	5
0	**1**	3	0	0	0	1	1	1
1	**0**	4	2	0	2	0	1	3
2	**1**	5	1	0	4	2	1	5

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

- Since they are closed under intersection and complement.

Non Determinism

In a non-deterministic machine (NFA), each state may have any number of transitions with the same input symbol, leaving to different successor states.



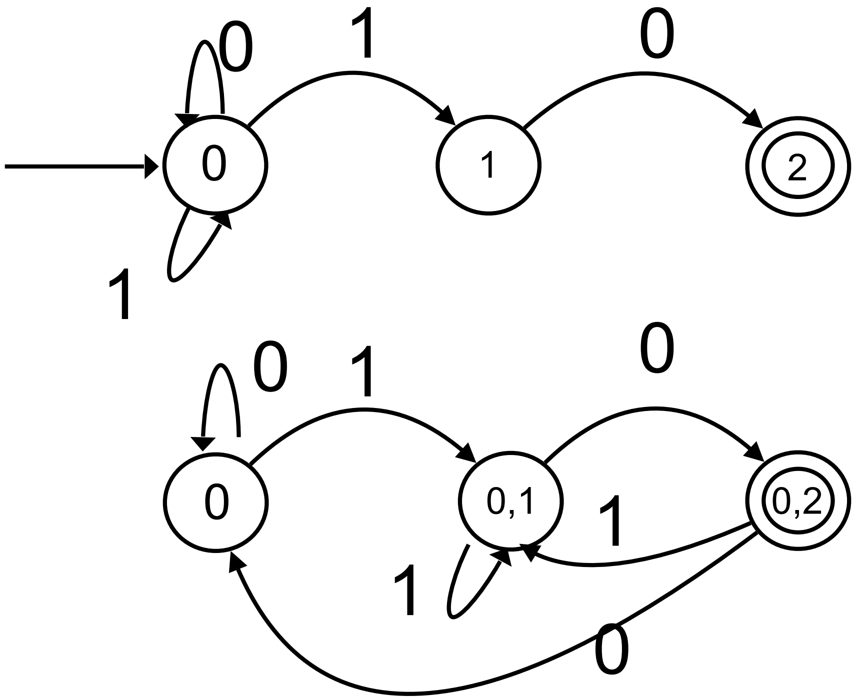
In a non-deterministic machine (NFA), each state may lead to any number of transitions with the same input symbol, leaving to different successor states.

	0	1
0	0	0,1
1	2	
2		

We have a transition relation

Non Determinism

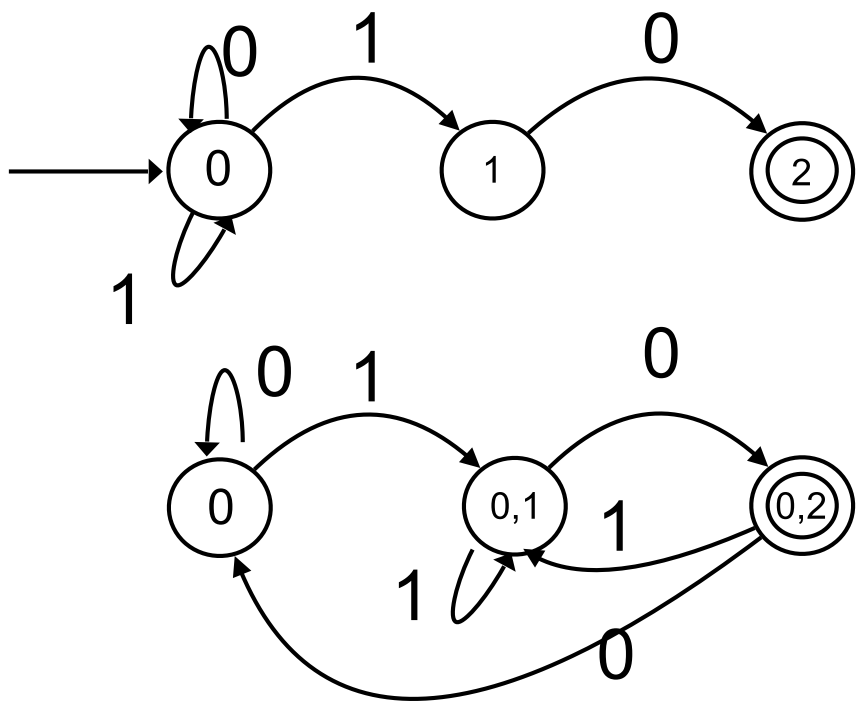
In a non-deterministic machine (NFA), each state may have any number of transitions with the same input symbol, leaving to different successor states.



	0	**1**
0	0	0,1
1	2	
2		
0,1	0,2	0,1
0,2	0	0,1

Non Determinism

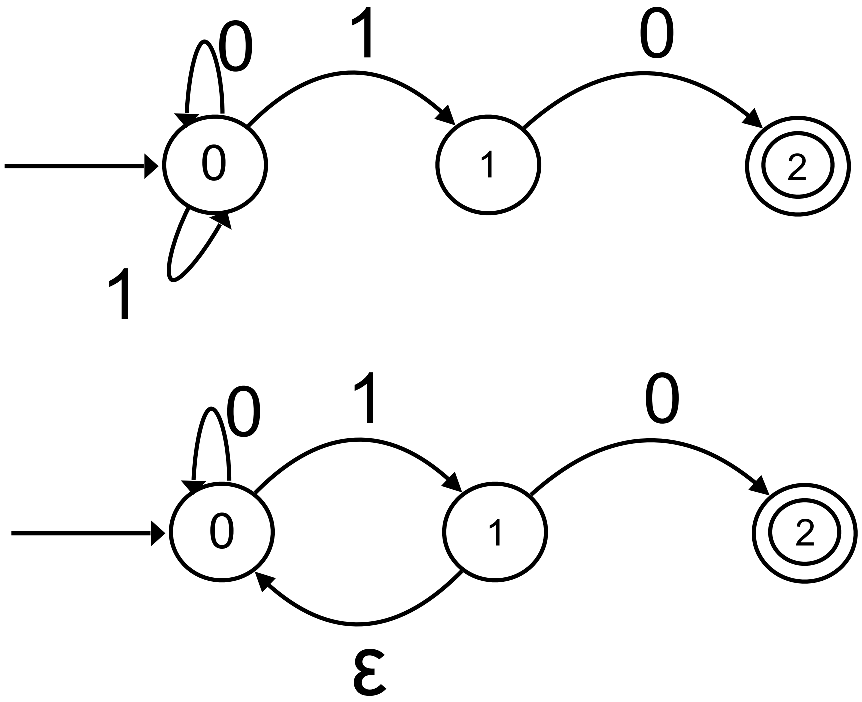
We can simulate a non-deterministic machine using a deterministic machine – by keeping track of the set of states the NFA could possibly be in.



	0	1
0	0	0,1
1	2	
2		
0,1	0,2	0,1
0,2	0	0,1

Internal Transitions

We sometimes add an internal transition ε to a non-deterministic machine (NFA)This is a state change that consumes no input.



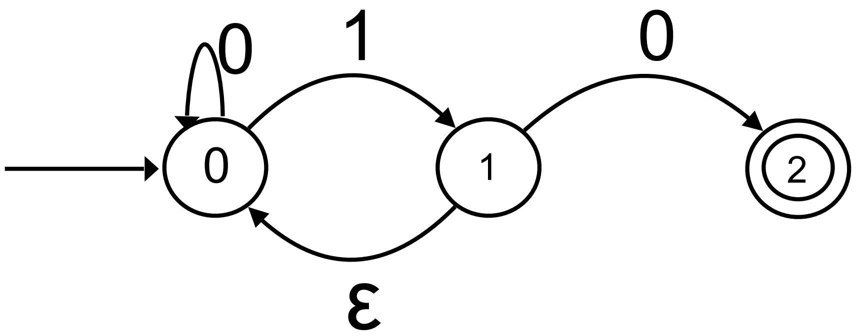
	0	1	ε
0	0	1	
1	2		0
2			

Internal Transitions

We sometimes add **internal transitions** – labelled ε – to a non-deterministic machine (NFA).

This is a state change that consumes no input.

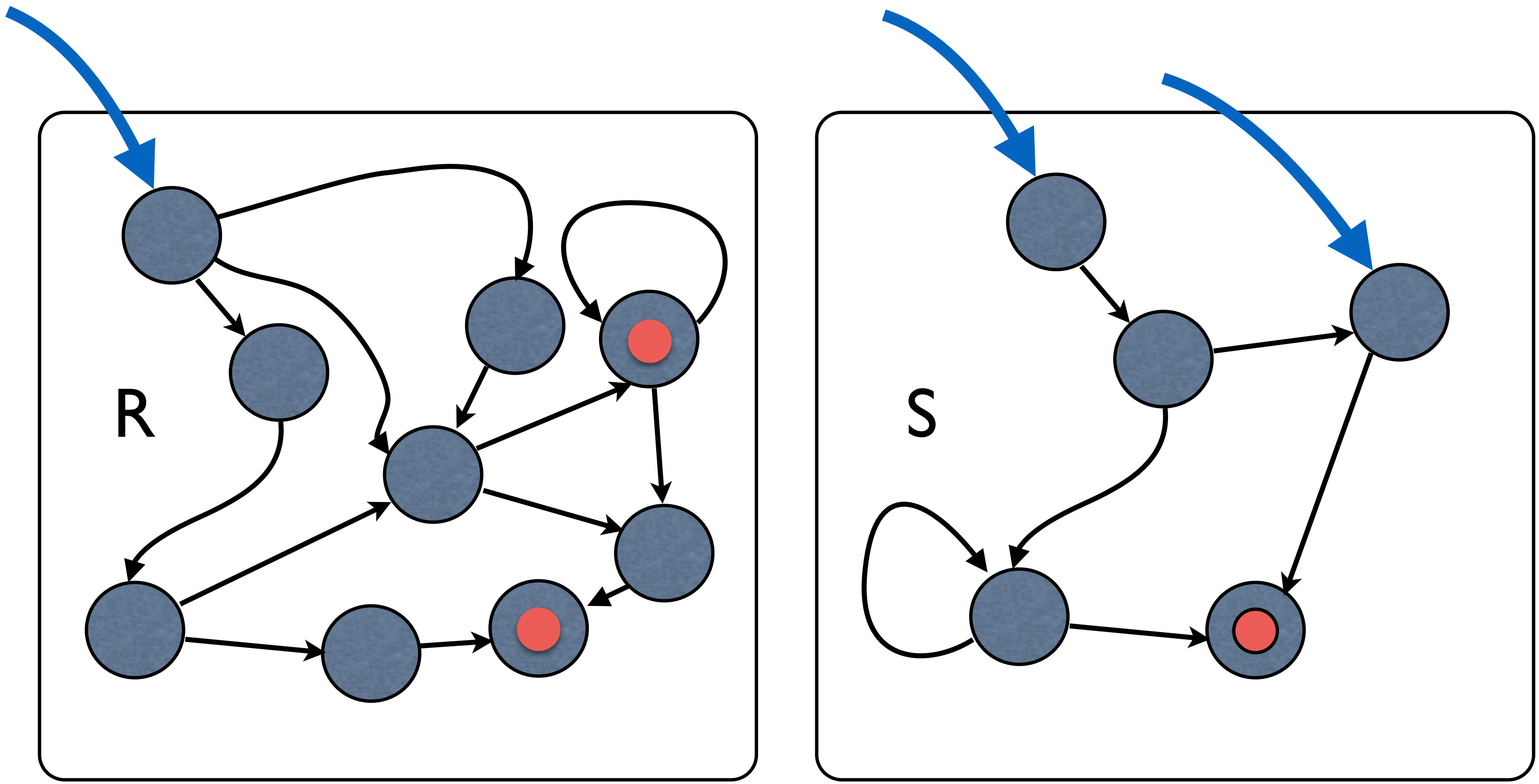
It introduces non-determinism in the observed behaviour of the machine.



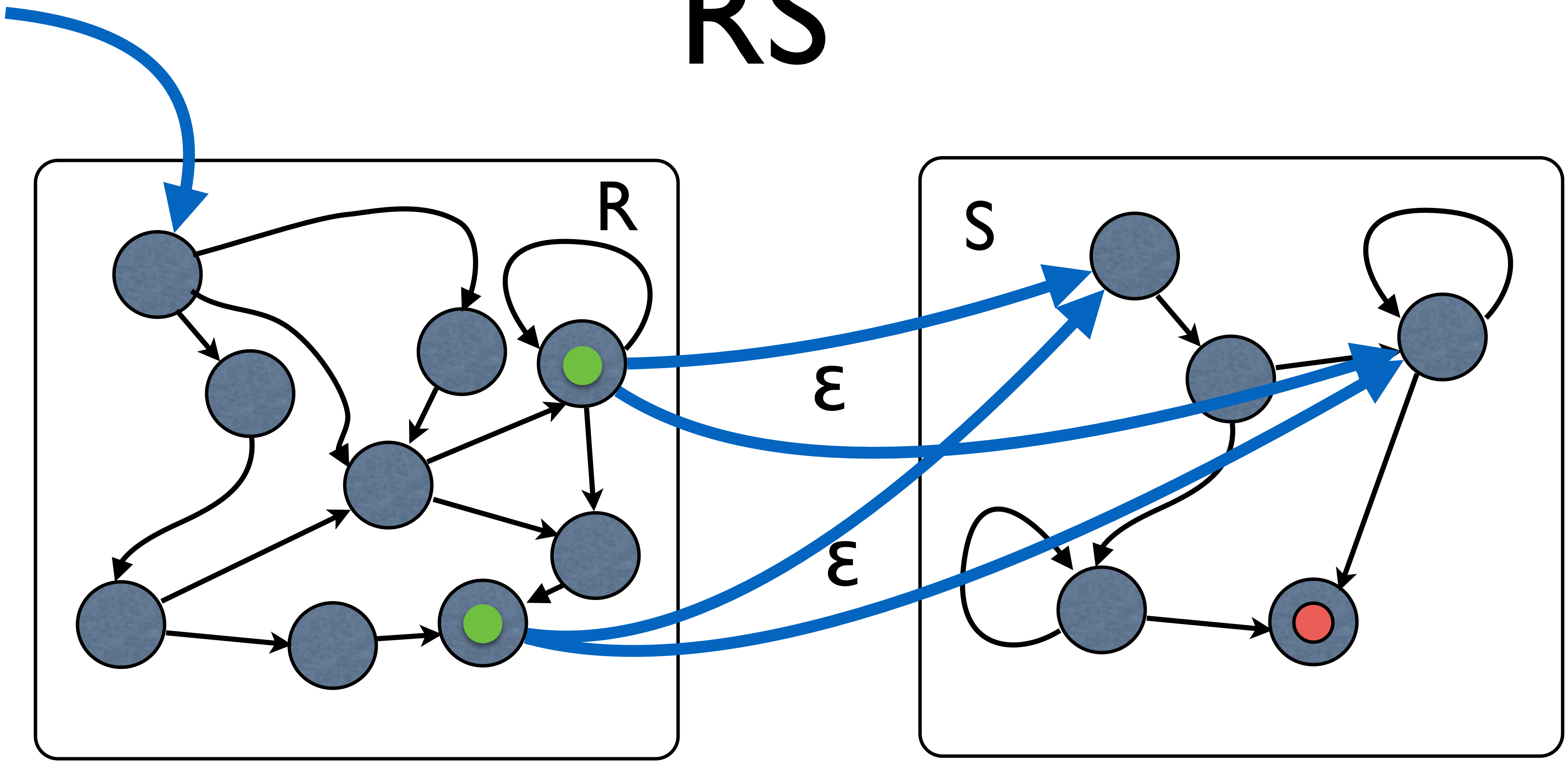
	0	1	ε
0	0	1	
1	2		0
2			

	0ε*	1ε*
0	0	1,0
1	2	
2		

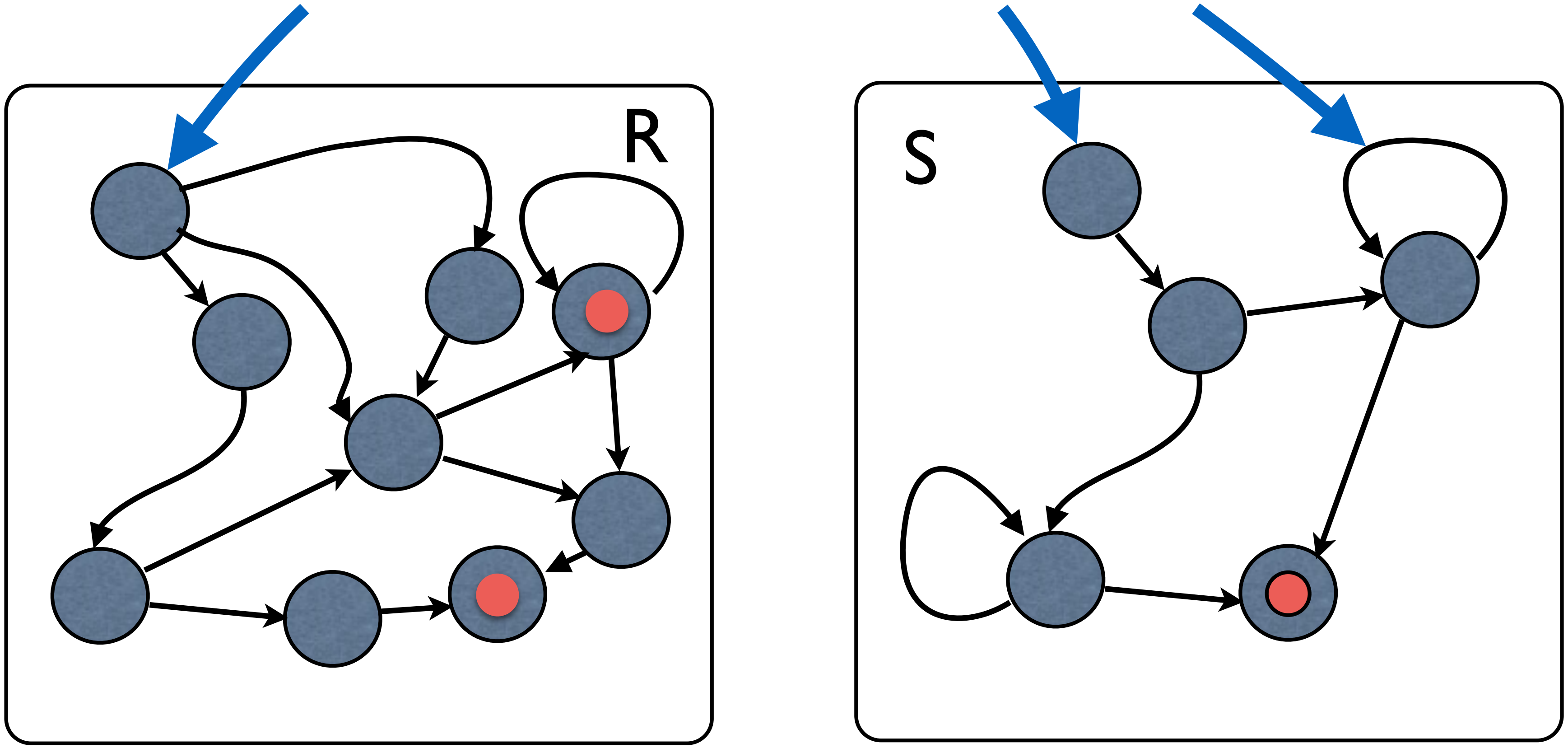
NFA any number of start states and accepting states



sequence
RS



alternation R|S



iteration R*