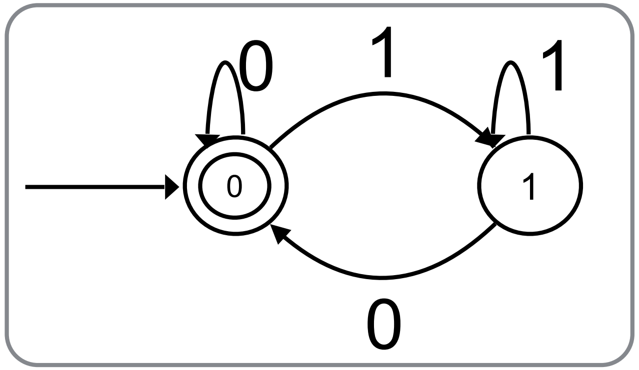
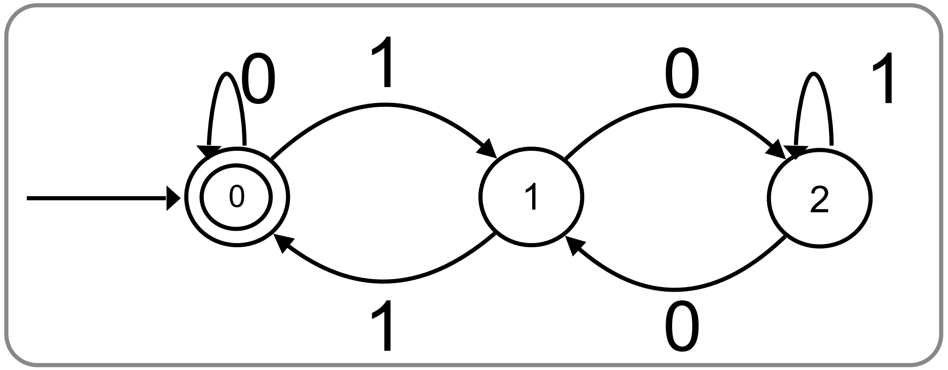
NFA and regex

cl

- the Boolean algebra of languages
- non-deterministic machines
- regular expressions

The intersection of two regular languages is regular



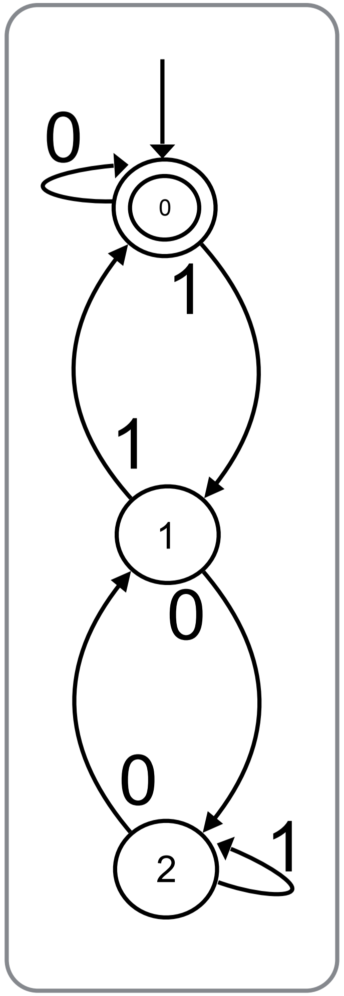


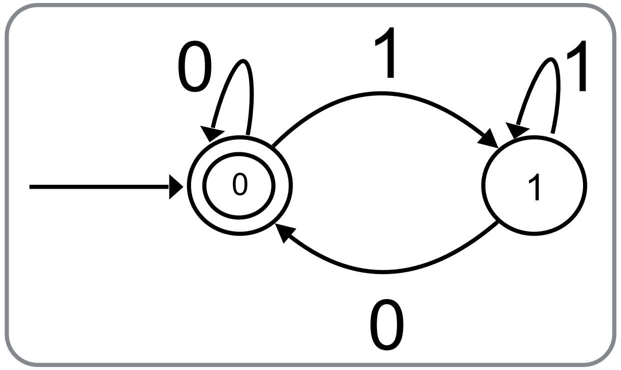
Run both machines in parallel?

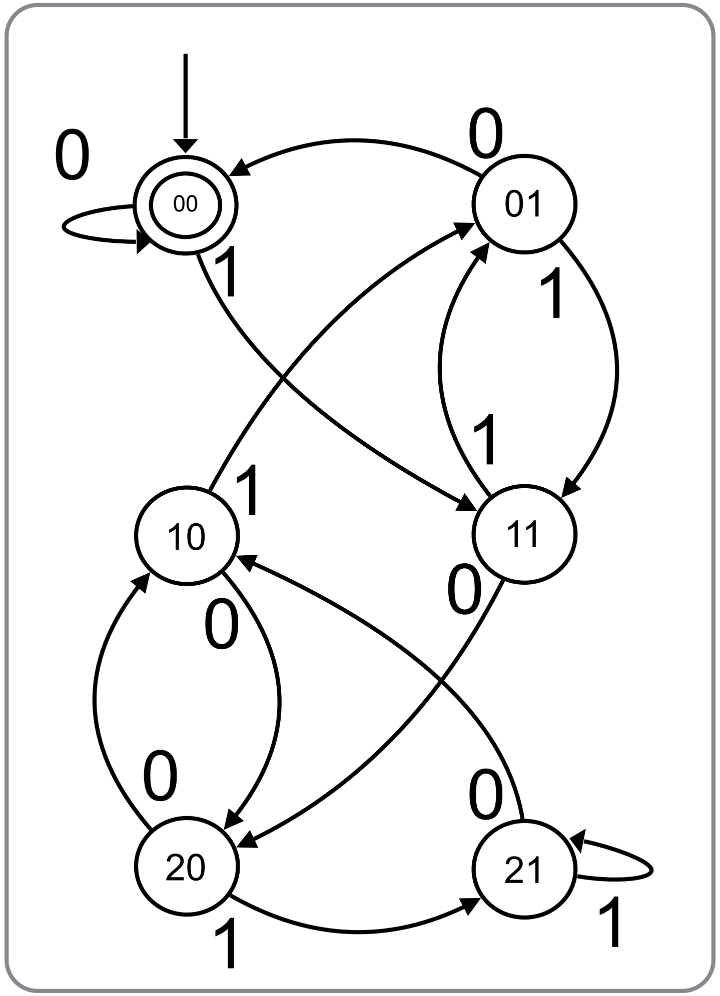
Build one machine that simulates two machines running in parallel!

Keep track of the state of each machine.

The intersection of two regular languages is regular

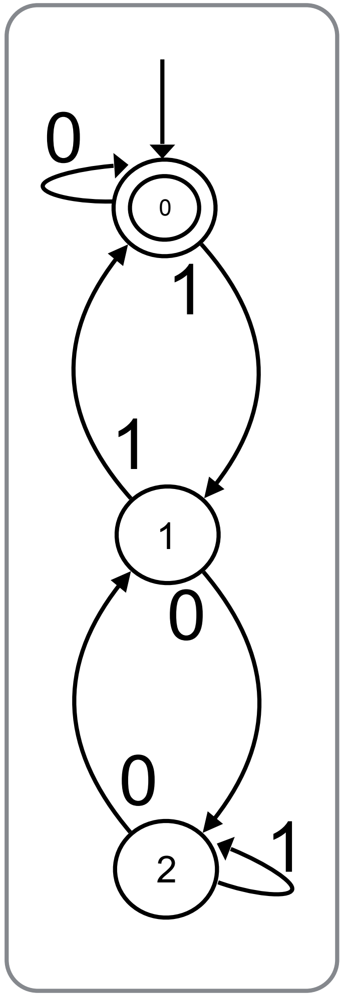




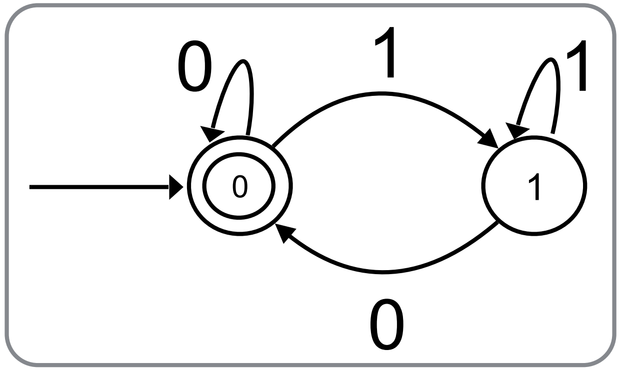


mod 3	mod 2	0		1	
		3	2	3	2
0	0	0	0	1	1
1	0	2	0	0	1
2	0	1	0	2	1
0	1	0	0	1	1
1	1	2	0	0	1
2	1	1	0	2	1

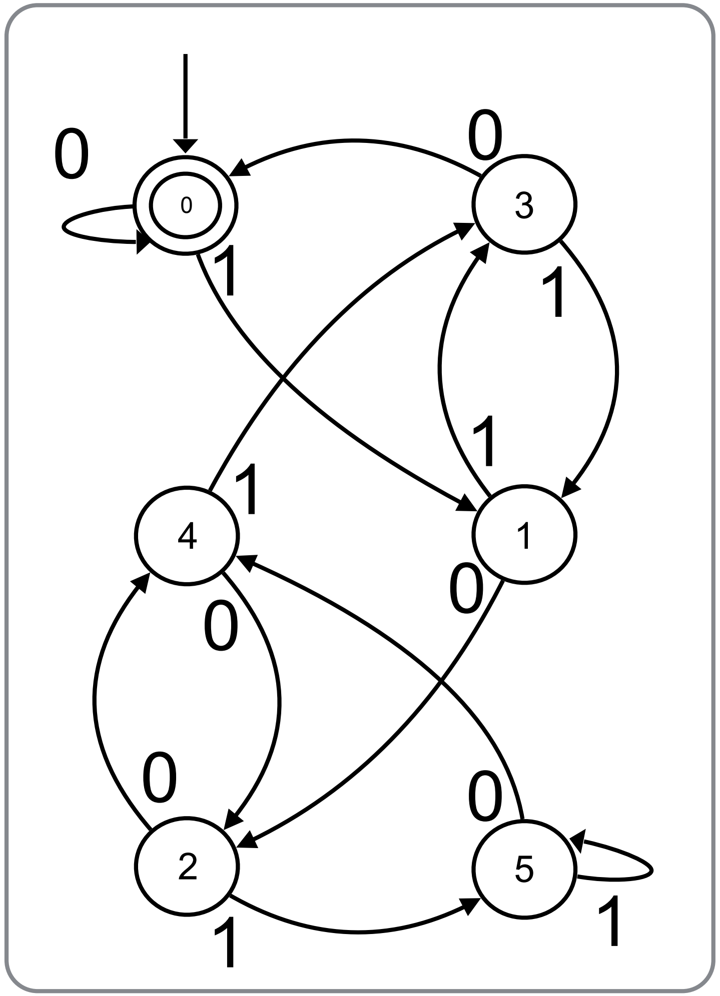
The intersection of two regular languages is regular



||

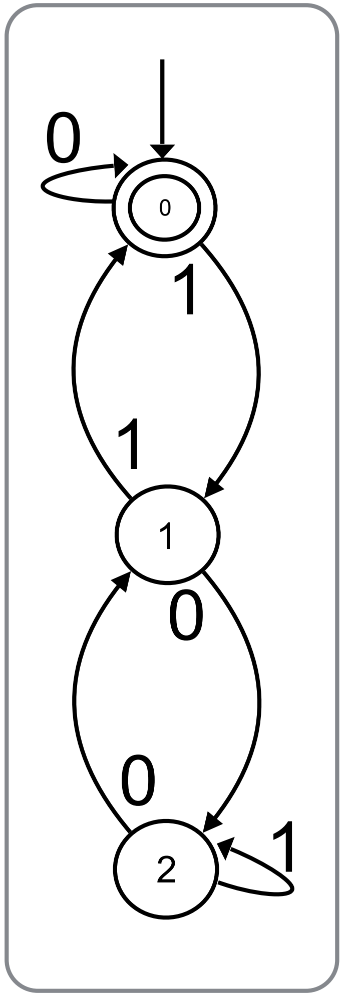


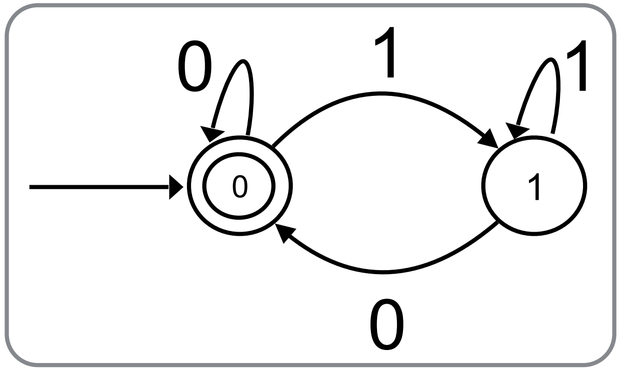
=

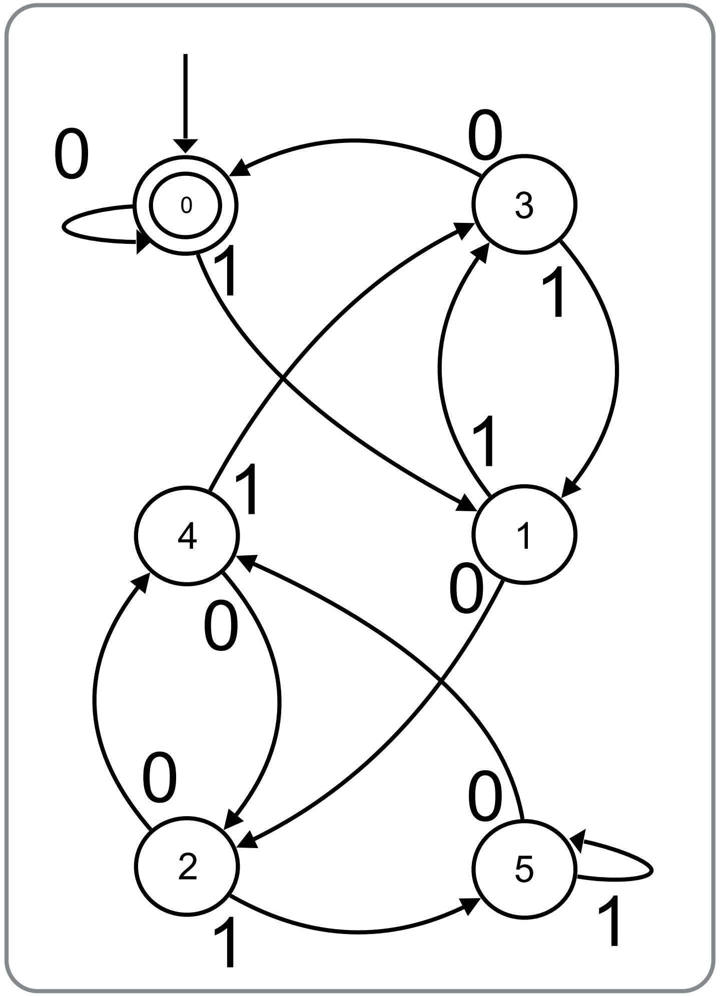


mod 3	mod 2	mod 6	0			1		
			3	2	6	3	2	6
0	0	0	0	0	0	1	1	1
1	0	4	2	0	2	0	1	3
2	0	2	1	0	4	2	1	5
0	1	3	0	0	0	1	1	1
1	1	1	2	0	2	0	1	3
2	1	5	1	0	4	2	1	5

The intersection of two regular languages is regular







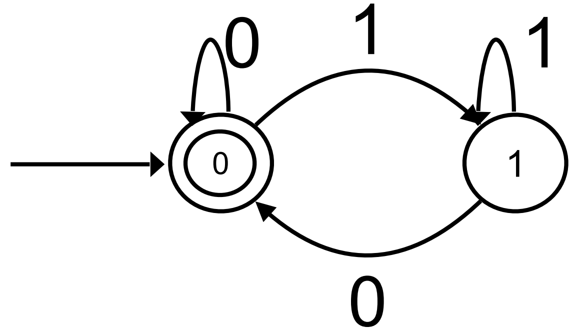
mod 3	mod 2	mod 6	0: 3	0: 2	0: 6	1: 3	1: 2	1: 6
0	0	0	0	0	0	1	1	1
1	1	1	2	0	2	0	1	3
2	0	2	1	0	4	2	1	5
0	1	3	0	0	0	1	1	1
1	0	4	2	0	2	0	1	3
2	1	5	1	0	4	2	1	5

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

- Since they are closed under intersection and complement.

Two examples

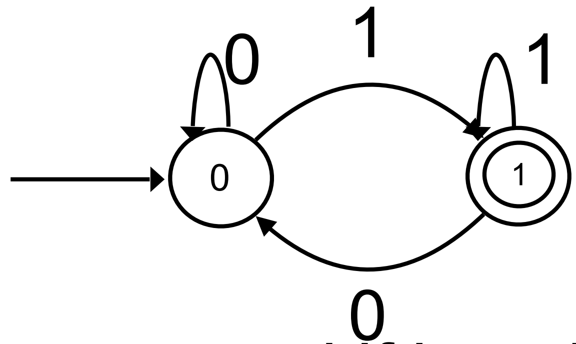
	×2 0	×2 + 1 1
0	0	1
1	0	1



Even binary numbers

Input sequence is accepted if it ends with a zero.

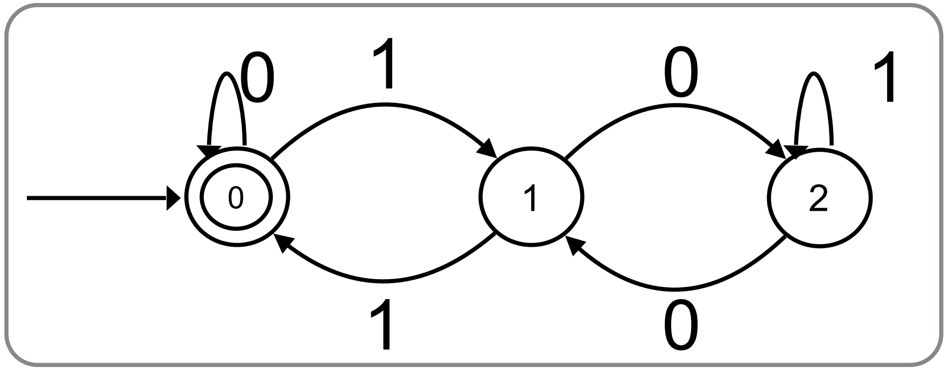
	×2 0	×2 + 1 1
0	0	1
1	0	1



Odd binary numbers

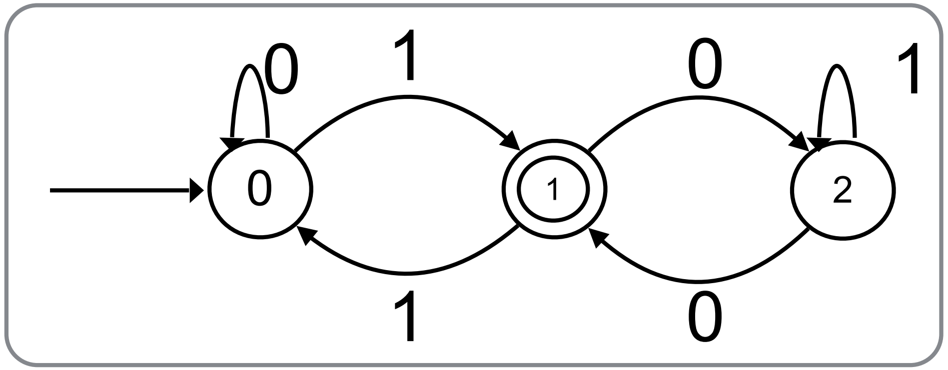
Input sequence is accepted if it ends with a one.

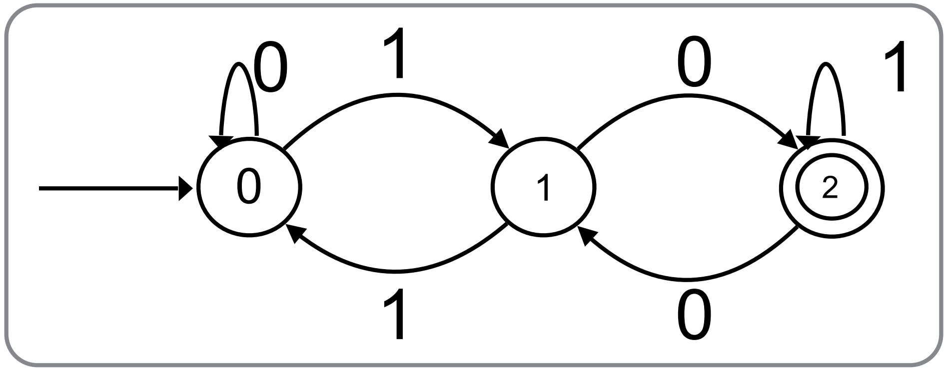
Three examples



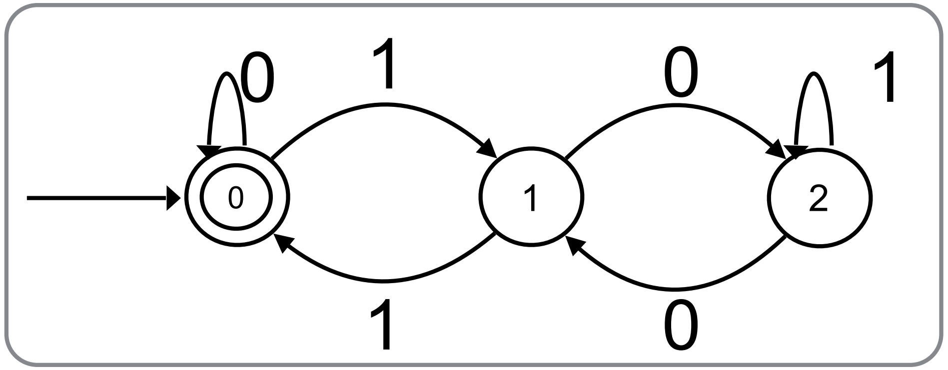
Which binary numbers are accepted?

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

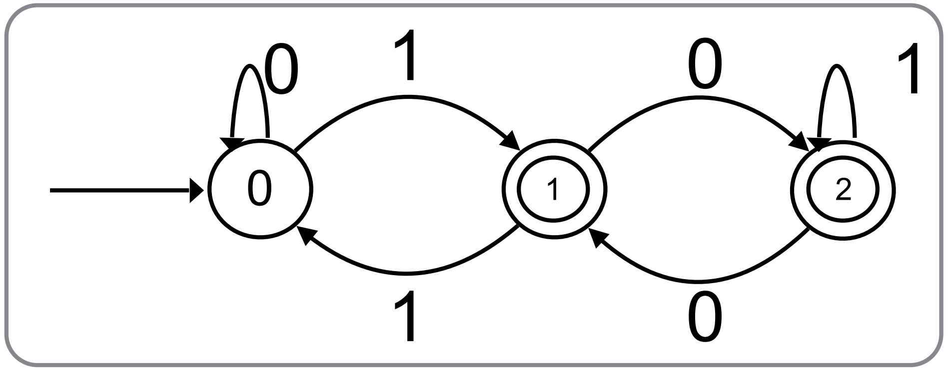




By three or not by three?

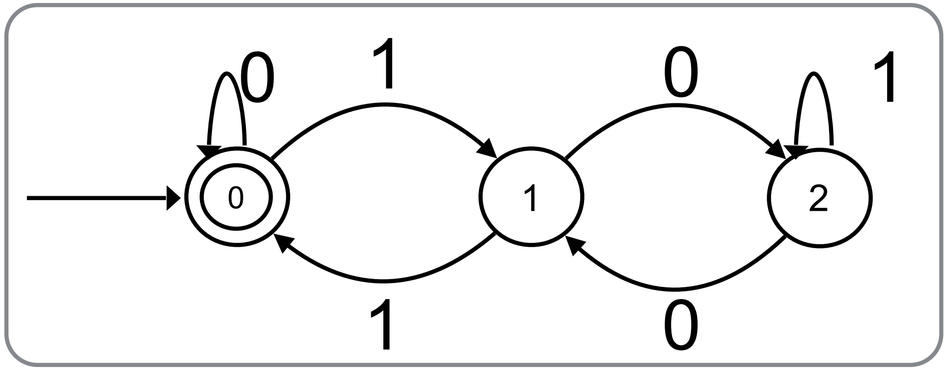


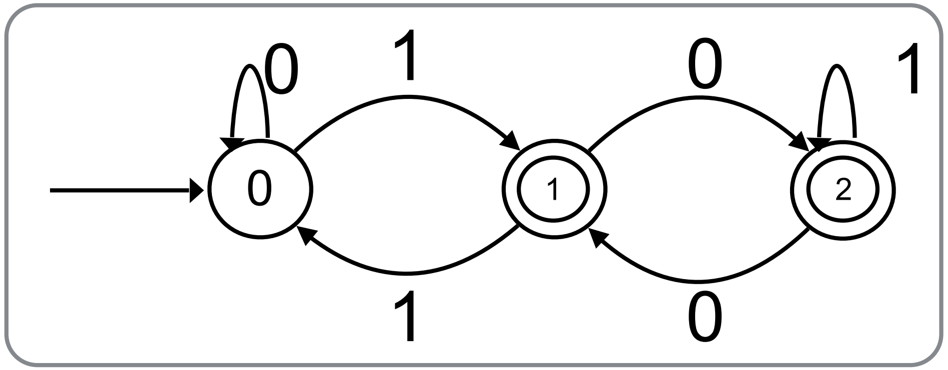
divisible by three



not
divisible by three

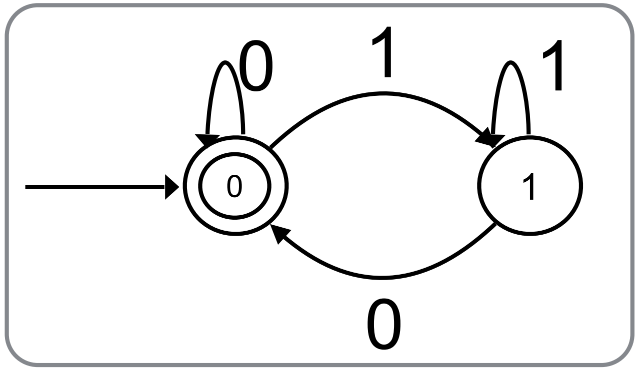
The complement of a regular language is regular

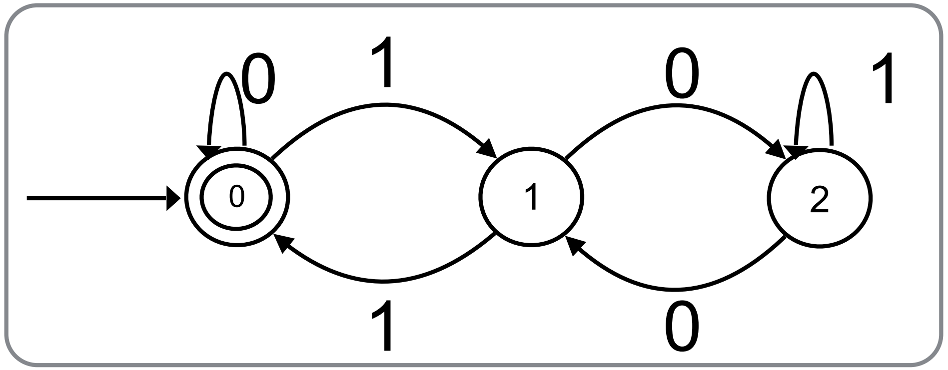




If $A \subseteq \Sigma^*$ is recognised by M then $\bar{\mathbf{A}} = \Sigma^* \setminus A$ is recognised by $\bar{\mathbf{M}}$ where $\bar{\mathbf{M}}$ and **M** are identical except that the accepting states of $\bar{\mathbf{M}}$ are the non-accepting states of **M** and vice-versa

The intersection of two regular languages is regular





divisible by 6

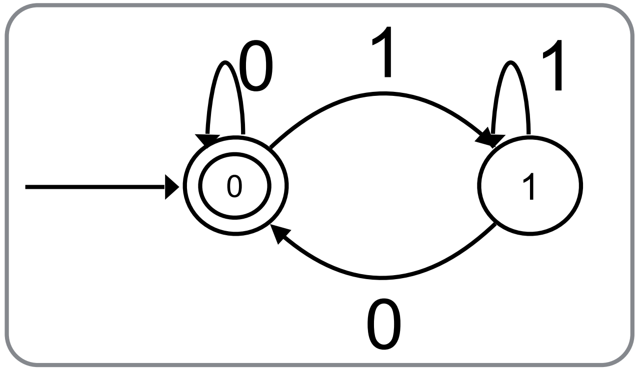
≣

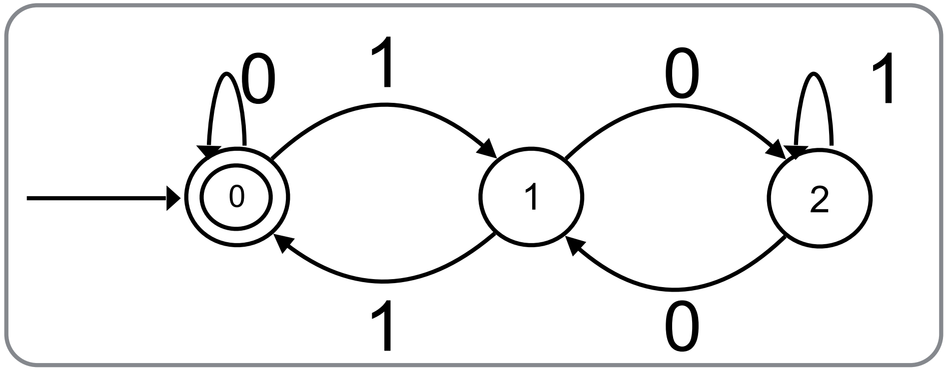
divisible by 2

and

divisible by 3

The intersection of two regular languages is regular



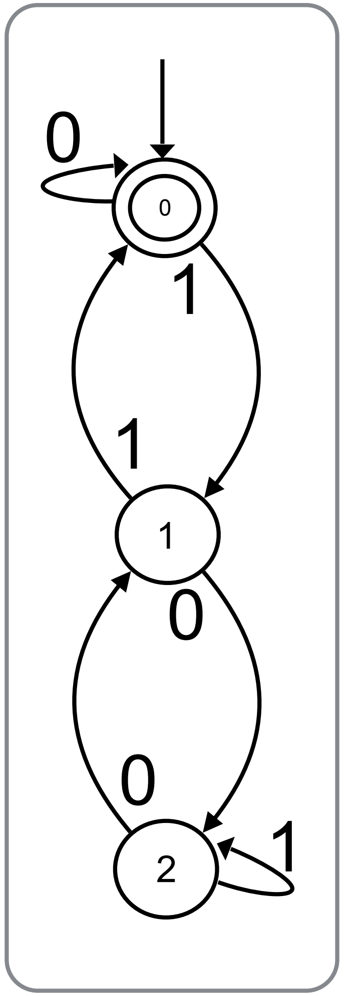


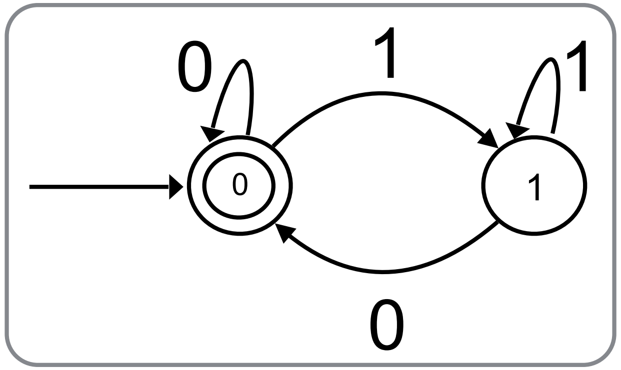
Run both machines in parallel?

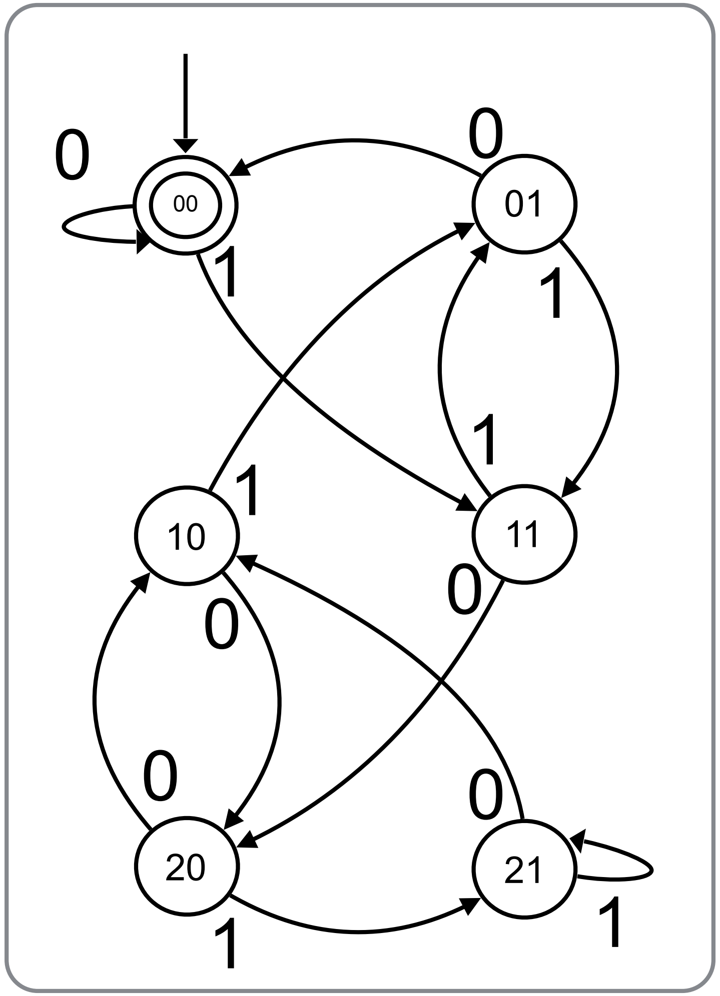
Build one machine that simulates two machines running in parallel!

Keep track of the state of each machine.

The intersection of two regular languages is regular

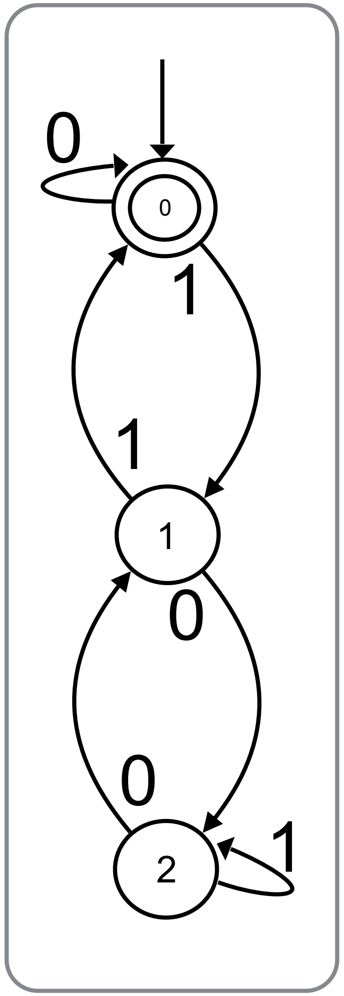


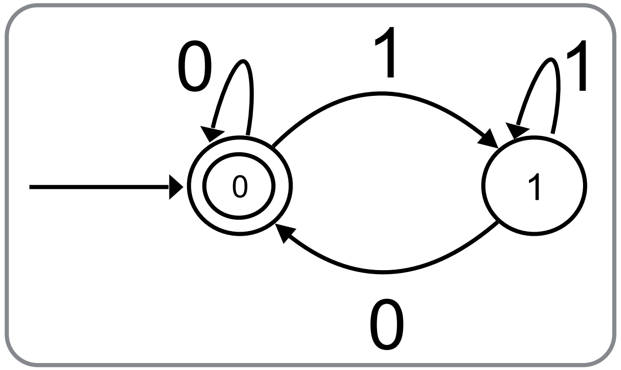


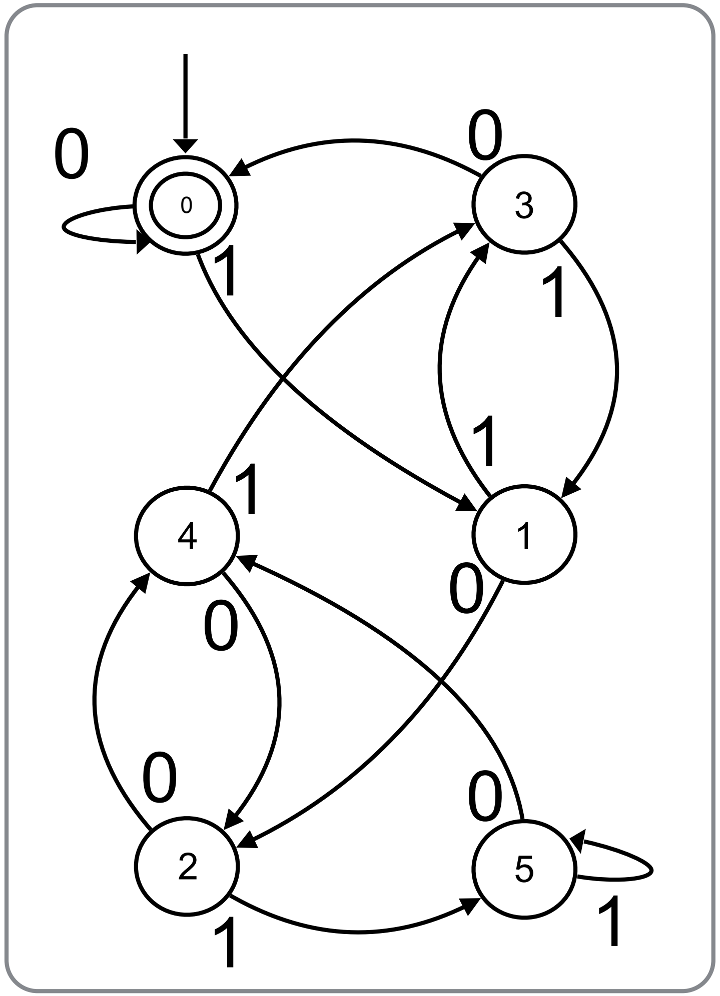


mod 3	mod 2	0		1	
		3	2	3	2
0	0	0	0	1	1
1	0	2	0	0	1
2	0	1	0	2	1
0	1	0	0	1	1
1	1	2	0	0	1
2	1	1	0	2	1

The intersection of two regular languages is regular

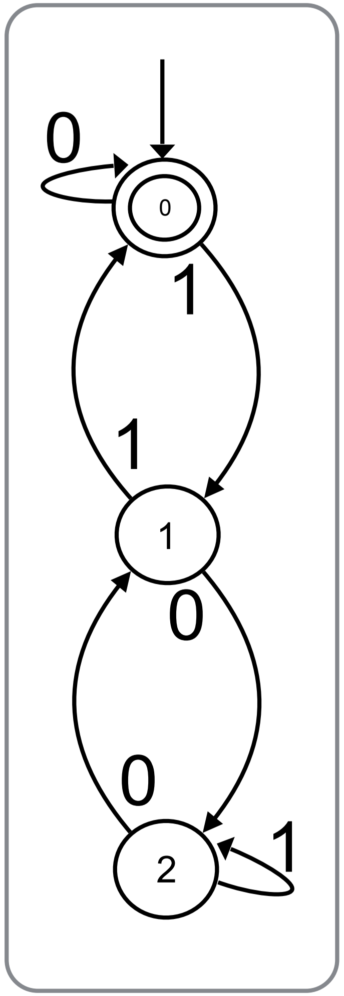




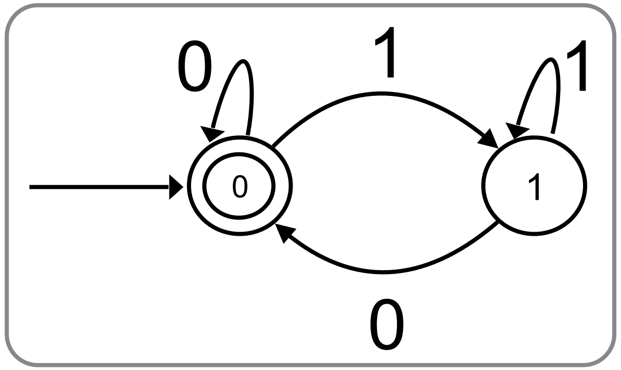


mod 3	mod 2	mod 6	0: 3	0: 2	0: 6	1: 3	1: 2	1: 6
0	0	0	0	0	0	1	1	1
1	0	4	2	0	2	0	1	3
2	0	2	1	0	4	2	1	5
0	1	3	0	0	0	1	1	1
1	1	1	2	0	2	0	1	3
2	1	5	1	0	4	2	1	5

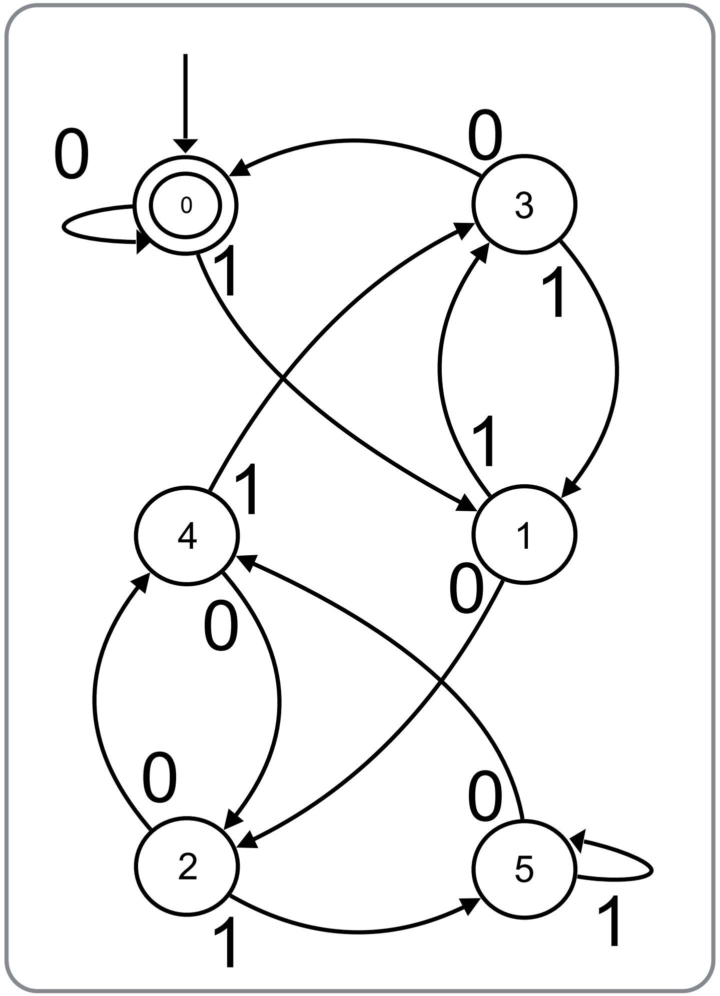
The intersection of two regular languages is regular



||



=

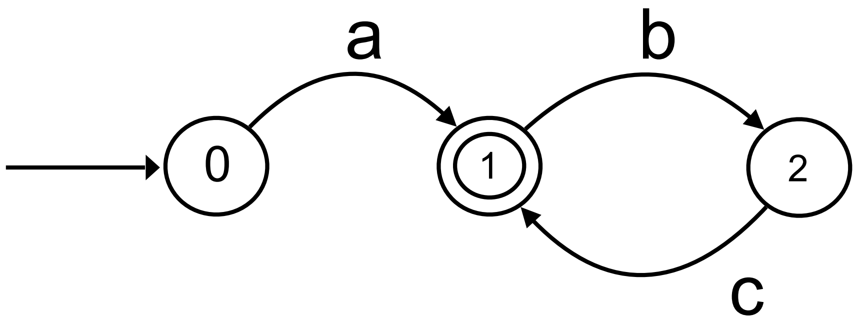


mod 3	mod 2	mod 6	0			1		
			3	2	6	3	2	6
0	0	0	0	0	0	1	1	1
1	1	1	2	0	2	0	1	3
2	0	2	1	0	4	2	1	5
0	1	3	0	0	0	1	1	1
1	0	4	2	0	2	0	1	3
2	1	5	1	0	4	2	1	5

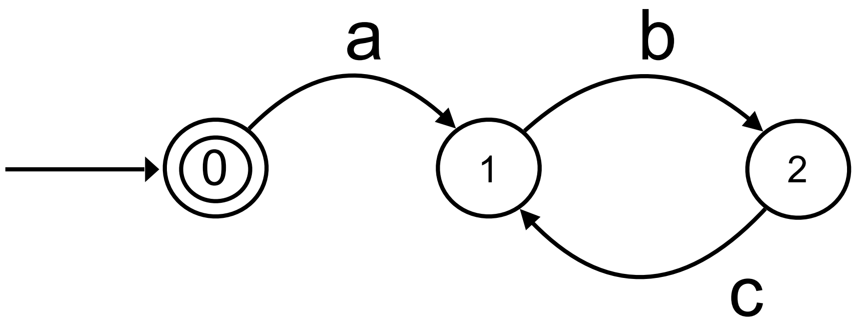
The regular languages $A \subseteq \Sigma^*$ form a Boolean Algebra

- Since they are closed under intersection and complement.

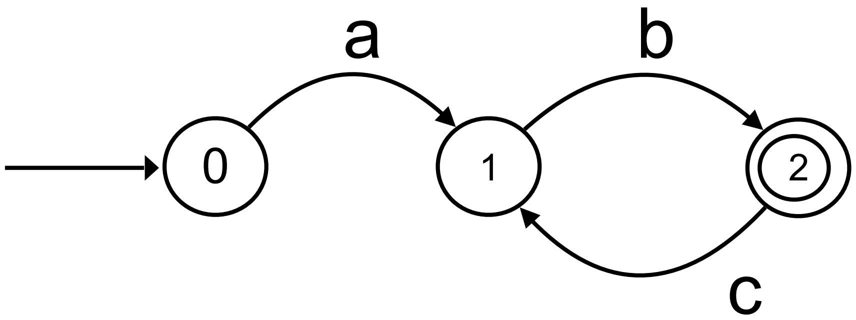
Is there a regular expression for every FSM?



$L_1 =$

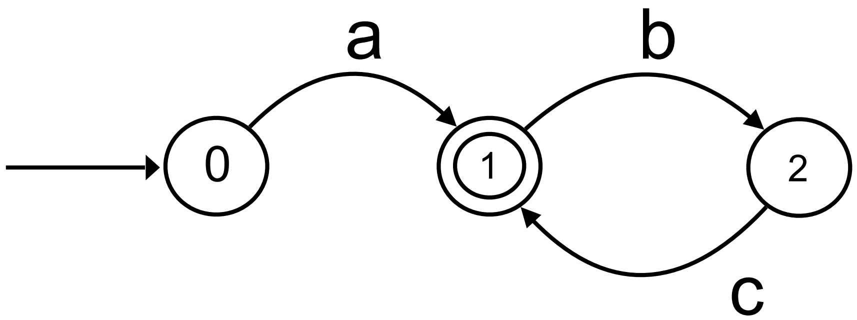


$L_0 =$

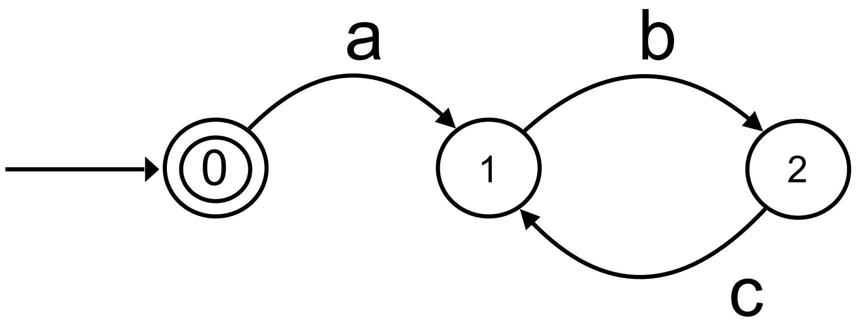


$L_2 =$

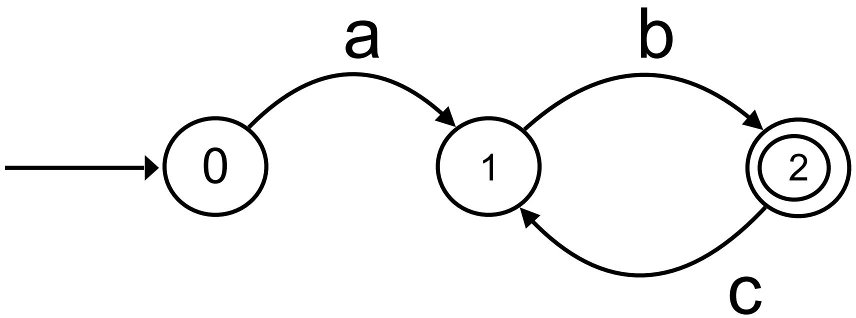
Is there a regular expression for every FSM?



$L_1 =$

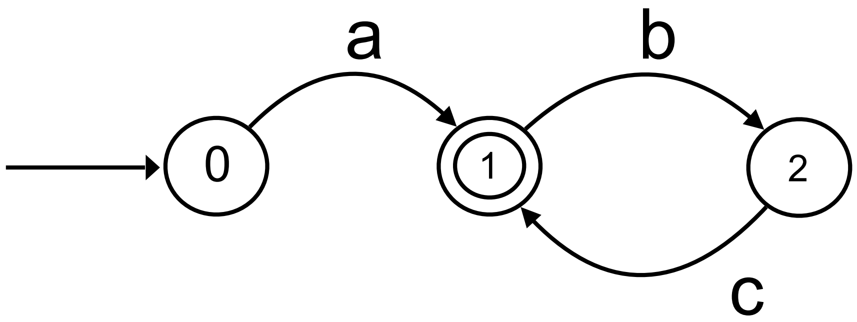


$L_0 = \varepsilon$

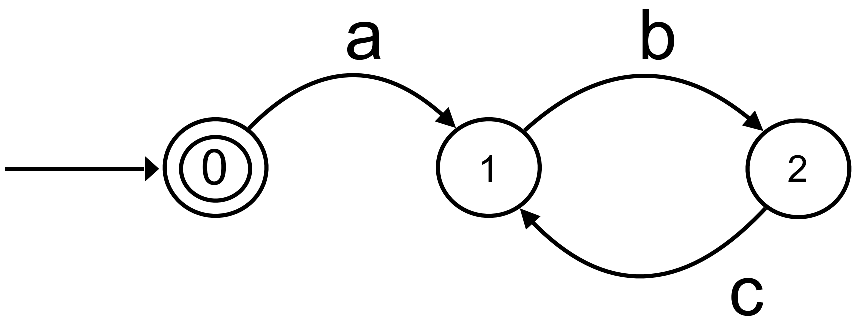


$L_2 =$

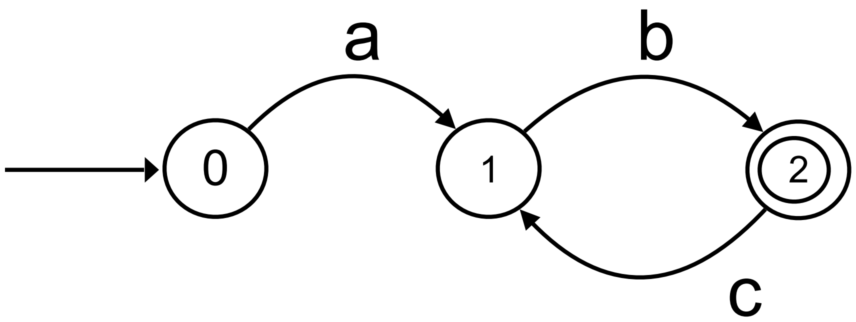
Is there a regular expression for every FSM?



$$L_1 = L_0\, a \mid L_2\, c$$

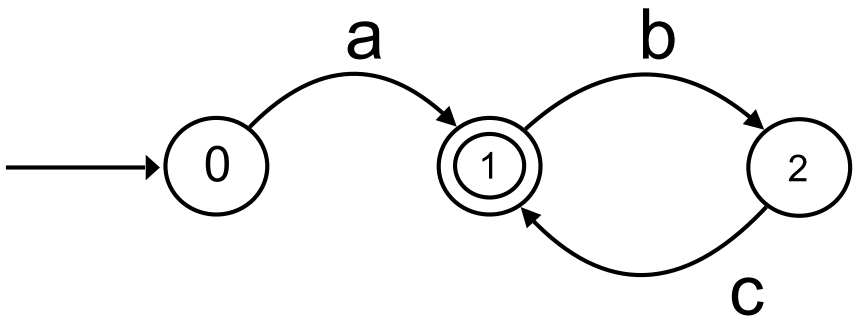


$$L_0 = \varepsilon$$

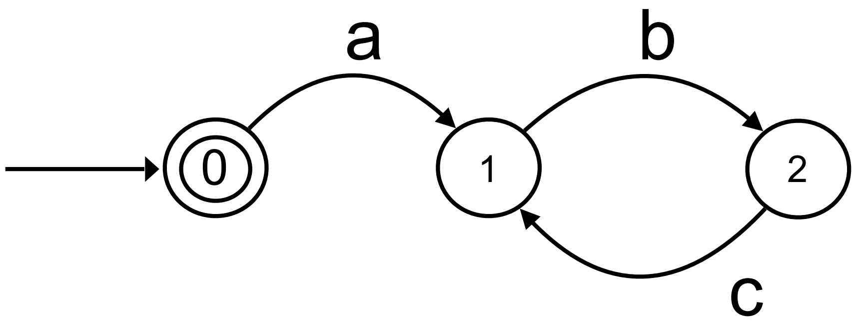


$$L_2 = L_1\, b$$

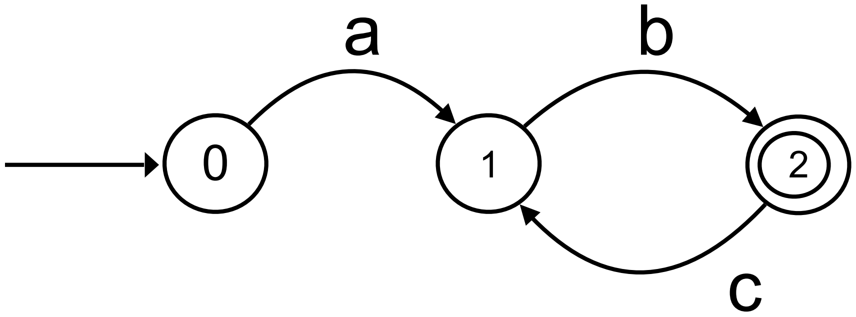
Is there a regular expression for every FSM?



$$L_1 = L_0\, a \mid L_2\, c$$
$$= a \mid L_1\, bc$$

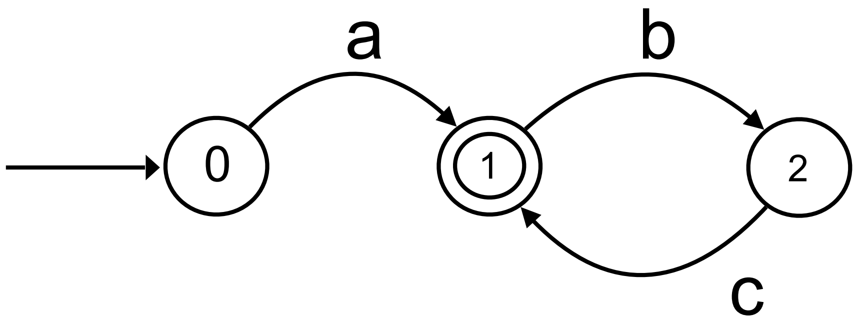


$$L_0 = \varepsilon$$

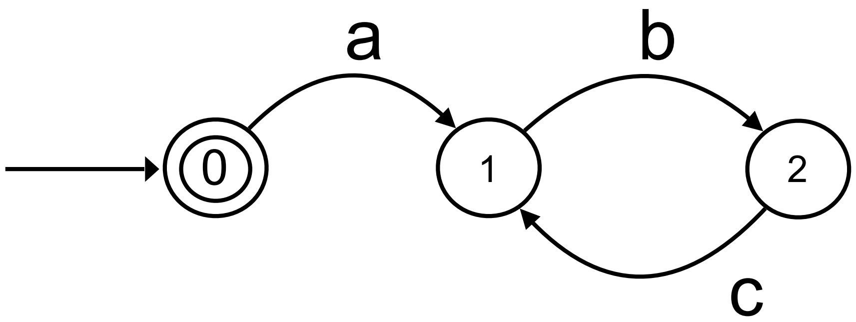


$$L_2 = L_1\, b$$

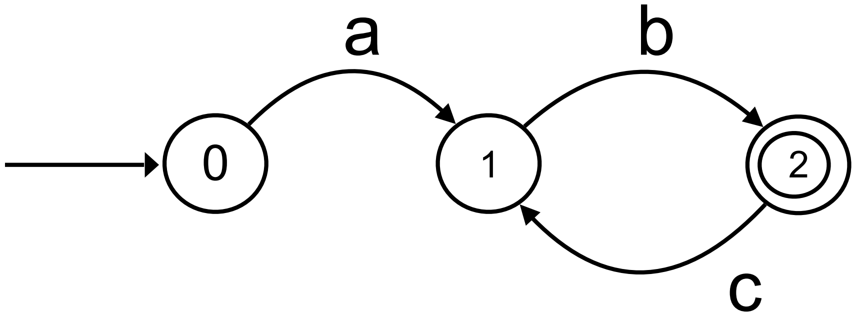
Is there a regular expression for every FSM?



$$L_1 = L_0\, a \mid L_2\, c$$
$$= a \mid L_1\, bc$$
$$L_1 = a \mid (bc)^*$$



$$L_0 = \varepsilon$$



$$L_2 = L_1\, b$$

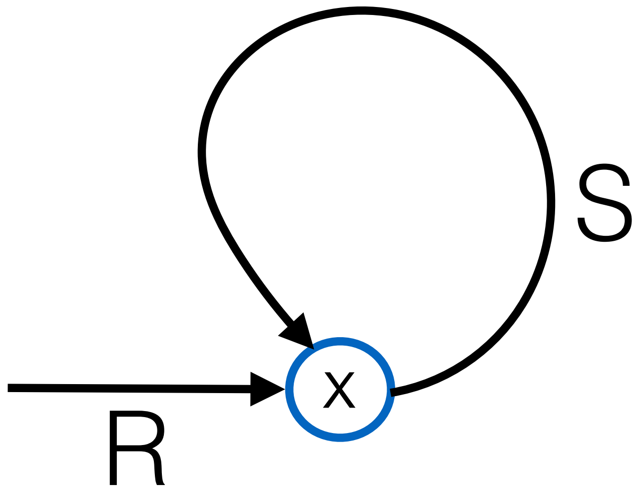
Arden's Lemma

If R and S are regular expressions then the equation

$X = R \mid X\ S$

has a solution $X = R\ S^*$

If $\varepsilon \notin L(S)$ then this solution is unique.

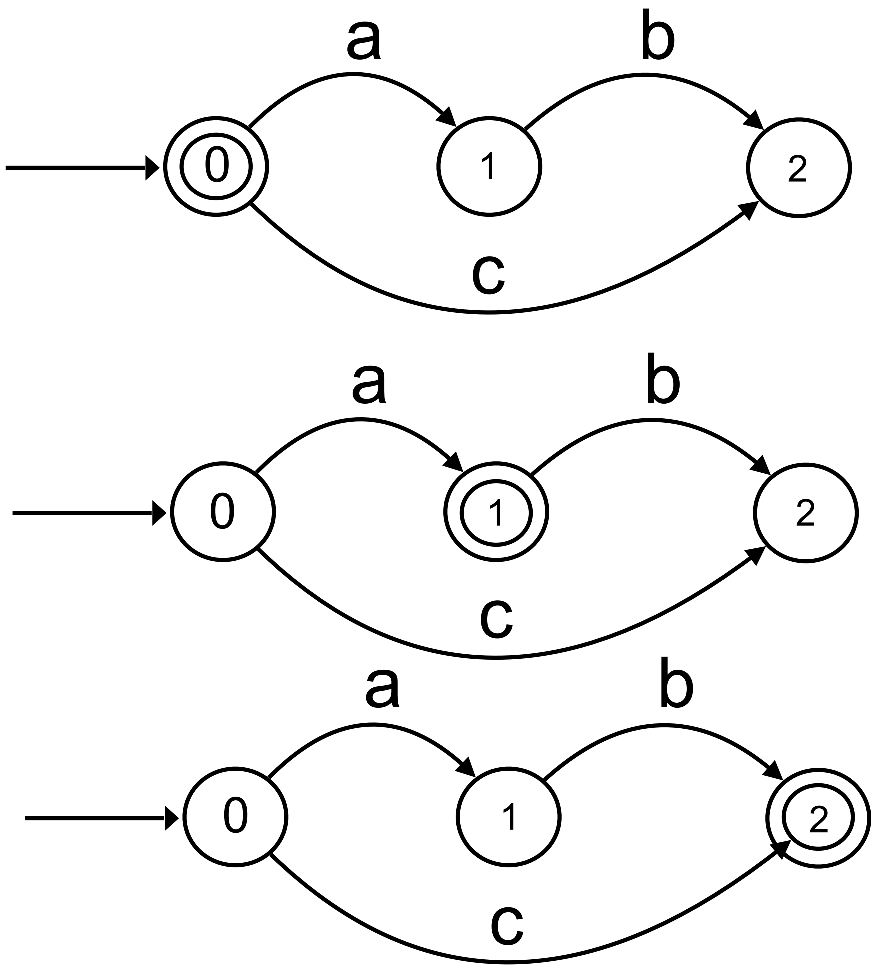


Is there a regular expression for every FSM?

L_0

L_1

L_2



Let L_i be the language accepted if **i** is the accepting state

$L_0 = \varepsilon$

$L_1 = L_0\, a$

$L_2 = L_1\, b \mid L_0\, c$

$L_2 = L_0\, a\, b \mid \varepsilon\, c$

$L_2 = \varepsilon\, a\, b \mid \varepsilon\, c$

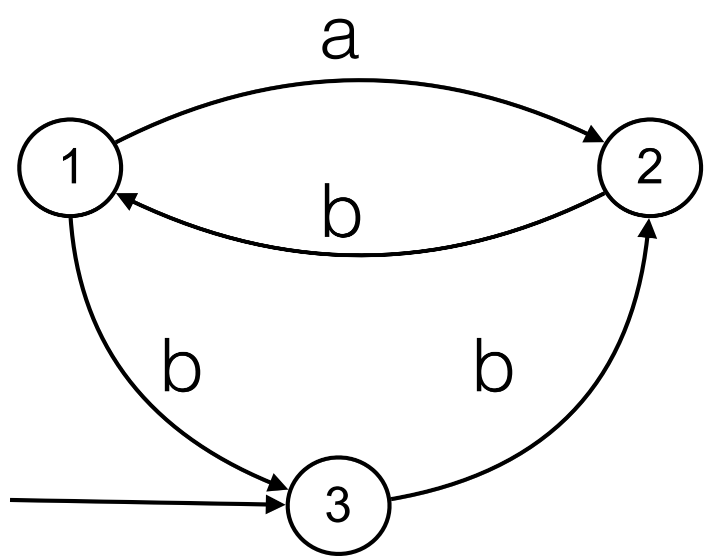
$L_2 = a\, b \mid c$

Is there a regular expression for every FSM?

$$L_1 = L_2\, b$$

$$L_2 = L_3\, b \mid L_1\, a$$

$$L_3 = \varepsilon \mid L_1\, b$$
$$= \varepsilon \mid L_2\, b\, b$$

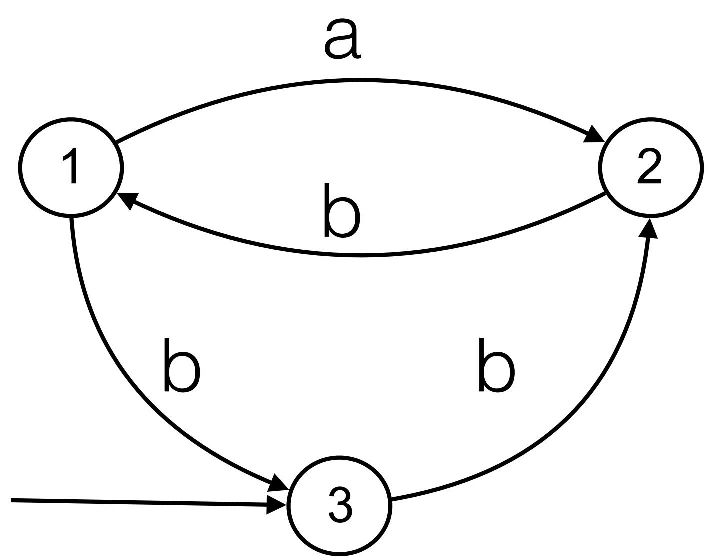


$$L_2 = (\varepsilon \mid L_2\, b\, b)\, b \mid L_2\, b\, a$$
$$= b \mid L_2\, b\, b\, b \mid L_2\, b\, a$$
$$= b \mid L_2\, (b\, b\, b \mid b\, a)$$

Is there a regular expression for every FSM?

$$L_2 = b \mid L_2\,(b\,b\,b \mid b\,a)$$

$$L_2 = b\,(b\,b\,b \mid b\,a)^*$$



$$L_1 = L_2\,b = b\,(b\,b\,b \mid b\,a)^*\,b$$

$$L_3 = \varepsilon \mid L_2\,b\,b = \varepsilon \mid b\,(b\,b\,b \mid b\,a)^*\,bb$$

Arden's Lemma

If R and S are regular expressions then the equation

$$X = R \mid X\,S$$

has a solution $X = R\,S^*$

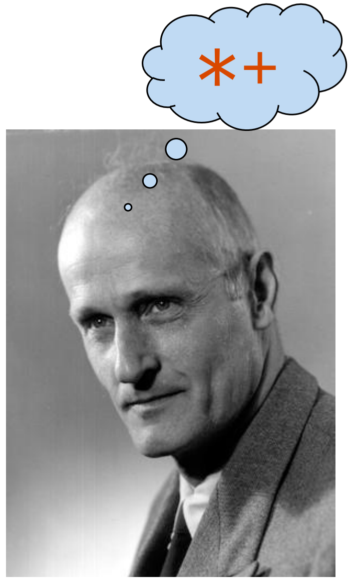
If $\varepsilon \notin L(S)$ then this solution is unique.

$L_2 = b \mid L_2\,(b\,b\,b \mid b\,a)$

$L_2 = b\,(b\,b\,b \mid b\,a)^*$

regular expressions

Kleene *, +



Stephen Cole Kleene

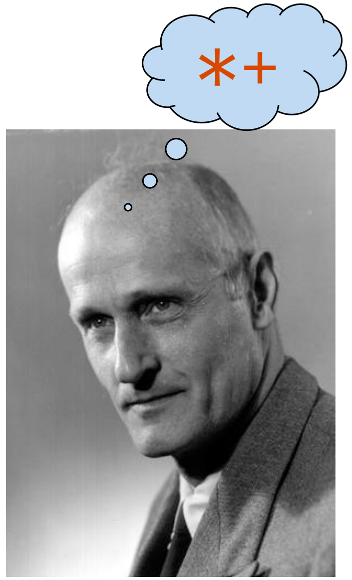
1909-1994

- any character is a regexp
  - matches itself
- if R and S are regexps, so is RS
  - matches
    a match for R followed by a match for S
- if R and S are regexps, so is R|S
  - matches
    any match for R or S (or both)
- if R is a regexp, 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 matches the empty string

regular expressions denote
regular sets

- any character a is a regexp
  - {<a>}
- if R and S are regexs, so is RS
  - { r s | r ∈ R and s ∈ S }
- if R and S are regexps, so is R|S
  - R ∪ S
- if R is a regexp, so is R*
  - { r^n | n ∈ N and r ∈ R
- 0 0 | S = S = S | 0
  - ∅ empty set
- 1 1 S = S = S 1
  - {<>} singleton empty sequence:

Kleene *, +



Stephen Cole Kleene

1909-1994

https://en.wikipedia.org/wiki/Kleene_algebra