Digital World (2018)

Week 9, S1: State Machines

Chris Poskitt

SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

WHAT WAS IT?

"lunch today was exquisite!"

it

"the weather was atrocious"

it

"I just watched Stephen King's IT"

it

State machines

- state machines model systems for which the output depends on their history of inputs

- idea: find a set of states that capture the essential properties of the history of inputs

  => *use them to determine the current output…*

  => *…as well as the next state*

- state machines perform a transduction from a stream of inputs to a stream of outputs

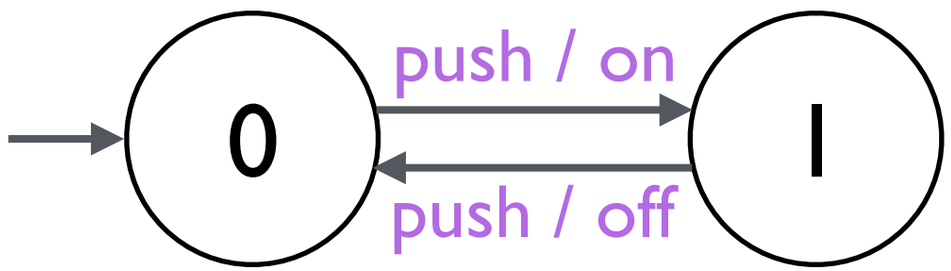
Abusing your torchlight

push push push push push push push push

transduce

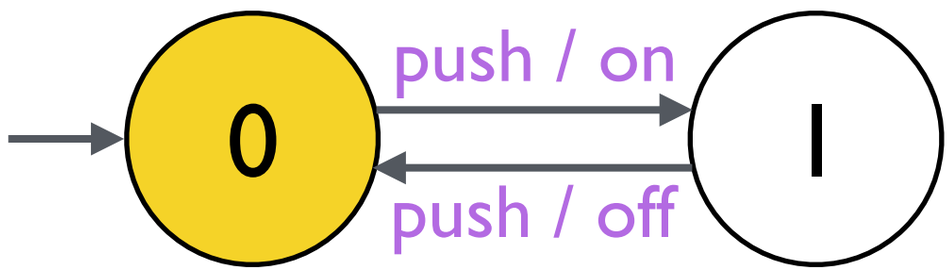
on off on off on off on

Abusing your torchlight



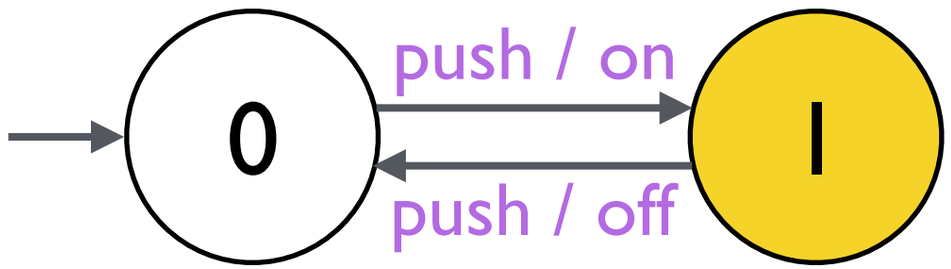
input:	*push*	*push*	*push*	*push*	*push*	*push*	*push*
output:							

Abusing your torchlight



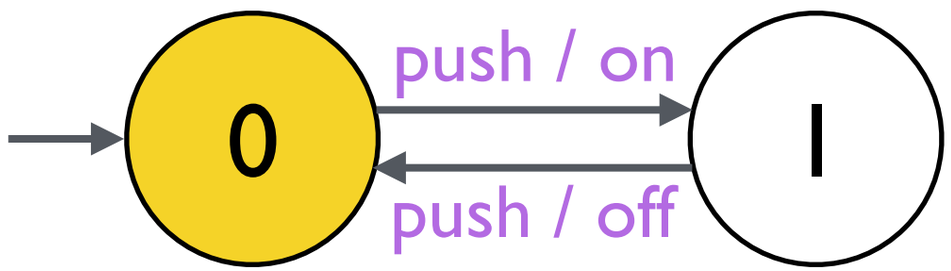
input:	*push*	*push*	*push*	*push*	*push*	*push*	*push*
output:							

Abusing your torchlight



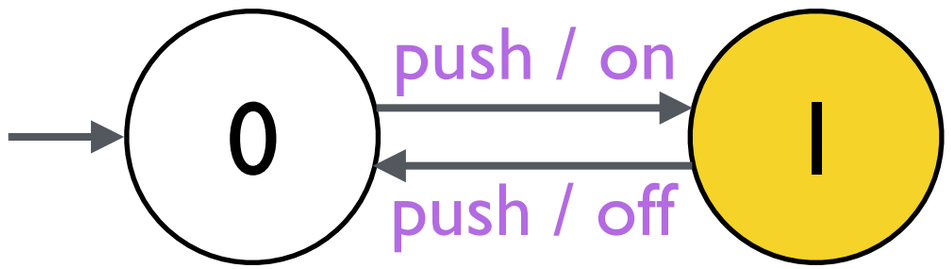
input:	*push*	*push*	*push*	*push*	*push*	*push*	*push*
output:	*on*						

Abusing your torchlight



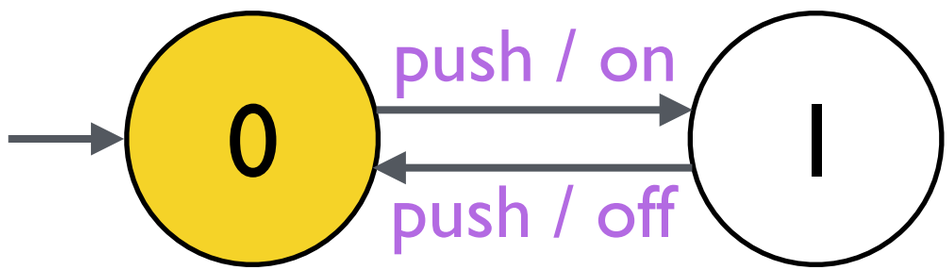
input:	*push*	*push*	*push*	*push*	*push*	*push*	*push*
output:	*on*	*off*					

Abusing your torchlight



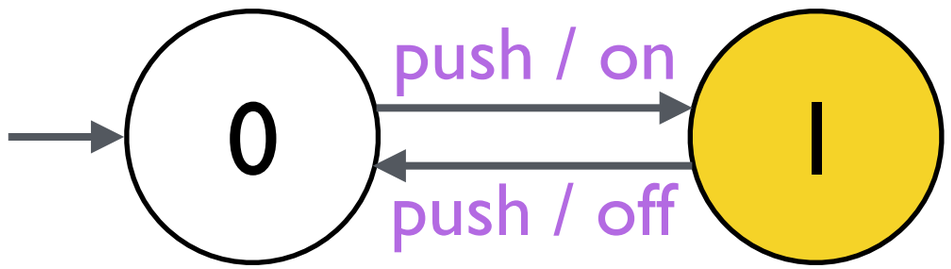
input:	*push*	*push*	*push*	*push*	*push*	*push*	*push*
output:	*on*	*off*	*on*				

Abusing your torchlight



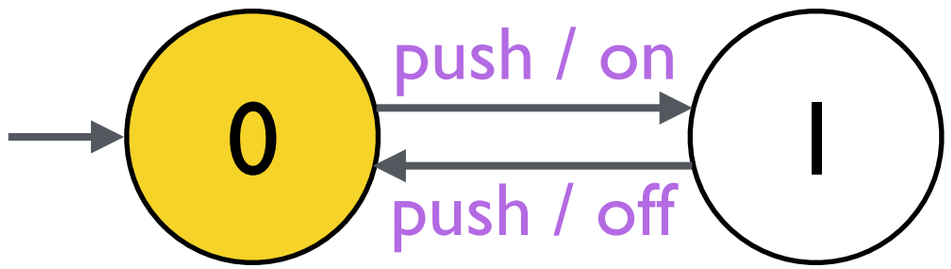
input:	*push*	*push*	*push*	*push*	*push*	*push*	*push*
output:	*on*	*off*	*on*	*off*			

Abusing your torchlight



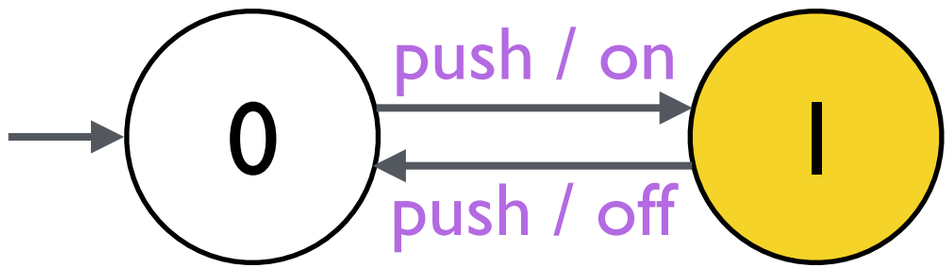
input:	*push*	*push*	*push*	*push*	*push*	*push*	*push*
output:	*on*	*off*	*on*	*off*	*on*		

Abusing your torchlight



input:	*push*	*push*	*push*	*push*	*push*	*push*	*push*
output:	*on*	*off*	*on*	*off*	*on*	*off*	

Abusing your torchlight



input:	*push*	*push*	*push*	*push*	*push*	*push*	*push*
output:	*on*	*off*	*on*	*off*	*on*	*off*	*on*

Language recogniser: *abcabcabcabc…*

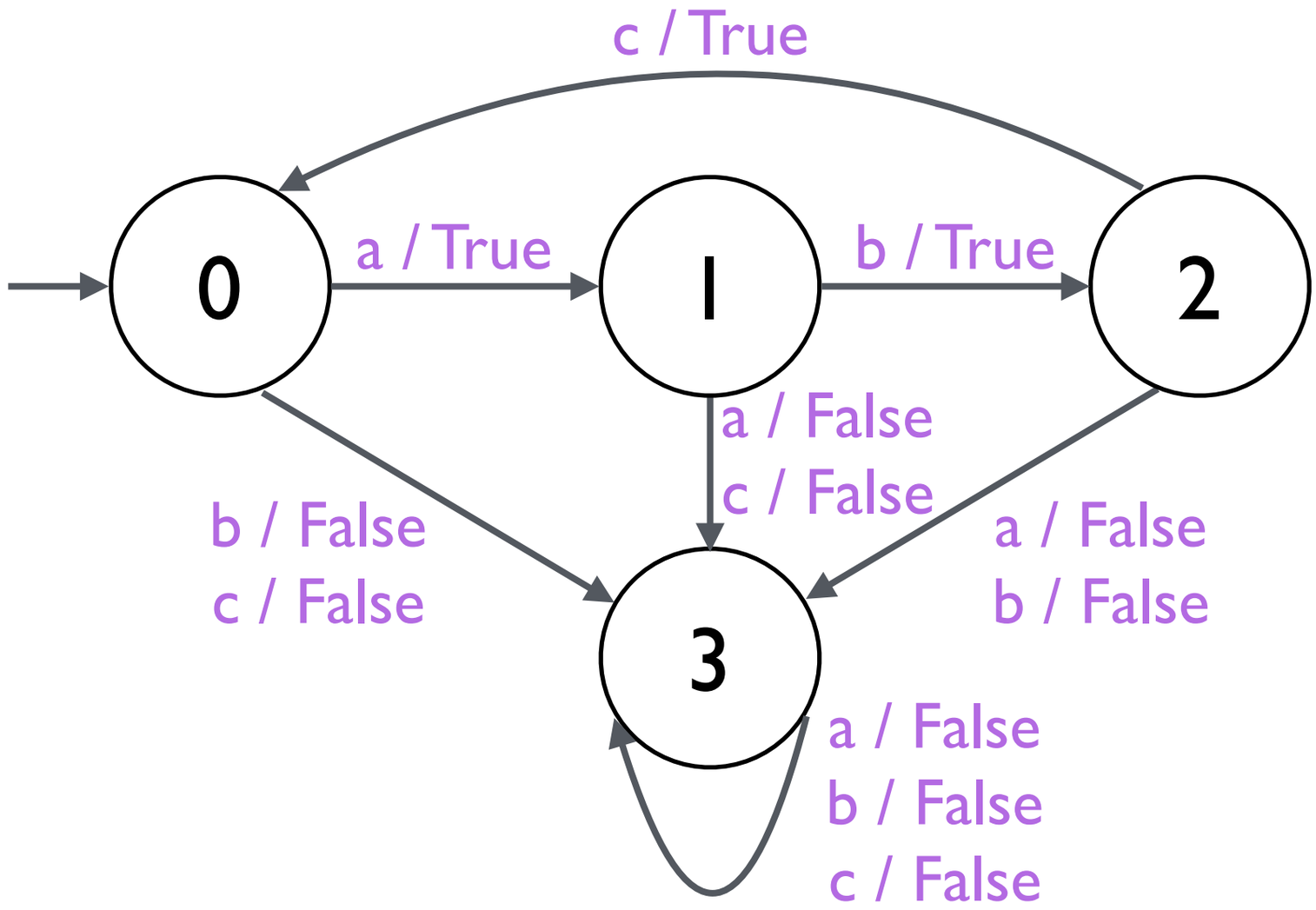
- abcabcabcabc ✓
- aabcabcabcabc X
- abcabcaaaaaaa X
- abccccccccc X

a b c c c

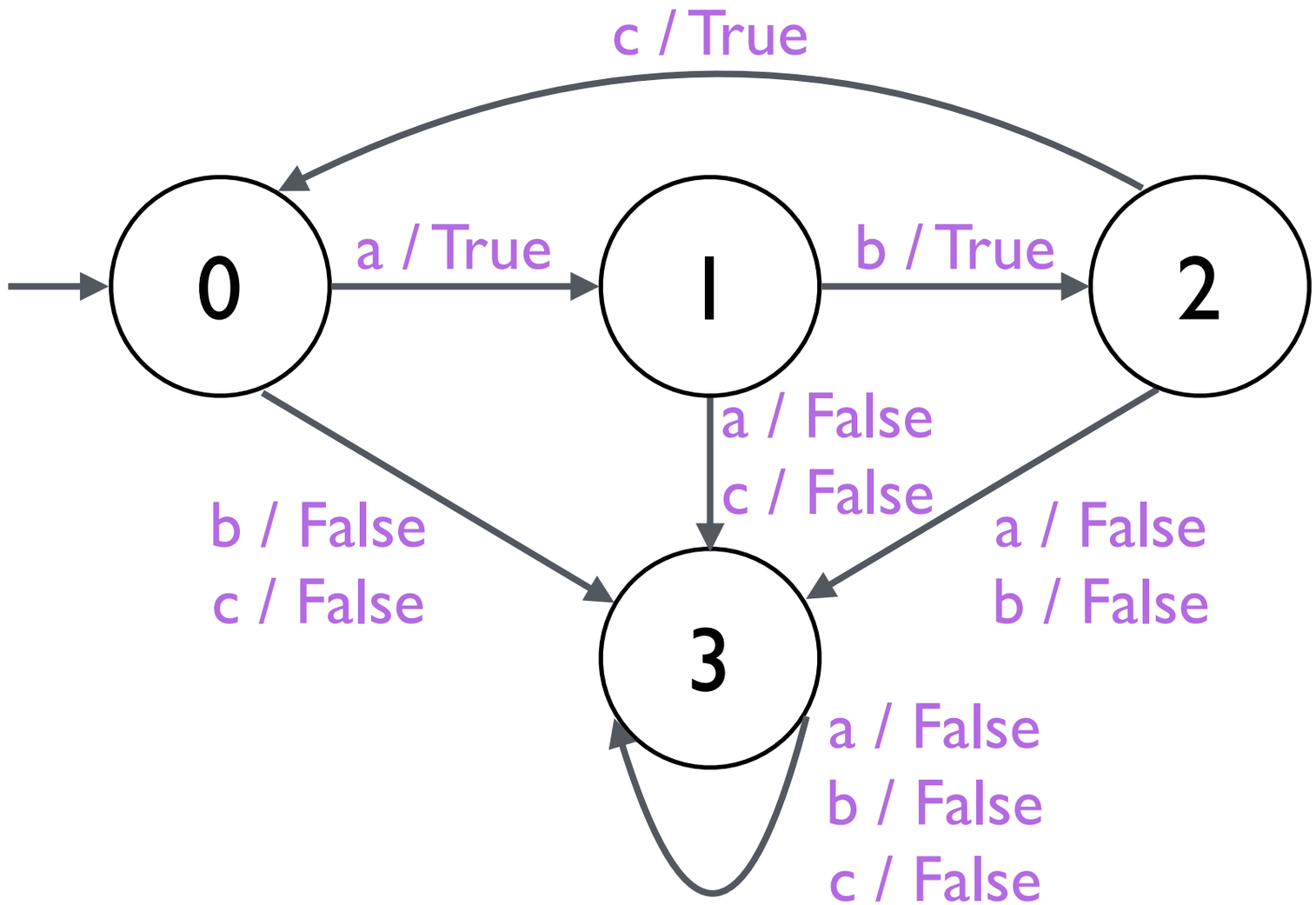
transduce

True True True False False

Language recogniser: *abcabcabcabc…*

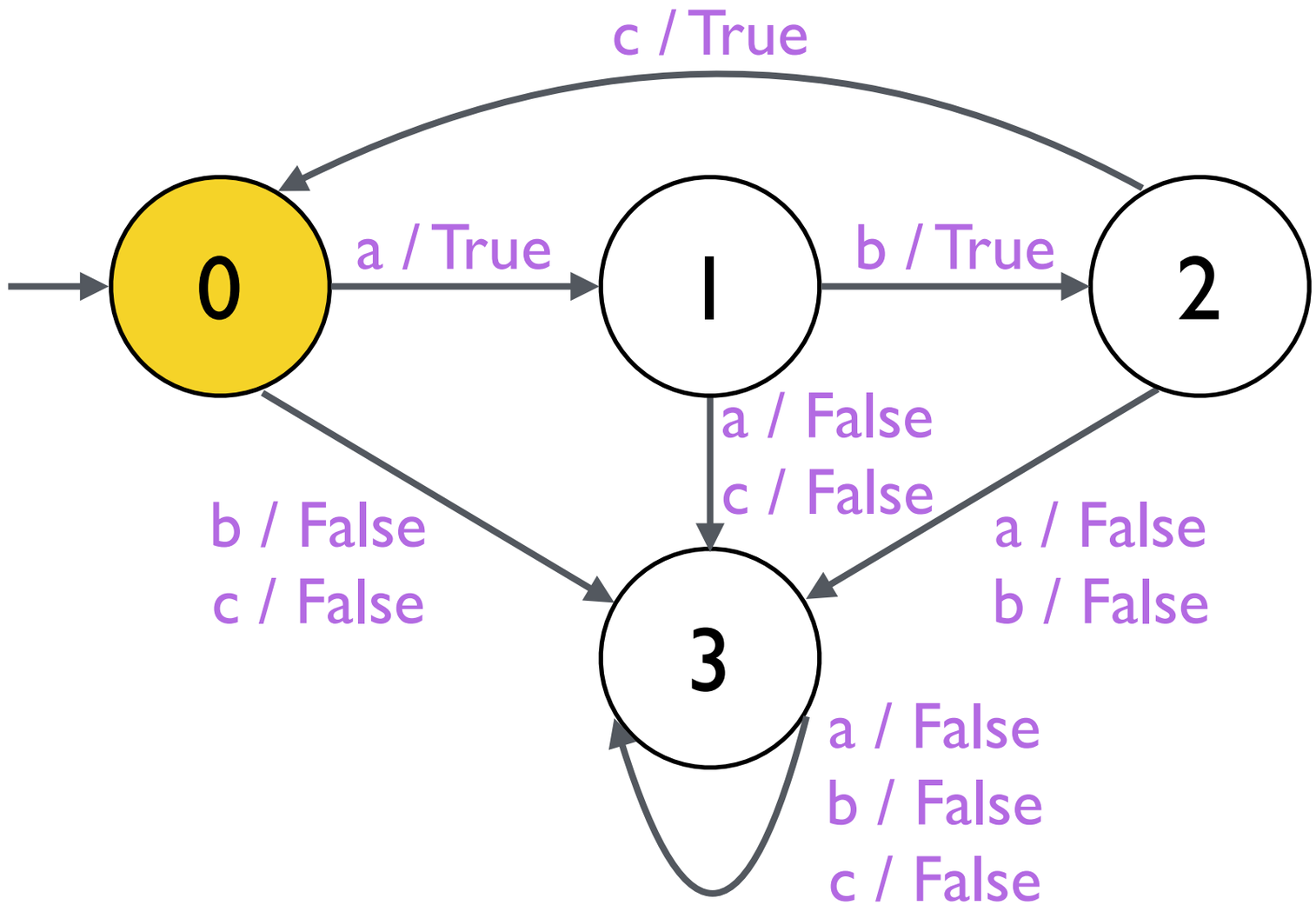


Language recogniser: *abcabcabcabc…*



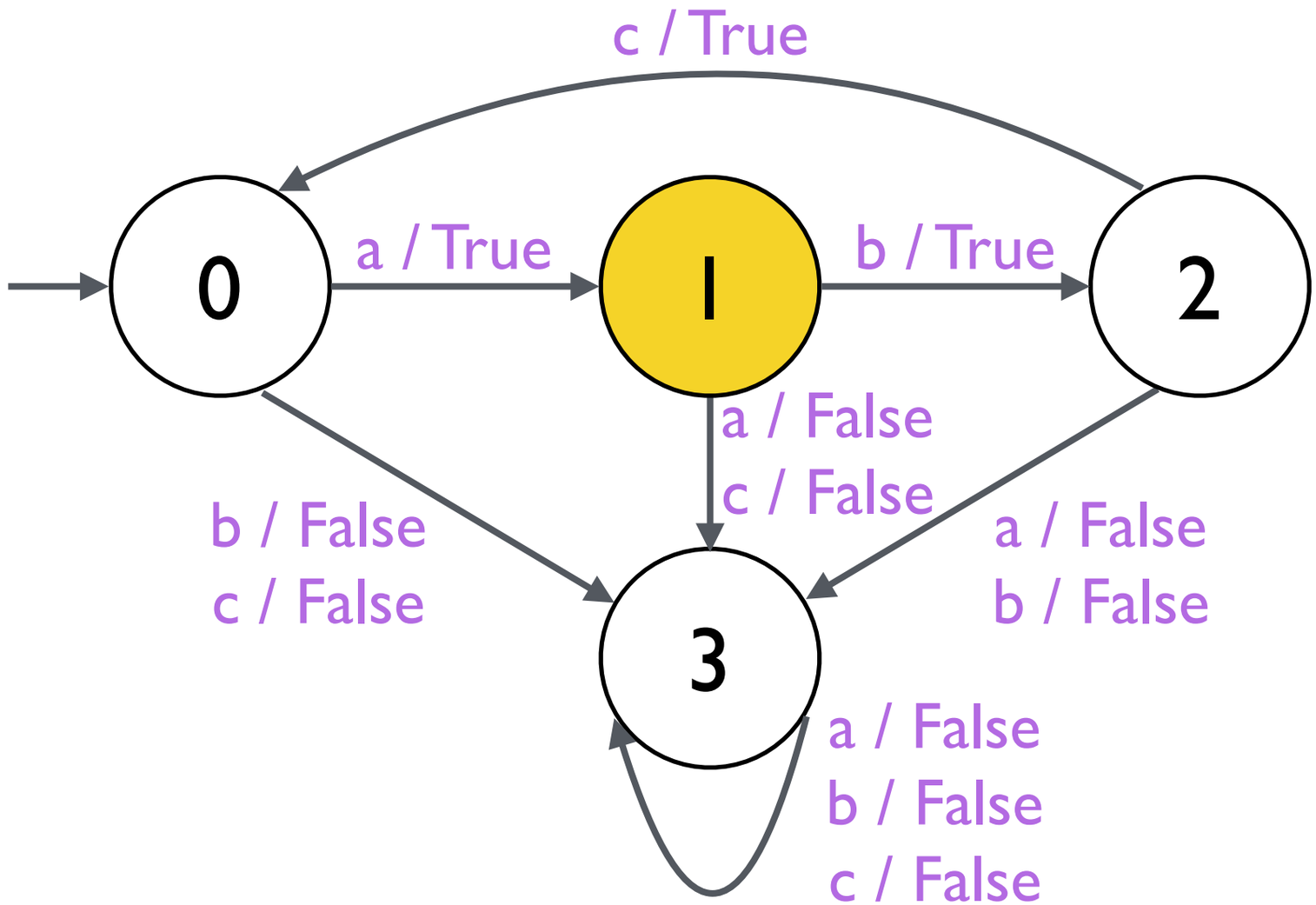
input:	*a*	*b*	*c*	*c*	*c*
output:					

Language recogniser: *abcabcabcabc…*



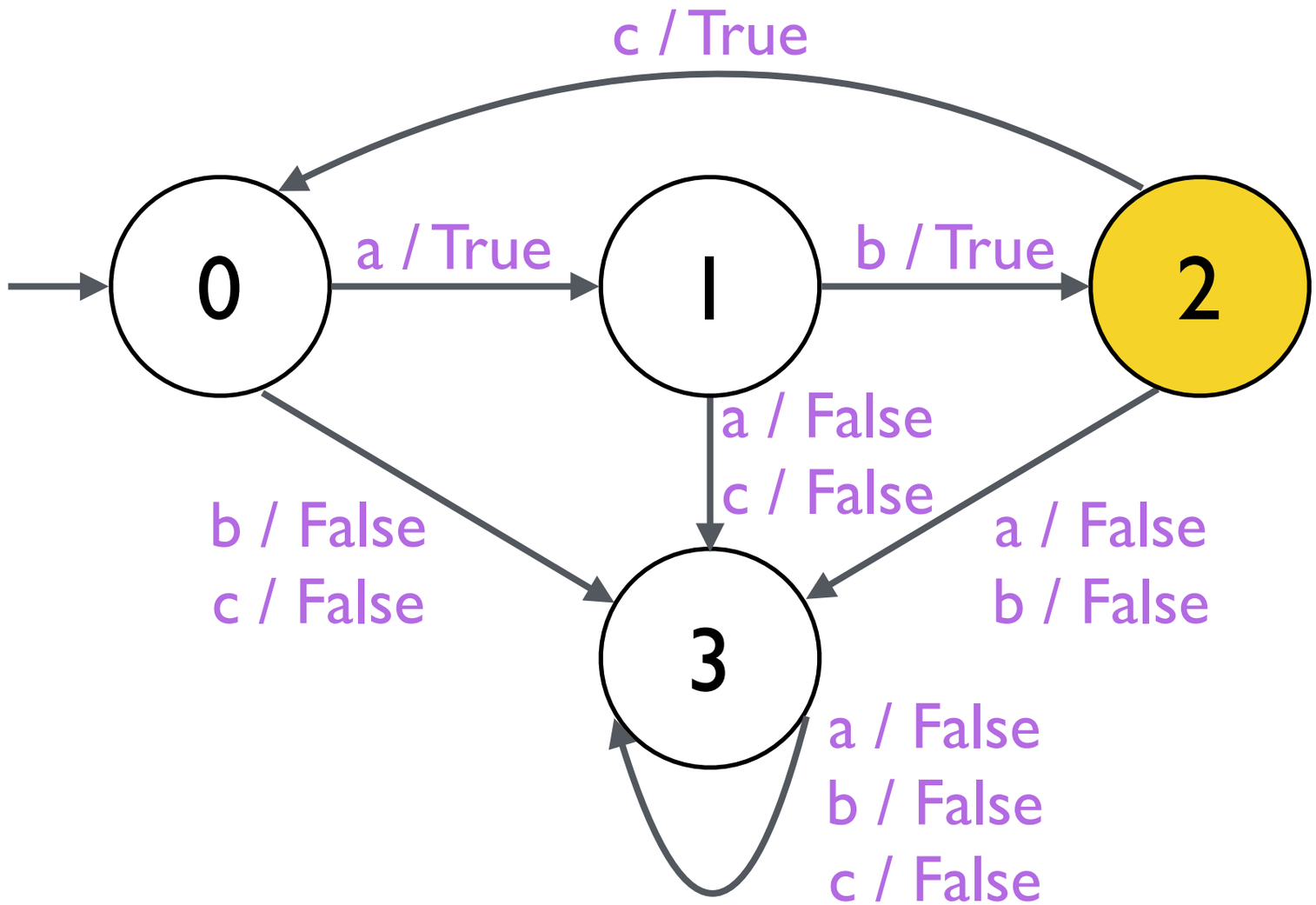
input:	*a*	*b*	*c*	*c*	*c*
output:					

Language recogniser: *abcabcabcabc…*



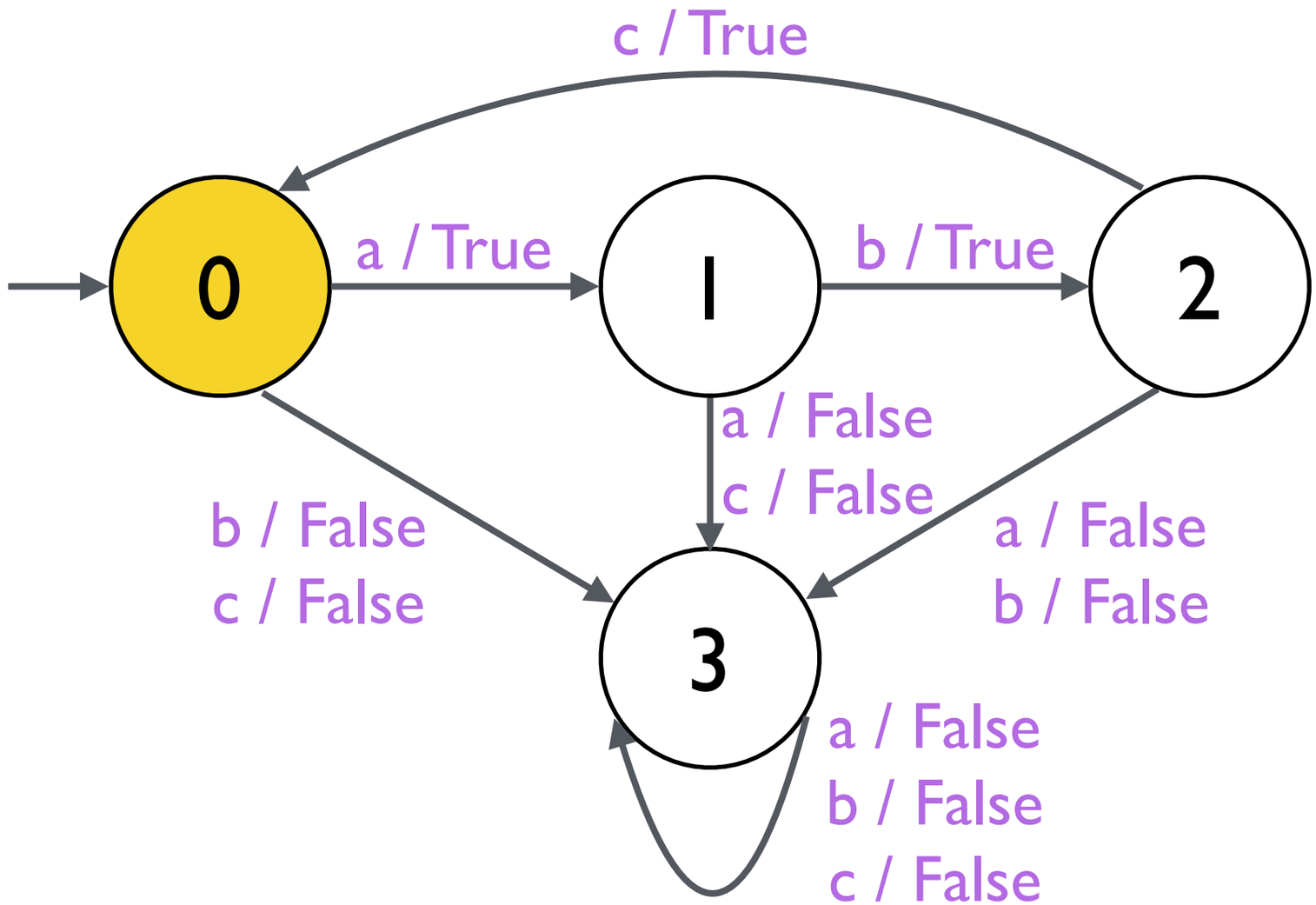
input:	*a*	*b*	*c*	*c*	*c*
output:	*True*				

Language recogniser: *abcabcabcabc…*



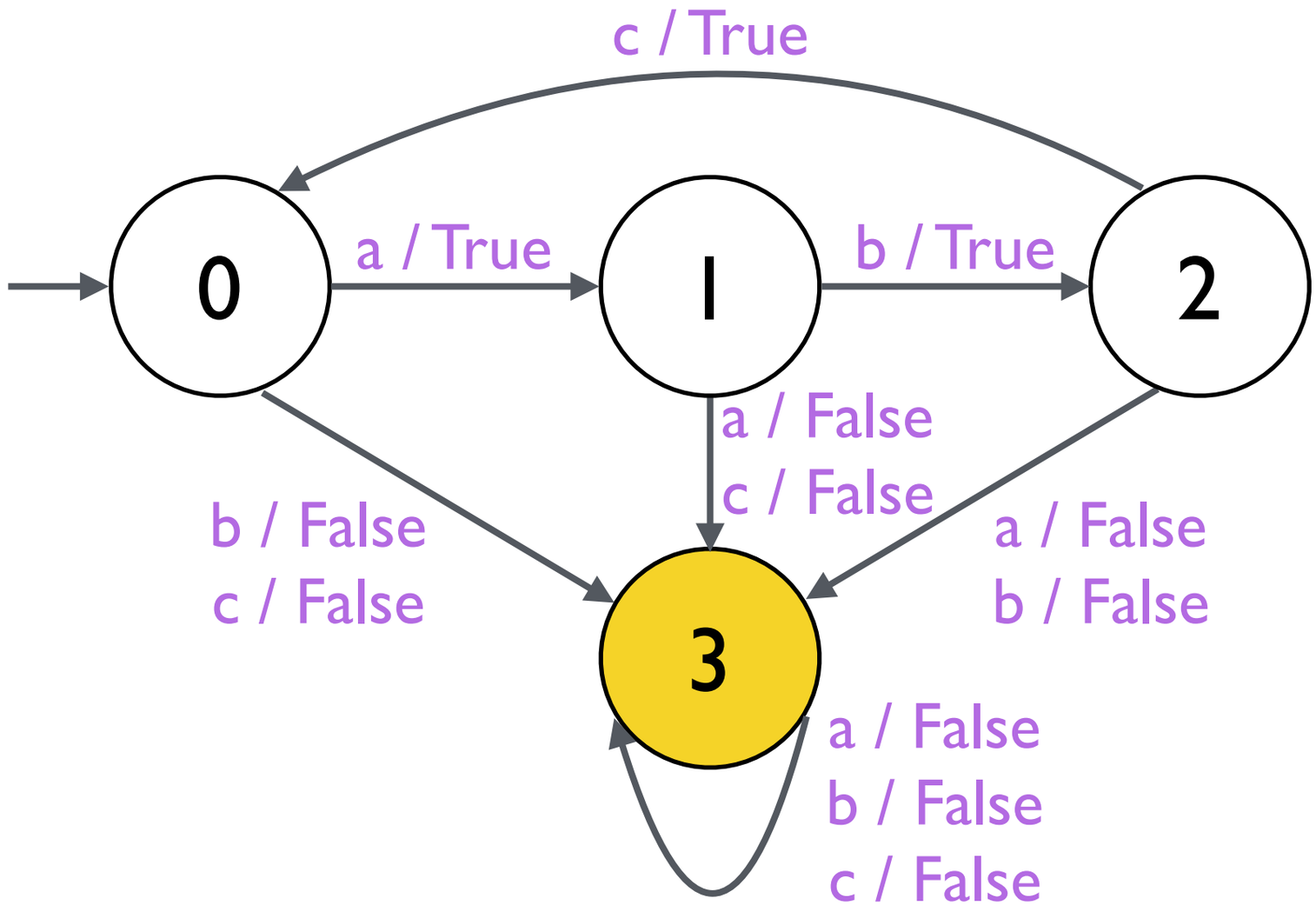
input:	*a*	*b*	*c*	*c*	*c*
output:	*True*	*True*			

Language recogniser: *abcabcabcabc…*



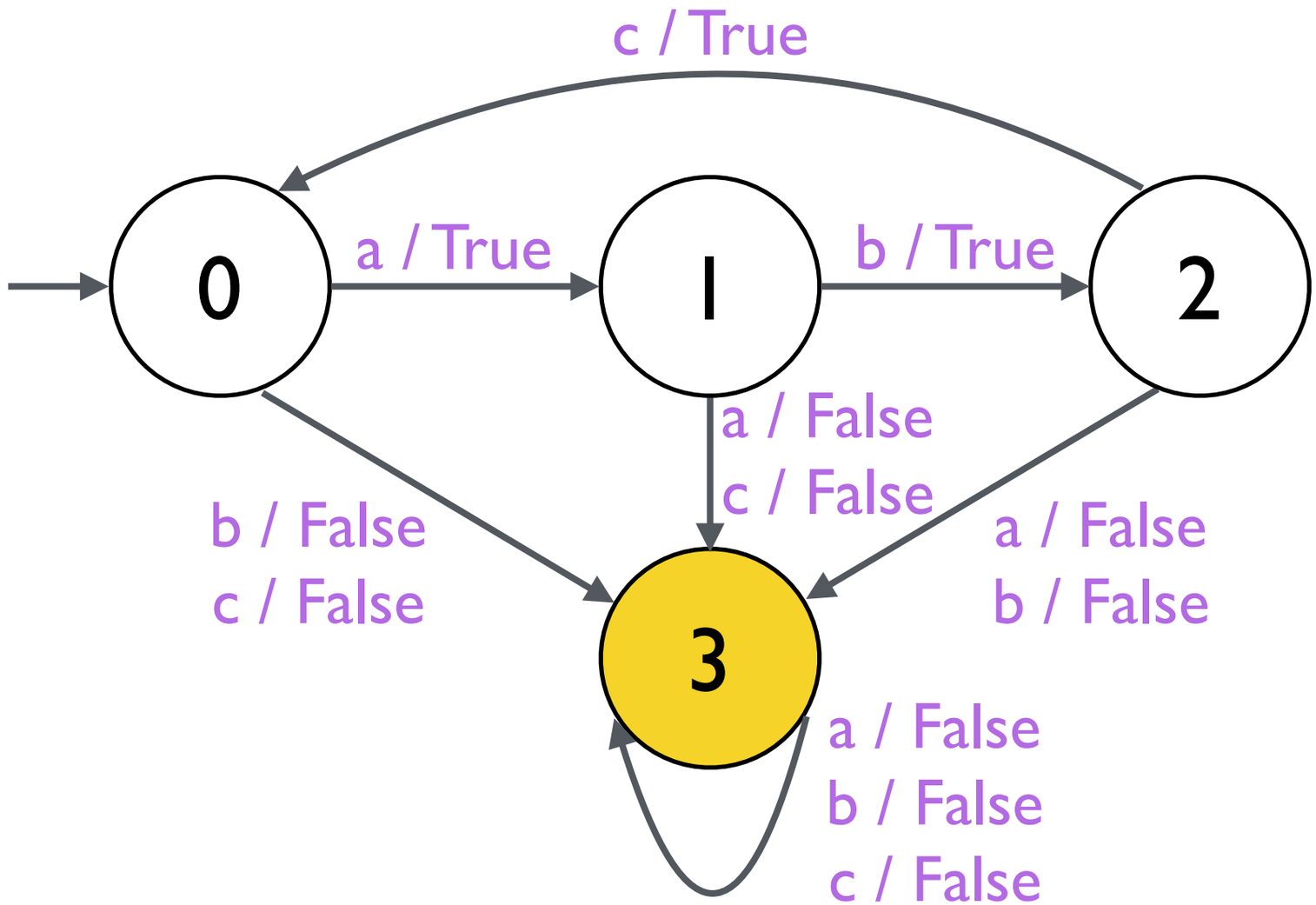
input:	*a*	*b*	*c*	*c*	*c*
output:	*True*	*True*	*True*		

Language recogniser: *abcabcabcabc…*



input:	*a*	*b*	*c*	*c*	*c*
output:	*True*	*True*	*True*	*False*	

Language recogniser: *abcabcabcabc…*



input:	*a*	*b*	*c*	*c*	*c*
output:	*True*	*True*	*True*	*False*	*False*

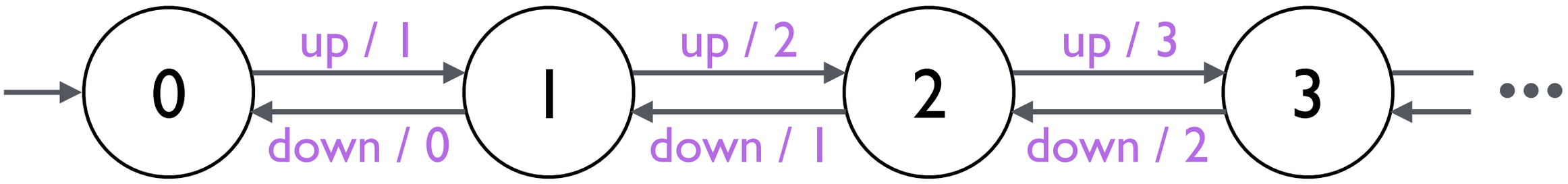
Infinite state machine: *up and down*

up down up up down

transduce

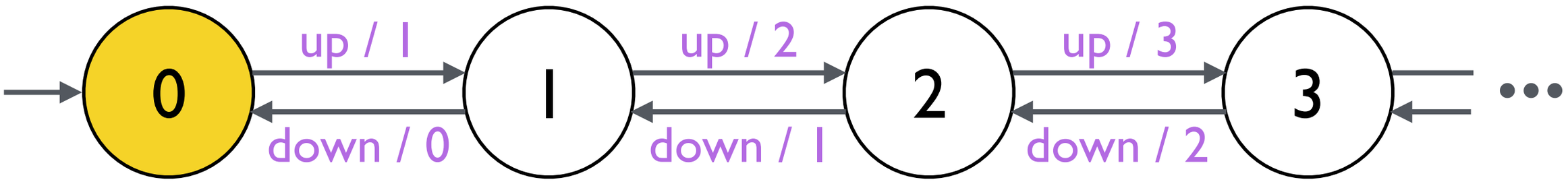
1 0 1 2 1

Infinite state machine: *up and down*



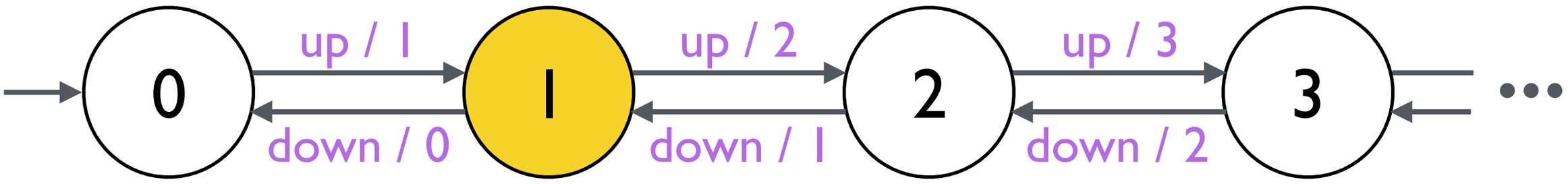
input:	*up*	*up*	*up*	*down*	*down*	*up*
output:						

Infinite state machine: *up and down*



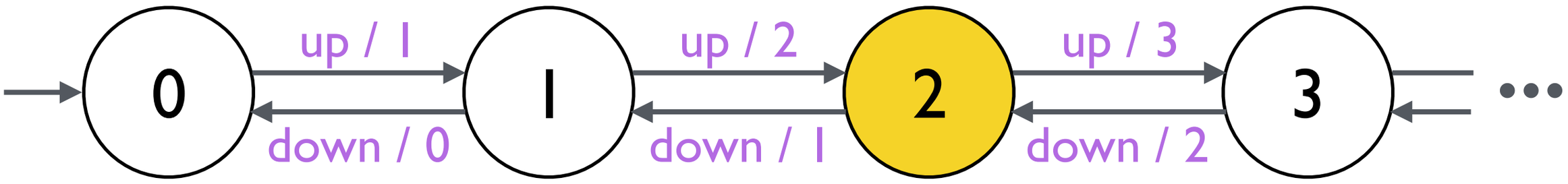
input:	*up*	*up*	*up*	*down*	*down*	*up*
output:						

Infinite state machine: *up and down*



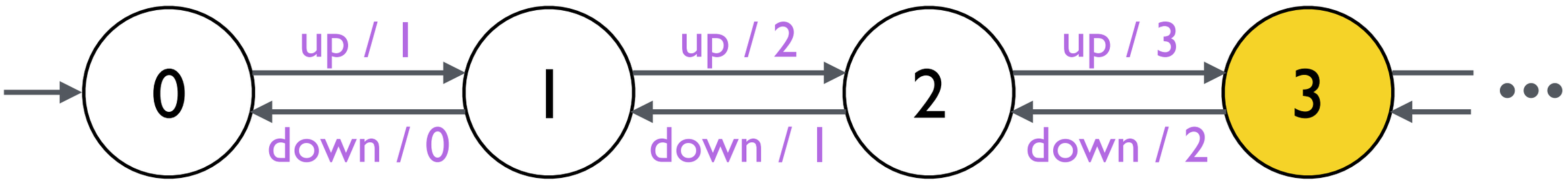
input:	*up*	*up*	*up*	*down*	*down*	*up*
output:	*1*					

Infinite state machine: *up and down*



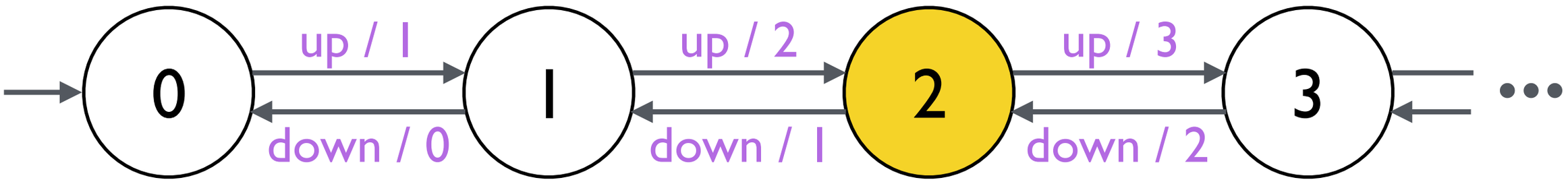
input:	*up*	*up*	*up*	*down*	*down*	*up*
output:	*1*	*2*				

Infinite state machine: *up and down*



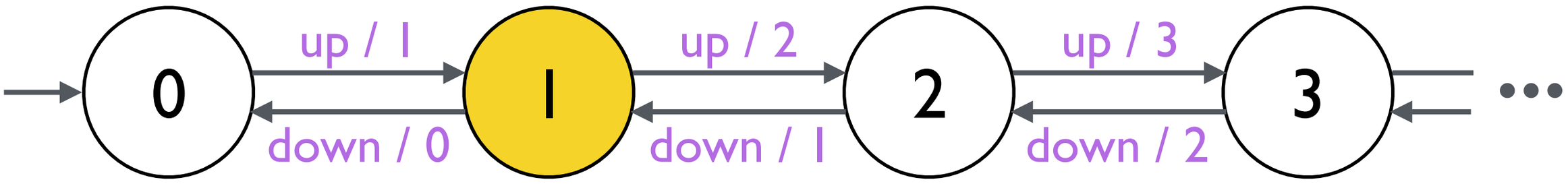
input:	*up*	*up*	*up*	*down*	*down*	*up*
output:	*1*	*2*	*3*			

Infinite state machine: *up and down*



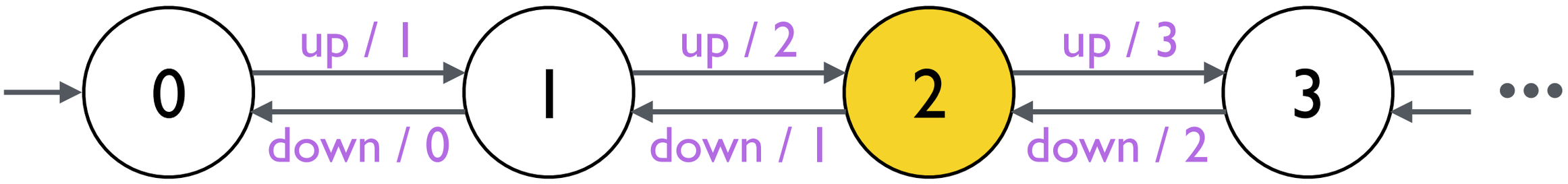
input:	*up*	*up*	*up*	*down*	*down*	*up*
output:	*1*	*2*	*3*	*2*		

Infinite state machine: *up and down*



input:	*up*	*up*	*up*	*down*	*down*	*up*
output:	*1*	*2*	*3*	*2*	*1*	

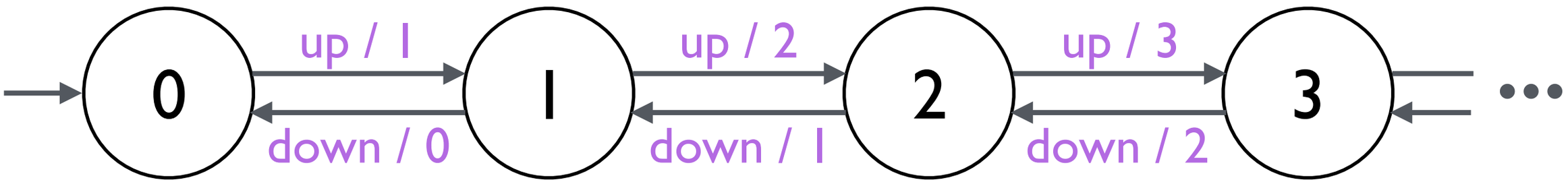
Infinite state machine: *up and down*



input:	*up*	*up*	*up*	*down*	*down*	*up*
output:	*1*	*2*	*3*	*2*	*1*	*2*

the outputs come from the transitions,
<u>not</u> the state IDs (even though they match in our model)

Infinite state machine: *up and down*



$$
\begin{aligned}
S &= \mathbb{N}_0 \\
I &= \{up, down\} \\
O &= \mathbb{N}_0 \\
n(s, i) &= \begin{cases} s+1 & \text{if } i == up \\ s-1 & \text{if } i == down \text{ and } s > 0 \end{cases} \\
o(s,i) &= n(s, i) \\
s0 &= 0
\end{aligned}
$$

State machines in Python

- we will implement state machines using the libdw library

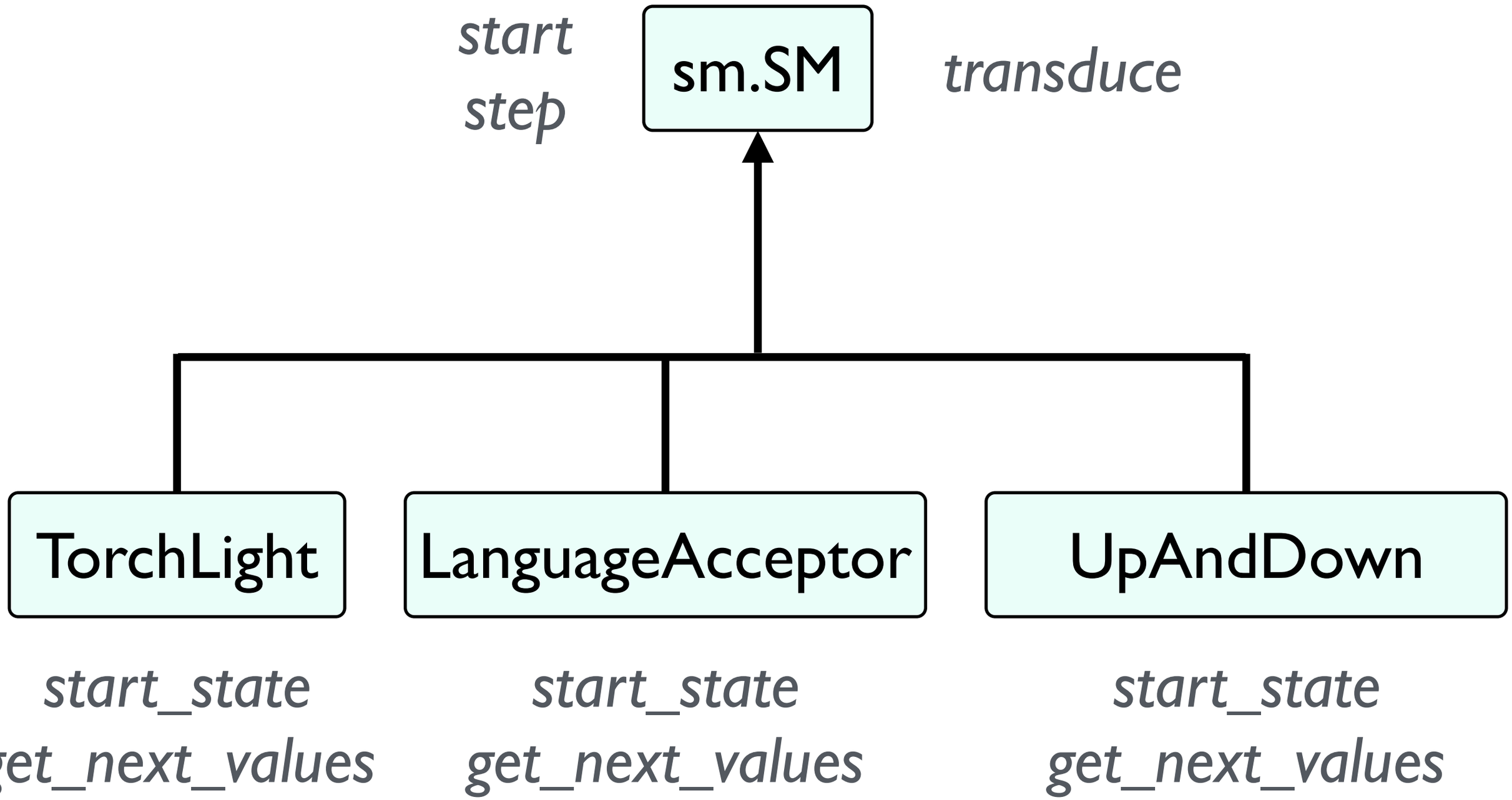
- libdw provides a class called sm.SM that we inherit from

  => *inherit several useful methods: s.start(), s.step(i), s.transduce(list)*

- we then define the start state via attribute start_state, and the transitions via query get_next_values(self, state, inp)

  => *these are used by the inherited methods (start, step, transduce, …)*
  to realise the state machine

Inheritance hierarchy for state machines



Summary

- state machines model systems for which the output depends on their history of inputs

- they perform a transduction from input streams to output streams

- we can express state machines in Python as classes that inherit from sm.SM (in the libdw library)

- we define transitions in those subclasses, inheriting general methods that fire/transduce them from sm.SM