Robustness Analysis of Networked Systems

Pascal Berrang

Are they reliable?

- Verification:
  System is **correct** or **incorrect**.

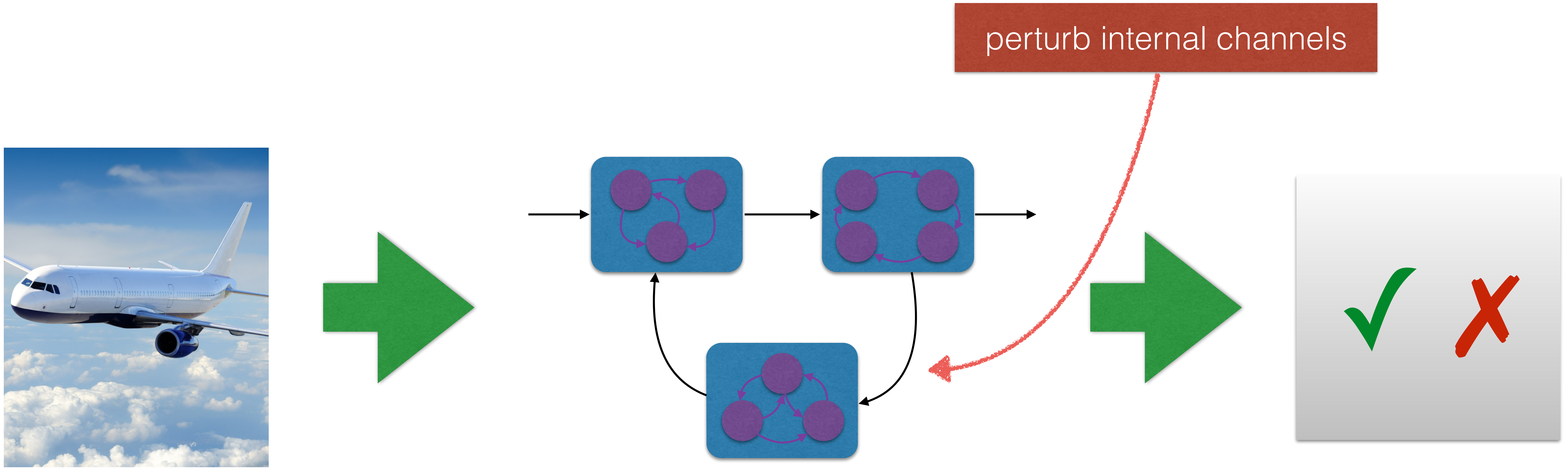
- Robustness:
  considers *uncertainty.*

Sensors

Network Channels

Software

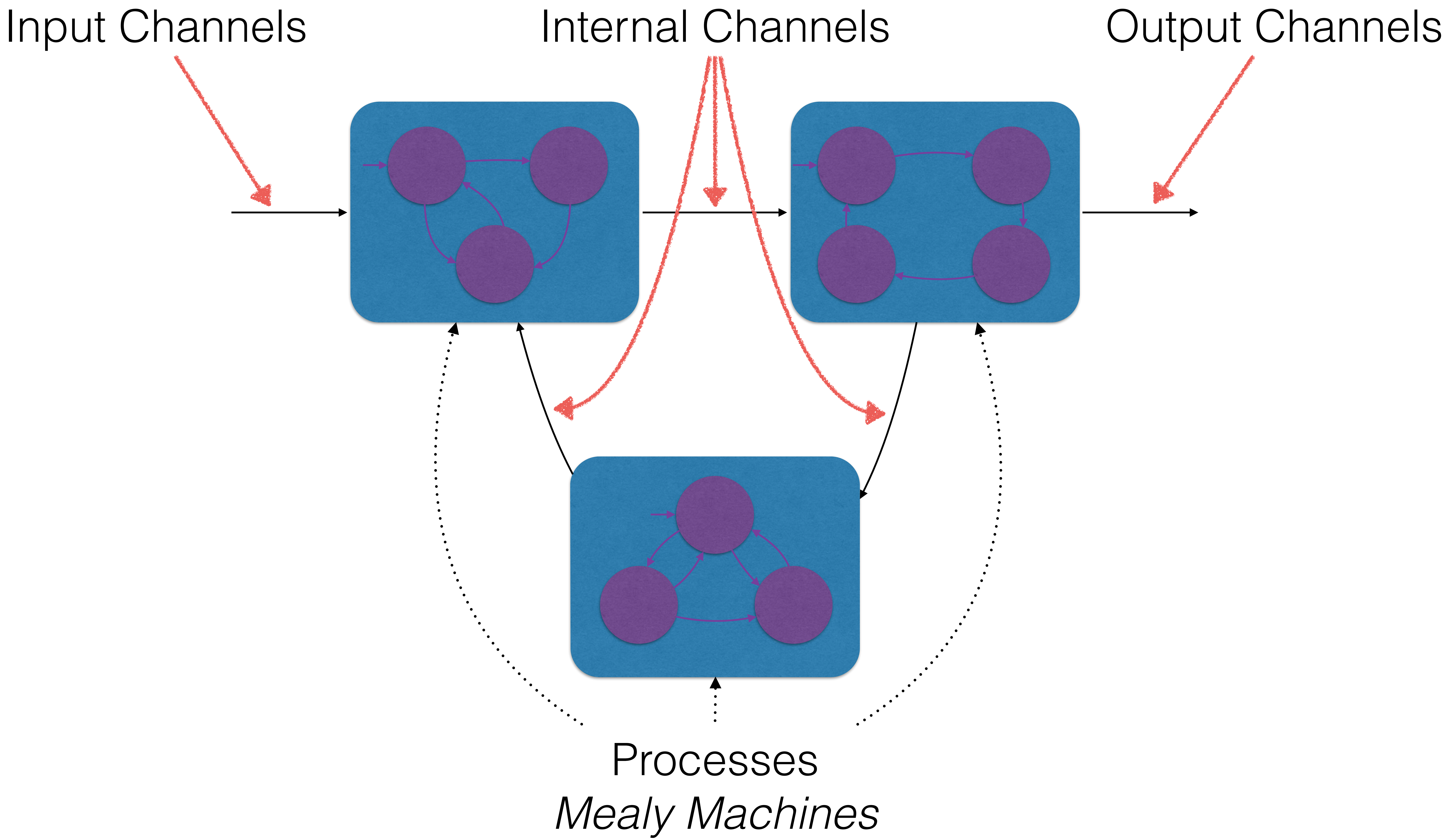
„Small perturbations to the environment or parameters do not change the observable behavior substantially.“



Networked System

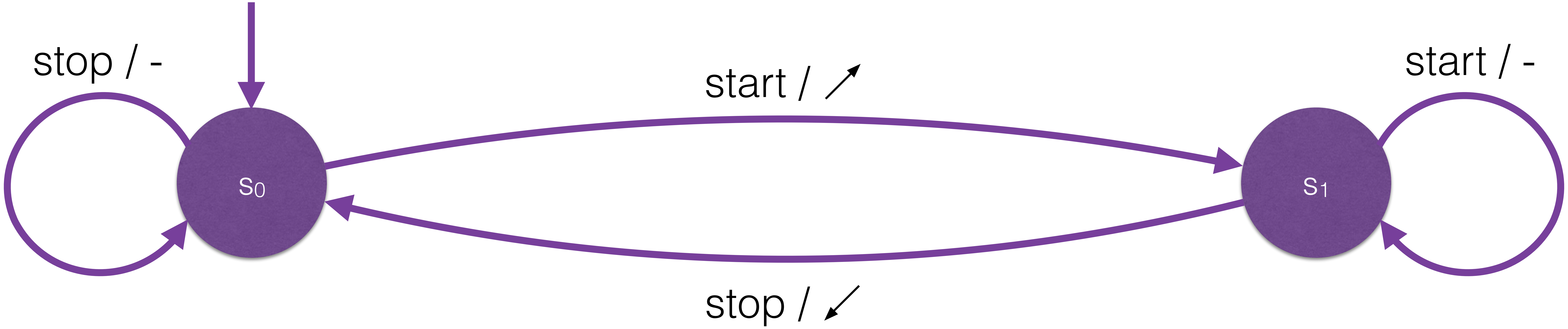
Model

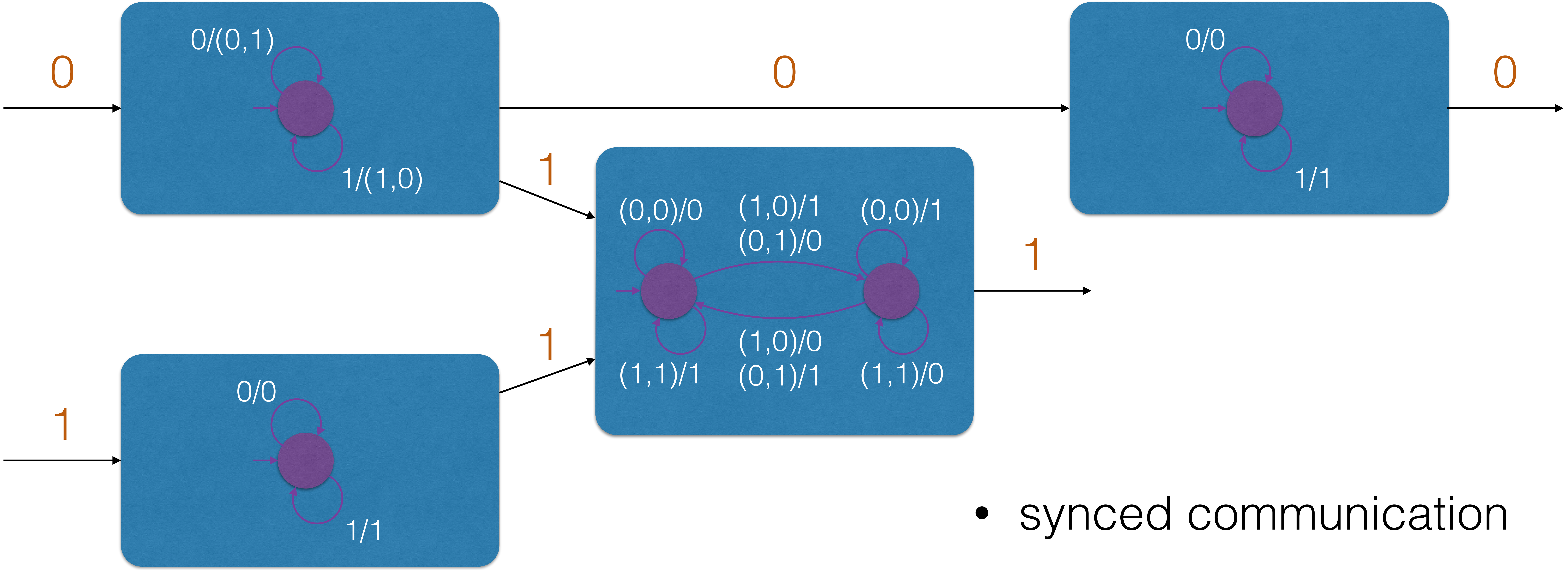
Check Robustness


Input Channels
Internal Channels
Output Channels
Processes
Mealy Machines

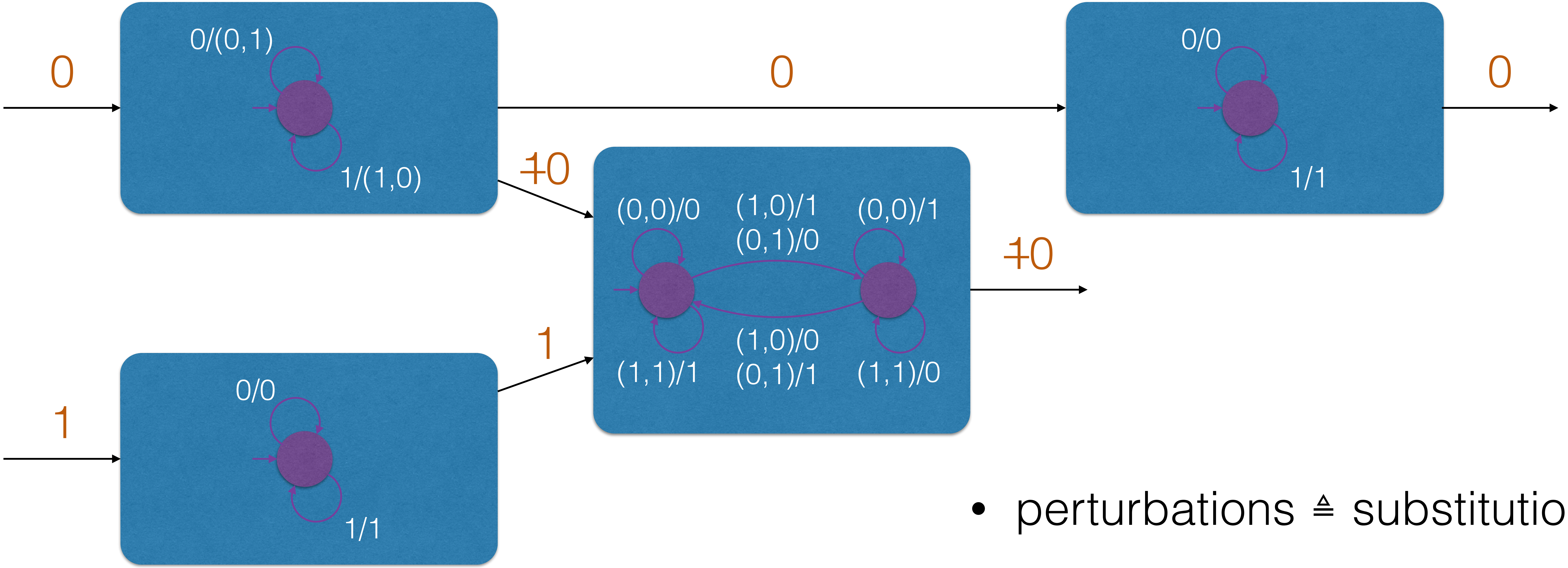
Mealy Machines

Input: *start, start, stop, stop*
Output: ↗, -, ↙, -





- synced communication
- instant message delivery



- perturbations ≜ substitutions
- deletions ≜ extra symbol

(δ,ε)-robustness

- **if** perturbations ≤ δ **then** error in output channels ≤ ε
- error measure: *d(normal output, perturbed output)*
  - Levenshtein distance
  - L_1 distance

Levenshtein distance

min. #insertions + #deletions + #substitutions

d(house, mode) = 3

			h	o	u	s	e
		0	1	2	3	4	5
	0	0	1	2	3	4	5
m	1	1	1	2	3	4	5
o	2	2	2	1	2	3	4
d	3	3	3	2	2	3	4
e	4	4	4	3	3	3	3

dynamic programming

L_1 distance

#differing positions

d(house, mode) = 4

h	o	u	s	e
m	o	d	e	#
1	0	1	1	1

Levenshtein distance

min. #insertions + #deletions + #substitutions

d(house, mode) = 3

			h	o	u	s	e
		0	1	2	3	4	5
	0	0	1				
m	1	1	1				
o	2						
d	3						
e	4						

dynamic programming

L_1 distance

#differing positions

d(house, mode) = 4

h	o	u	s	e
m	o	d	e	#
1	0	1	1	1

Levenshtein distance

min. #insertions + #deletions + #substitutions

$$d(\mathit{house}, \mathit{mode}) = 3$$

			h	o	u	s	e
		0	1	2	3	4	5
	0	0	1	2			
m	1	1	1	2			
o	2	2	2	1			
d	3						
e	4						

dynamic programming

L_1 distance

#differing positions

$$d(\mathit{house}, \mathit{mode}) = 4$$

h	o	u	s	e
m	o	d	e	#
1	0	1	1	1

Levenshtein distance

min. #insertions + #deletions + #substitutions

d(house, mode) = 3

			h	o	u	s	e
		0	1	2	3	4	5
	0	0	1	2	3	4	5
m	1	1	1	2	3	4	5
o	2	2	2	1	2	3	4
d	3	3	3	2	2	3	4
e	4	4	4	3	3	3	3

dynamic programming

L_1 distance

#differing positions

d(house, mode) = 4

h	o	u	s	e
m	o	d	e	#
1	0	1	1	1

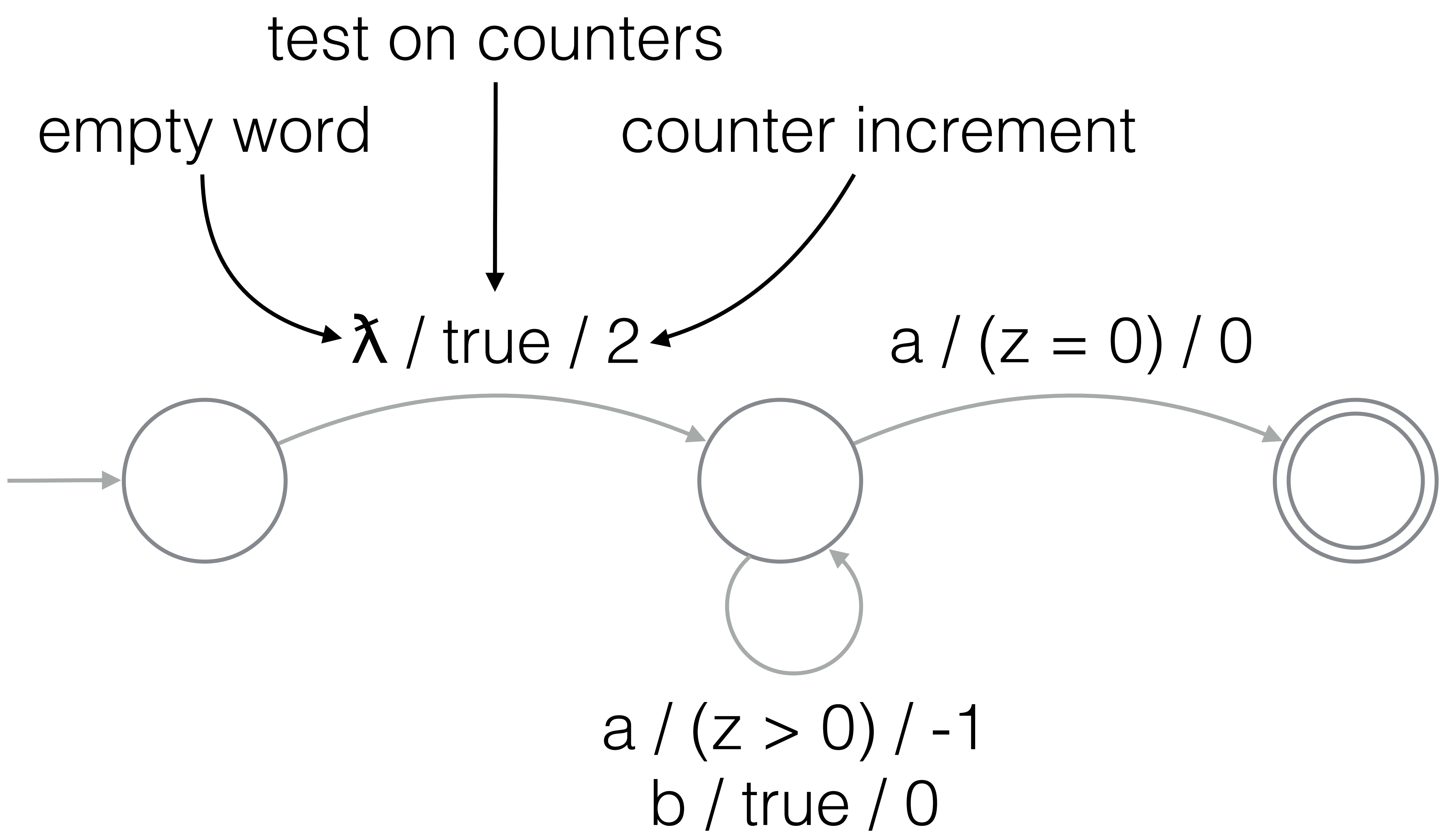
perturb internal channels
𝒜
keep track of δ, ε
accept if **not** (δ,ε)-robust
emptiness
✓ ✗
of 𝔏(𝒜)

emptiness ✓ ✗ of $\mathfrak{L}(\mathscr{A})$

⇒ easy for *Reversal-bounded Counter Machines*

constants = {-1, 0, 1, 2}
counters = {z}

r-reversal bounded ≜ counters change from <u>increment to decrement</u> (or vice versa) <u>at most r times</u>



accepts, if the input contains 3 a's

keep track of ε

Levenshtein distance

min. #insertions + #deletions + #substitutions

			c	c	f	f	#
		0	1	2	3	4	5
	0	0	1	2			
a	1	1	1	2	⊤		
c	2	2	1	1	2	⊤	
b	3		2	2	2	⊤	⊤
c	4			2	⊤	⊤	⊤
d	5				⊤	⊤	⊤

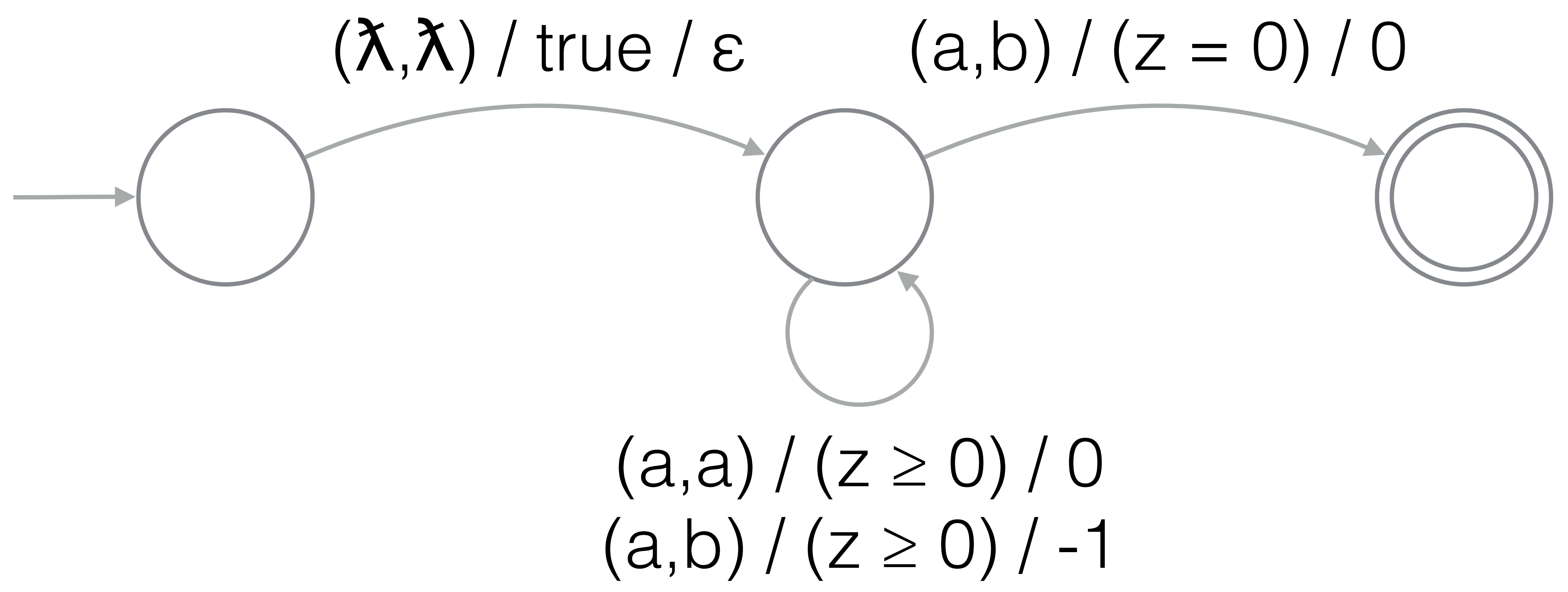
accepts, if
Levenshtein distance
for input > ε

keep track of ε

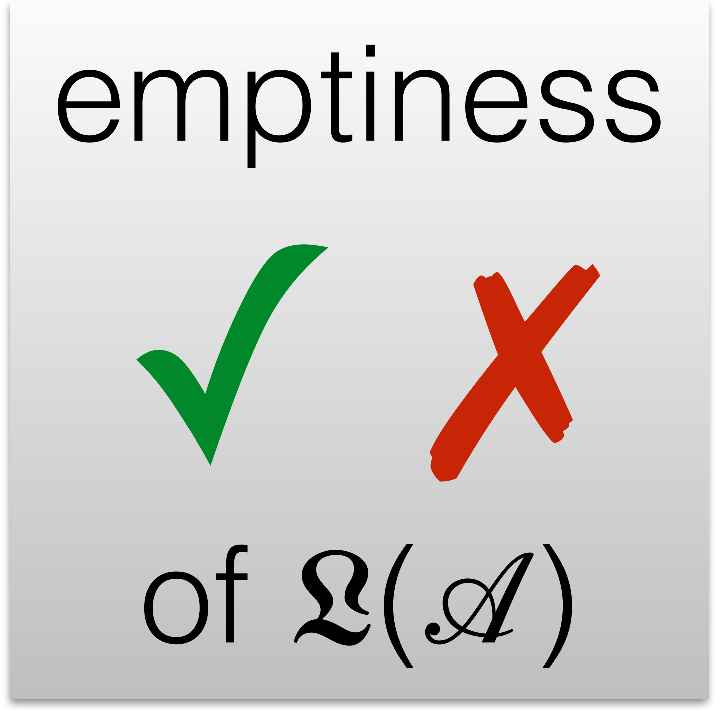
L_1 distance

#differing positions

constants = {ε, 0, -1}
counters = {z}



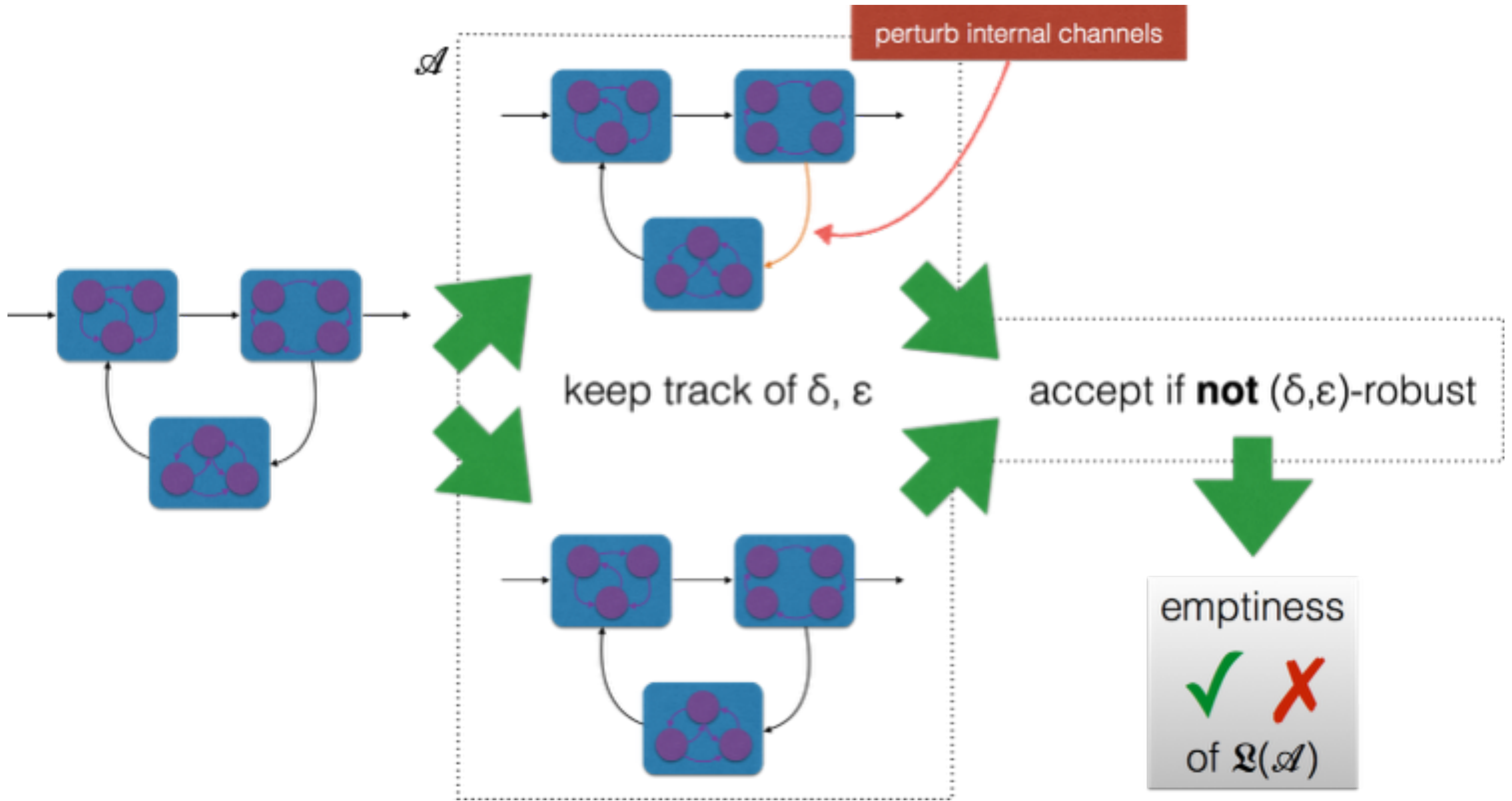
accepts, if L_1 distance for input > ε



- $\mathscr{A}^{\delta,\varepsilon}$ certifies *non-robustness*

- Input: string *s*
  - simulate unperturbed execution
  - simulate perturbed execution
  - keep track of the perturbations
  - keep track of the distance of the outputs

➡ 1-reversal-bounded counter machine



Limitations

- digital signals:
  - *d(house, mouse)* = 1
  - *d(10, 9)* = ?
- uncertainty:

Sensors

Network Channels

Software

Conclusion

- Networked systems often safety critical.
- Robustness is crucial in networked systems!
- Easy model for error-prone networks.
- Many distance metrics possible.
- Possible extension: generalize error model.

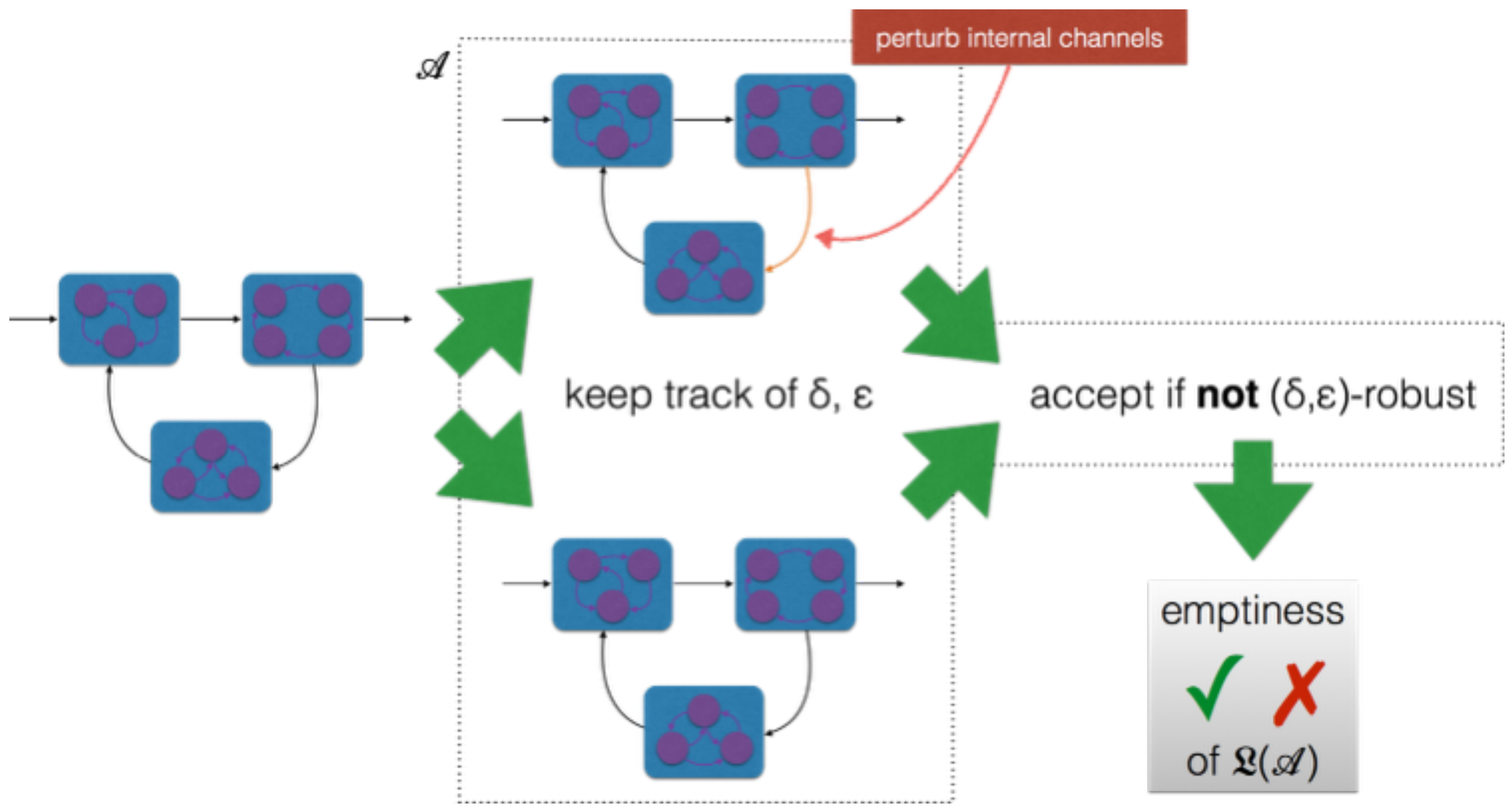


Image Sources

- Car (Audi A1) - http://www.extremetech.com/wp-content/uploads/2012/12/Audi-A1.jpg
- Power Plant - http://upload.wikimedia.org/wikipedia/commons/8/8d/Nuclear_Power_Plant_-_Grohnde_-_Germany_-_1-2.JPG
- Aircraft - http://cdns.designmodo.com/wp-content/uploads/2010/09/CivilAircraft_005019.jpg