CREST

A Continuous, REactive SysTems DSL

Stefan Klikovits Alban Linard Didier Buchs

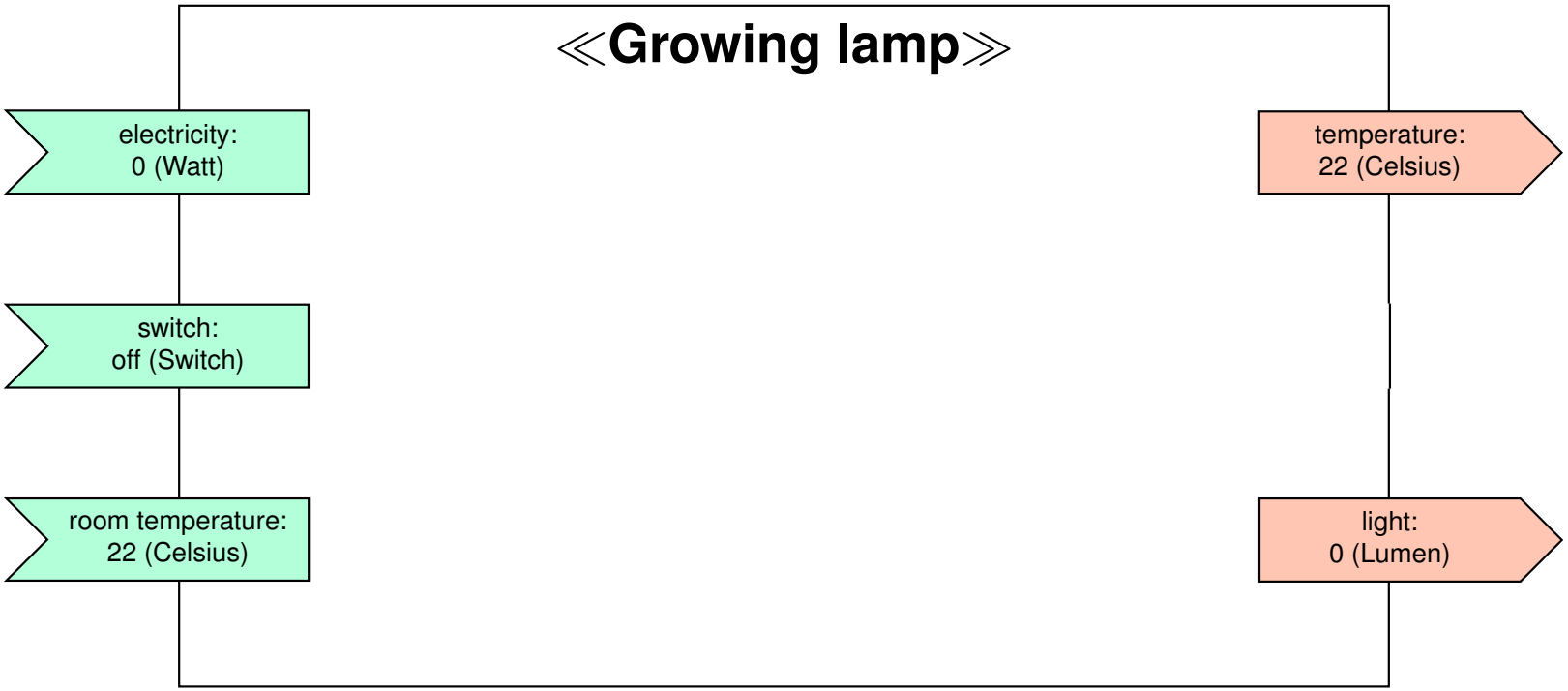
University of Geneva, Switzerland

Growing plants

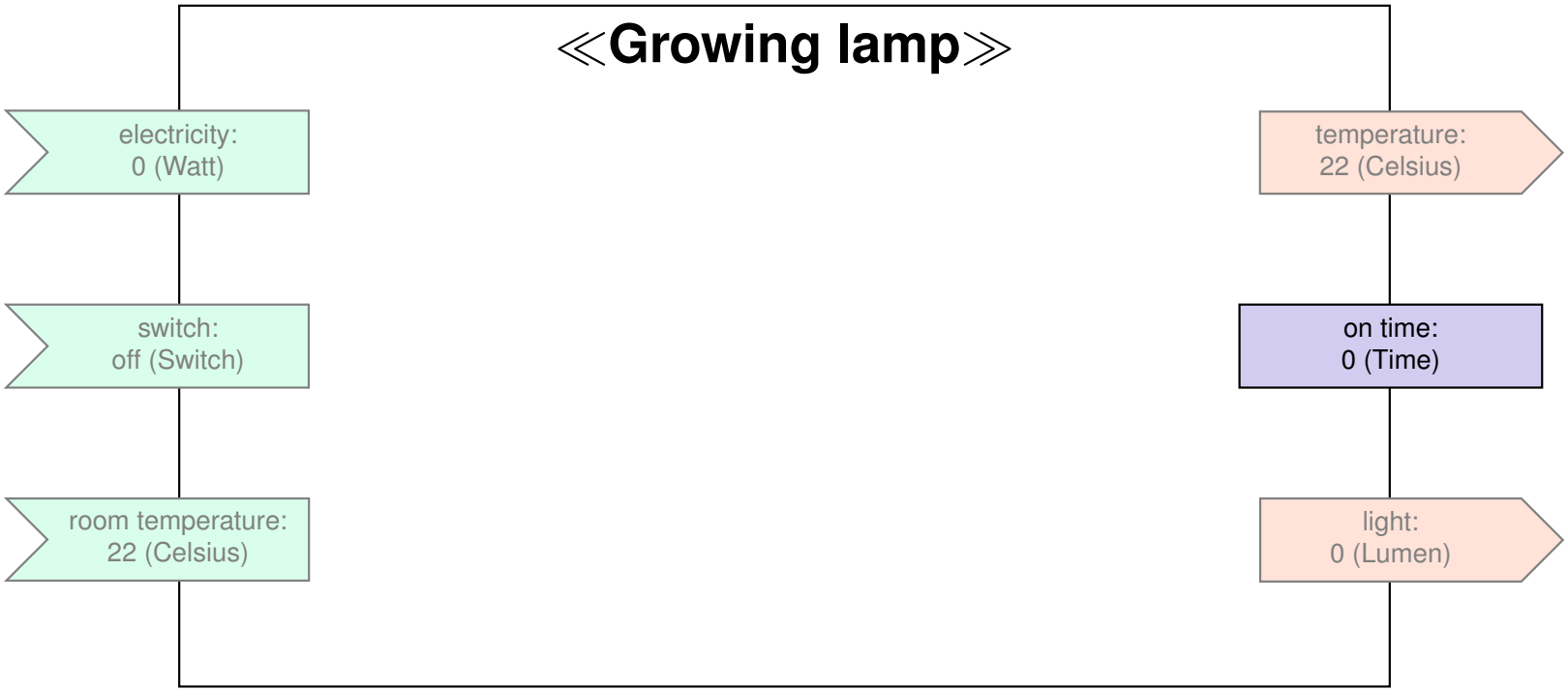
Growing plants

(c) Debra Roby
https://www.flickr.com/photos/darinhercules/

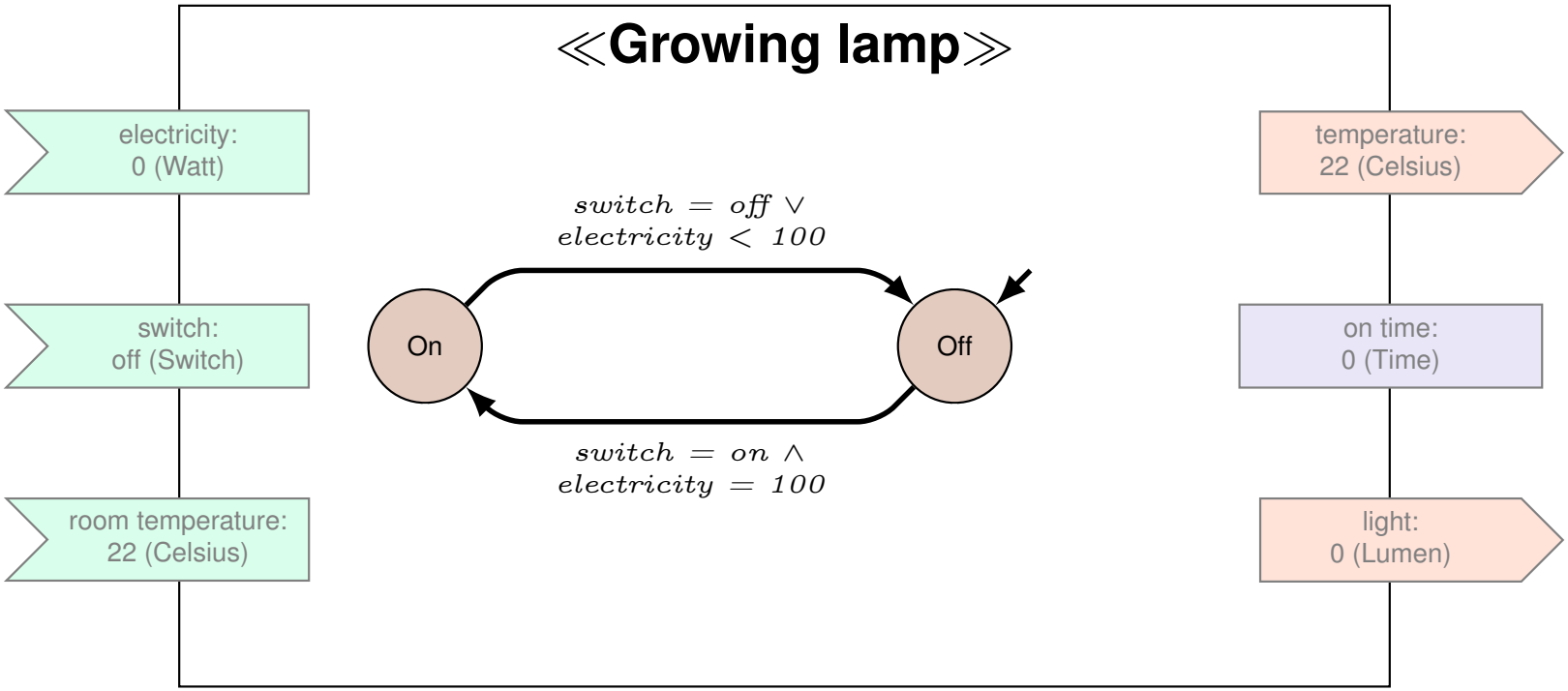
CREST diagrams



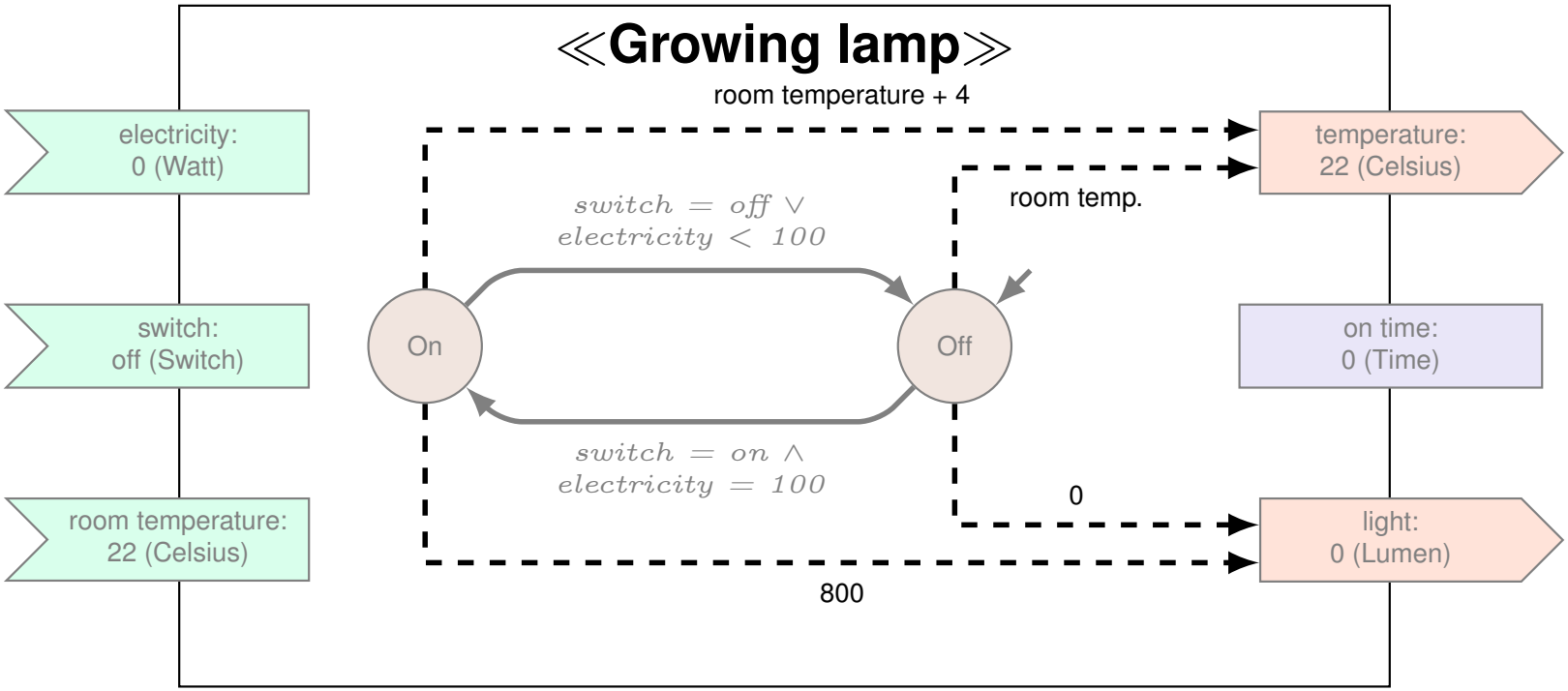
CREST diagrams



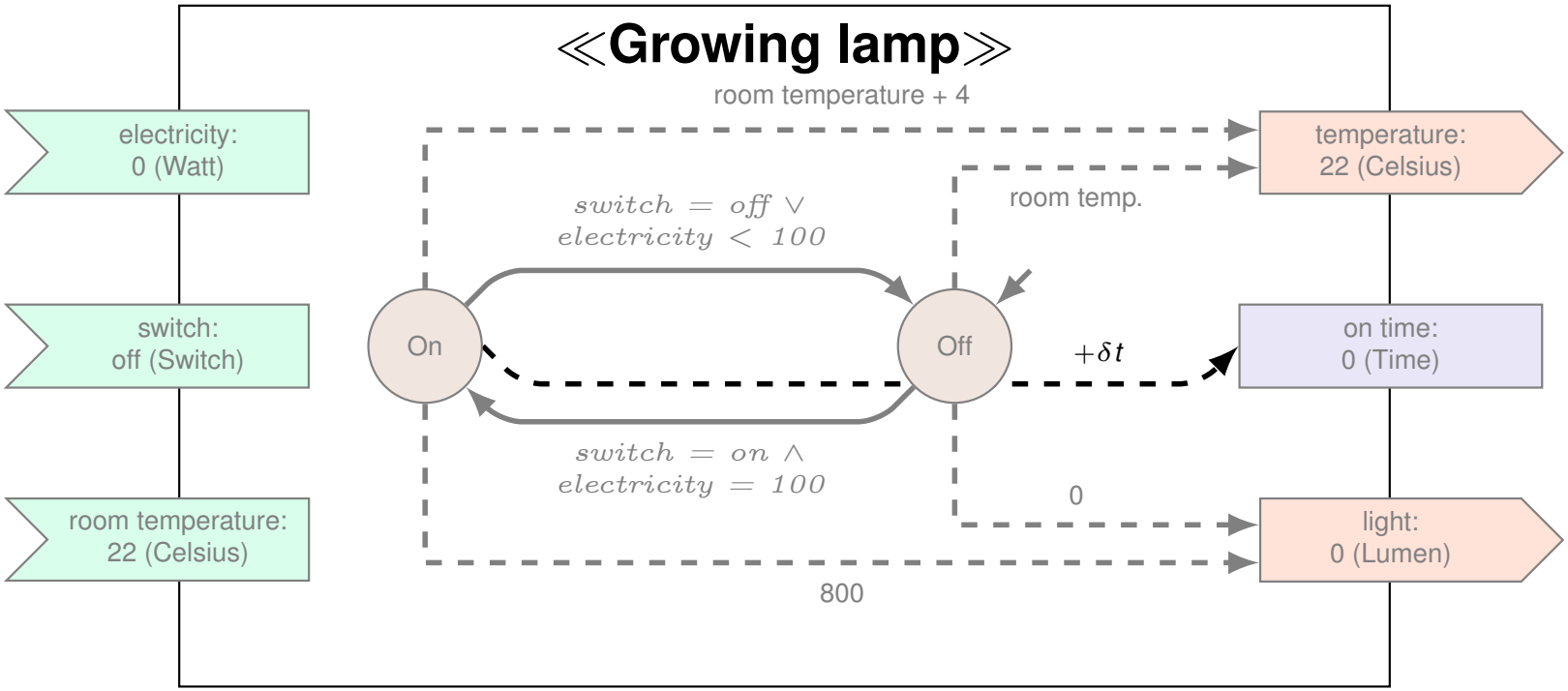
CREST diagrams



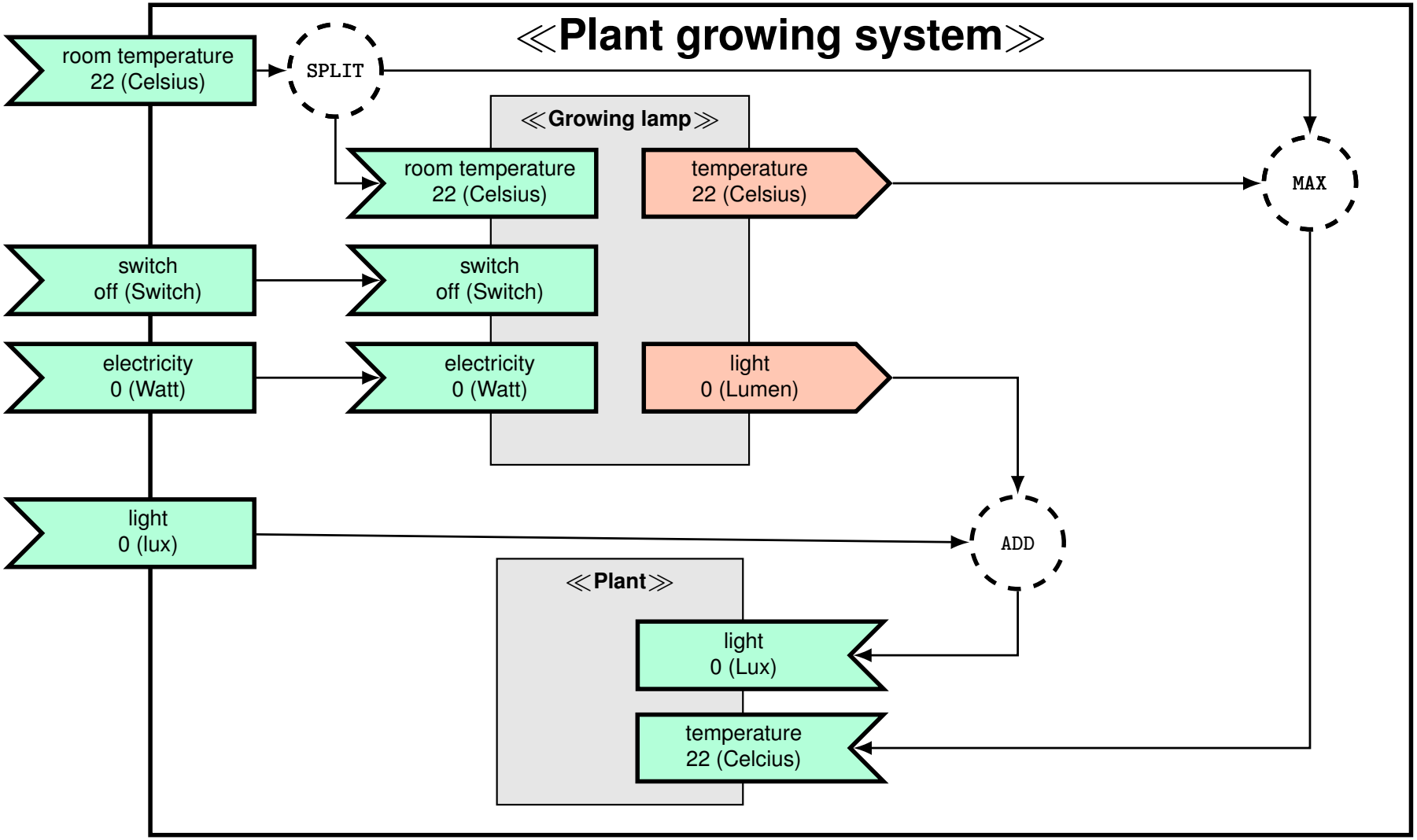
CREST diagrams



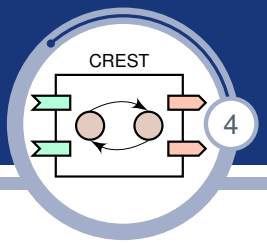
CREST diagrams



Composition



Composition



≪Plant growing system≫

room temperature
22 (Celsius)

SPLIT

≪Growing lamp≫

room temperature
22 (Celsius)

temperature
22 (Celsius)

MAX

switch
off (Switch)

switch
off (Switch)

electricity
0 (Watt)

electricity
0 (Watt)

light
0 (Lumen)

light
0 (lux)

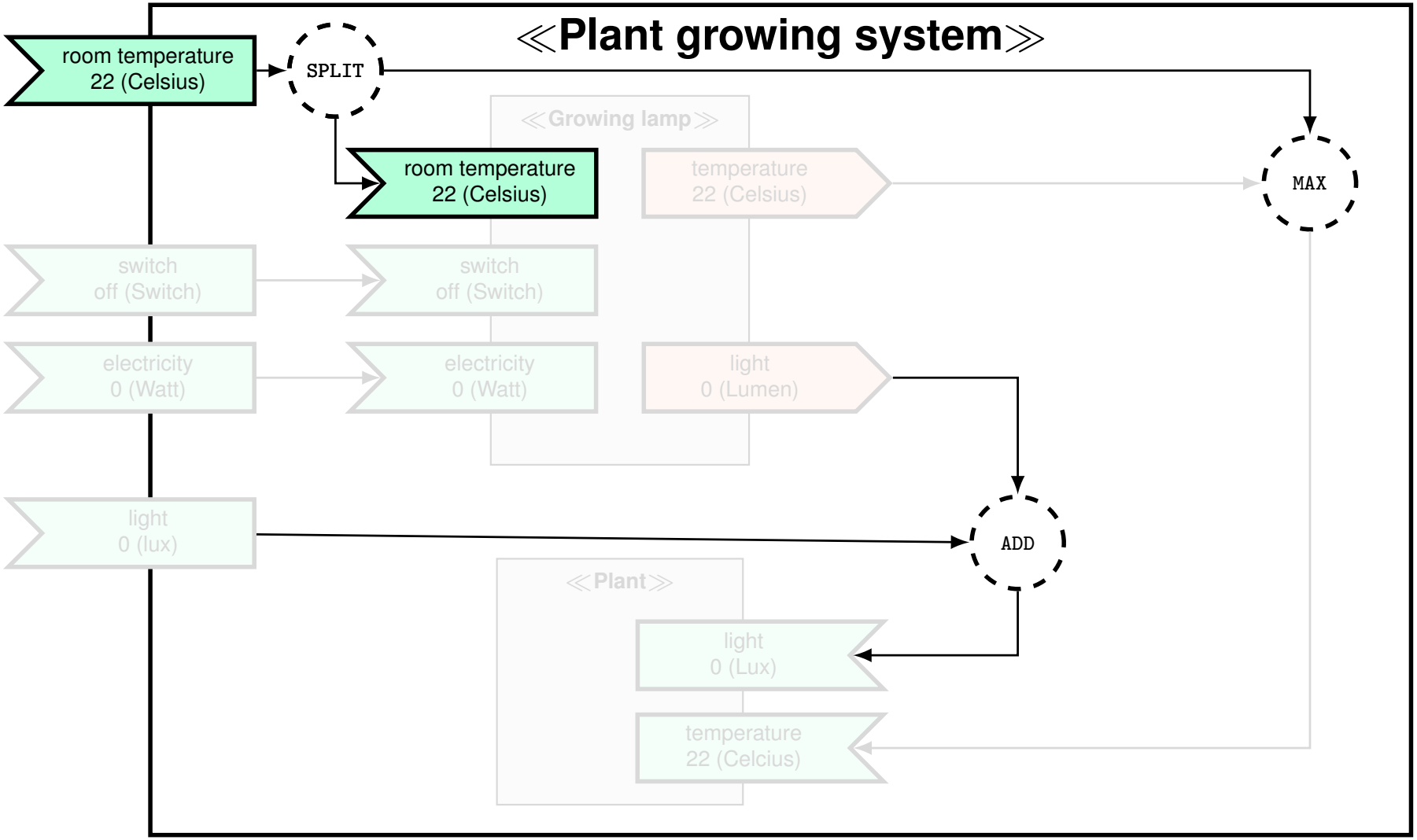
ADD

≪Plant≫

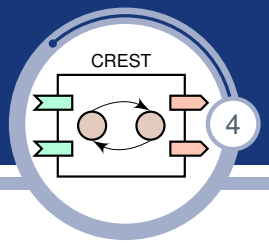
light
0 (Lux)

temperature
22 (Celcius)

Composition



Composition



≪Plant growing system≫

room temperature
22 (Celsius)

SPLIT

≪Growing lamp≫

room temperature
22 (Celsius)

temperature
22 (Celsius)

MAX

switch
off (Switch)

switch
off (Switch)

electricity
0 (Watt)

electricity
0 (Watt)

light
0 (Lumen)

≪ADD≫

lamp
0 (Lumen)

add

light
0 (lux)

external
0 (Lux)

lamp / 0.25
+external

≪Plant≫

light
0 (Lux)

light
0 (Lux)

temperature
22 (Celcius)

Language Realization


```
class GrowingLamp(Entity):
```

Language Realization


```
class GrowingLamp(Entity):

  electricity = Input(resource=res_electricity, init=0)
  light       = Output(resource=res_light_lumen, init=0)
```

Language Realization


```
class GrowingLamp(Entity):

  electricity = Input(resource=res_electricity, init=0)
  light       = Output(resource=res_light_lumen, init=0)

  on = current = State()
  off = State()
```

Language Realization


```
class GrowingLamp(Entity):

  electricity = Input(resource=res_electricity, init=0)
  light       = Output(resource=res_light_lumen, init=0)

  on = current = State()
  off = State()

  off_to_on = Transition(source=off, target=on,
        guard=(lambda lamp: lamp.switch.value == "on" and
                            lamp.electricity.value >= 100)
  )
```

Language Realization


```
class GrowingLamp(Entity):

  electricity = Input(resource=res_electricity, init=0)
  light       = Output(resource=res_light_lumen, init=0)

  on = current = State()
  off = State()

  off_to_on = Transition(source=off, target=on,
        guard=(lambda lamp: lamp.switch.value == "on" and
                            lamp.electricity.value >= 100)
  )

  @update(state=on)
  def set_light_on(lamp, dt=0):
      lamp.light.value = 800
```

Why not «formalism-XYZ»?

- powerful, but complex
- too generic
- feature workarounds
- architecture <u>and</u> behaviour

One DSL to rule them all?

CREST Formalization

$$\langle \mathbb{T}, \mathcal{R}, \mathcal{E}, e \rangle$$

$$e = \langle P_e, resource_e, TS_e, U_e, entities_e, Inf_e \rangle$$

$$P_e = I_e \sqcup O_e \sqcup L_e$$

$$TS = \langle S_e, \rightarrow_e \rangle; \rightarrow_e \subseteq S_e \times S_e \times G_e$$

$$\ldots$$

CREST Formalization

$$\langle \mathbb{T}, \mathcal{R}, \mathcal{E}, e \rangle$$

$$e = \langle P_e, resource_e, TS_e, U_e, entities_e, Inf_e \rangle$$

$$P_e = I_e \sqcup O_e \sqcup L_e$$

$$TS = \langle S_e, \rightarrow_e \rangle; \rightarrow_e \subseteq S_e \times S_e \times G_e$$

$$\ldots$$

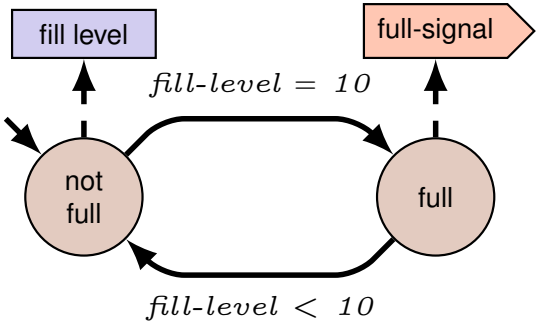
`it's in the paper`

How to design a simulator

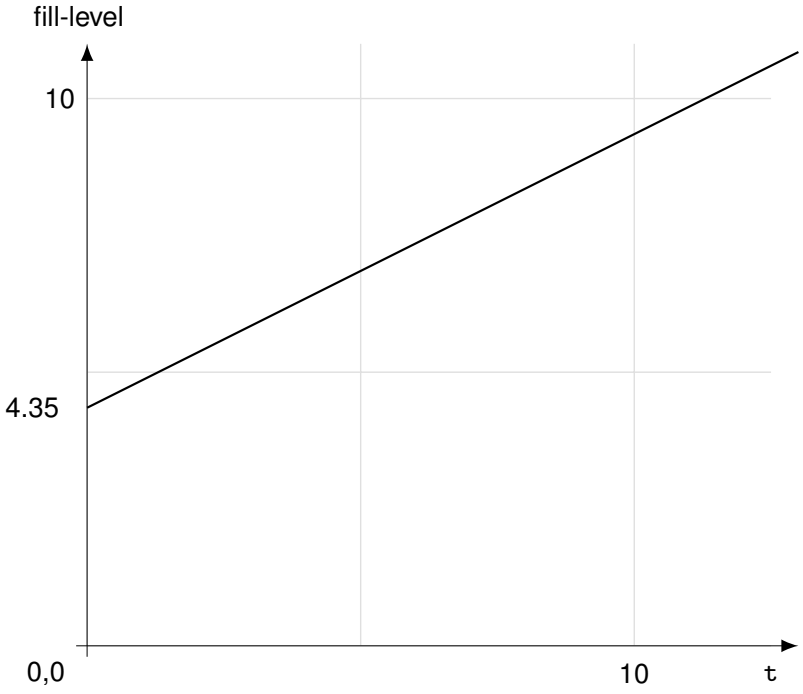
- advance time in constant steps?
- transition time calculation
- generate CREST diagrams

How to design a simulator

- advance time in constant steps?
- transition time calculation
- generate CREST diagrams

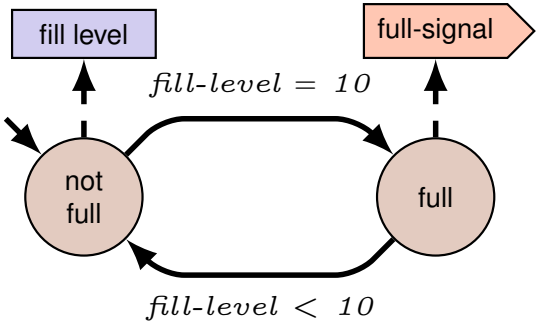


flow-rate: 0.5 per time step

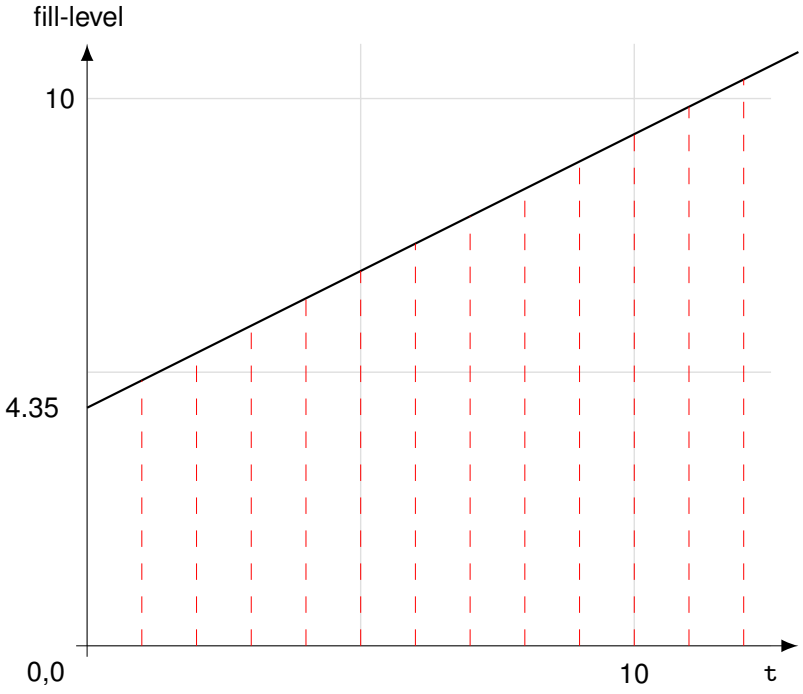


How to design a simulator

- advance time in constant steps?
- transition time calculation
- generate CREST diagrams

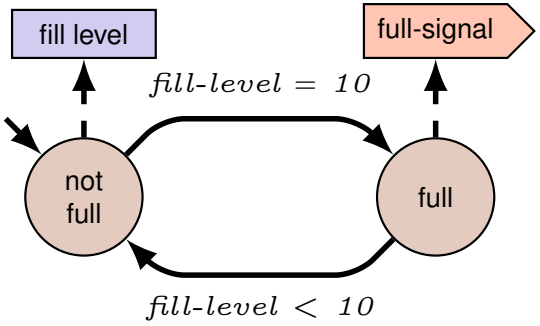


flow-rate: 0.5 per time step

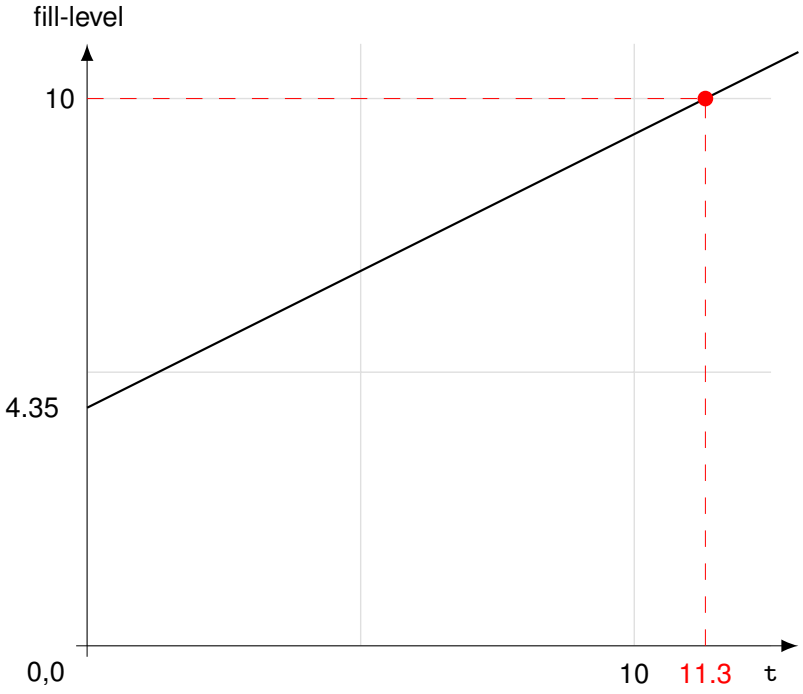


How to design a simulator

- advance time in constant steps?
- transition time calculation
- generate CREST diagrams



flow-rate: 0.5 per time step



Verification

- state space exploration
- zone/region-based verification

Verification

- state space exploration
- zone/region-based verification

http://www-i2.informatik.rwth-aachen.de/ sri/Slides/sri-zonebased.pdf

Verification

- state space exploration
- zone/region-based verification
- requirements language

Controller synthesis

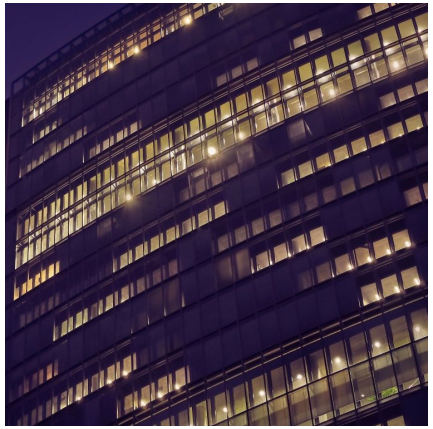
- planning + optimization + simulation
- changing parameters
- changing (sub-)systems

Applications

plant growing

home automation

job scheduling



office automation

https://cs.wikipedia.org/wiki/Roomba#/media/File:IRobot_Roomba_780.jpg

Conclusion

CREST

- architecture & behaviour
- continuous & reactive
- formal basis

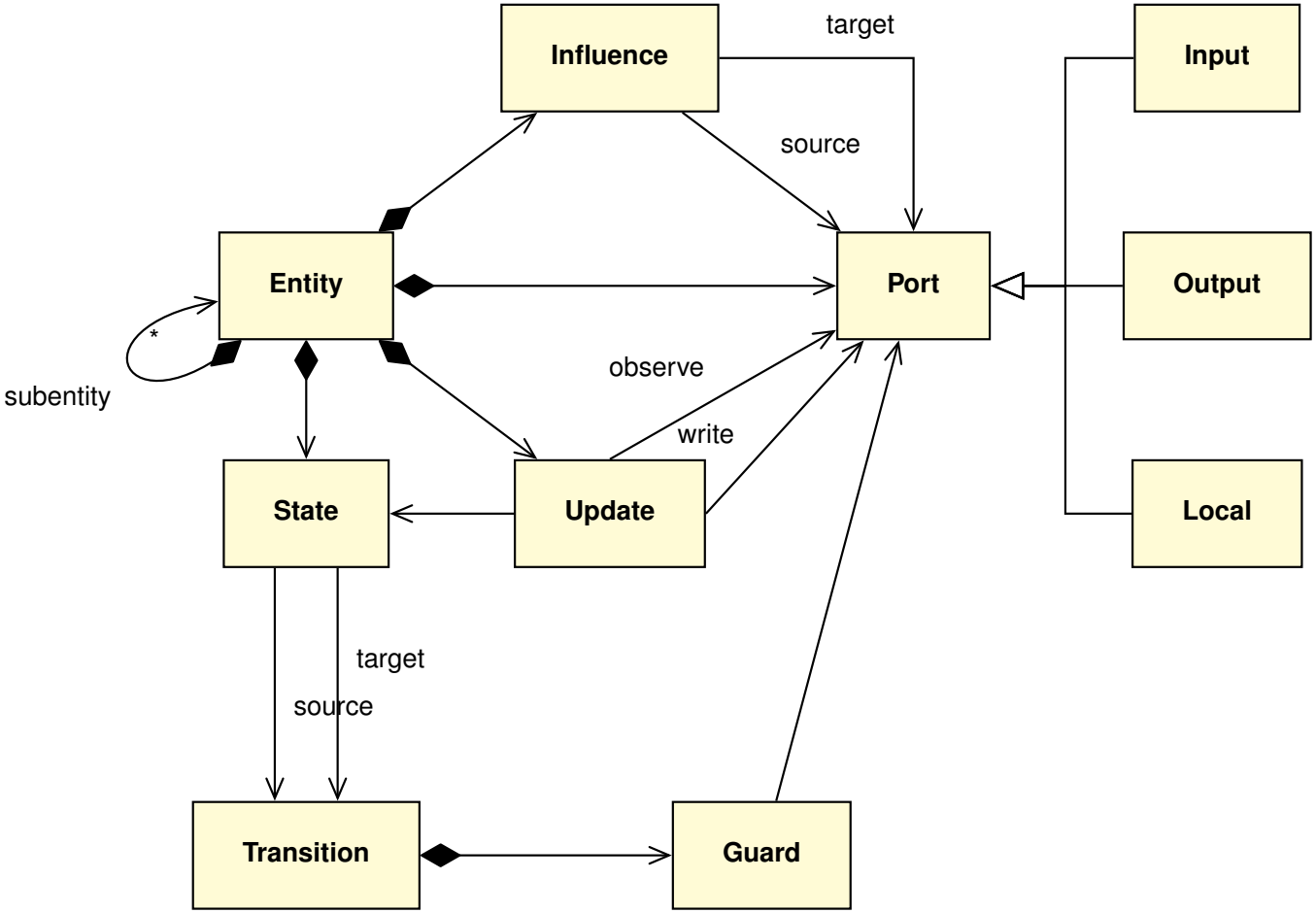
CREST

A Continuous, REactive SysTems DSL

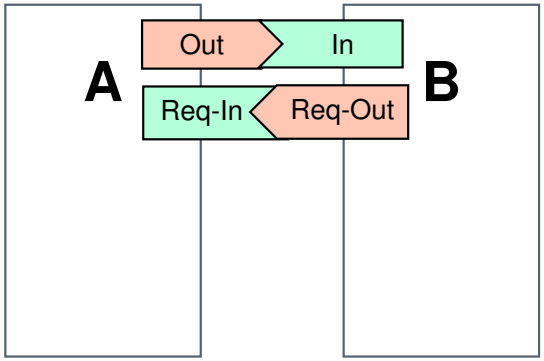
Stefan Klikovits Alban Linard Didier Buchs

University of Geneva, Switzerland

Meta-Model

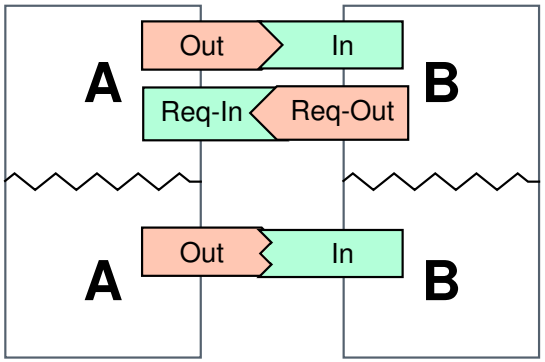

Influence
target
source
Input
Entity
Port
Output
*
subentity
observe
write
State
Update
Local
target
source
Transition
Guard

Complex Ports



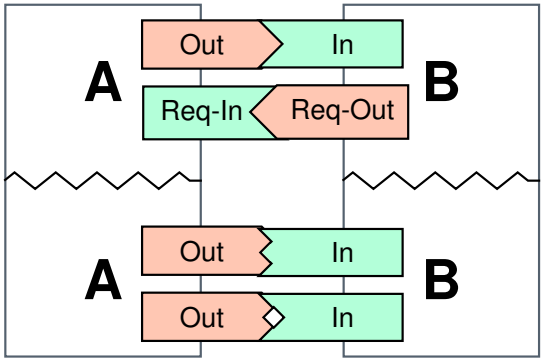
Requestable Resources

Complex Ports



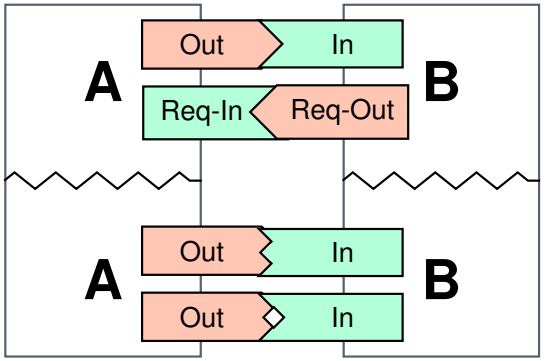
Requestable Resources

Complex Ports

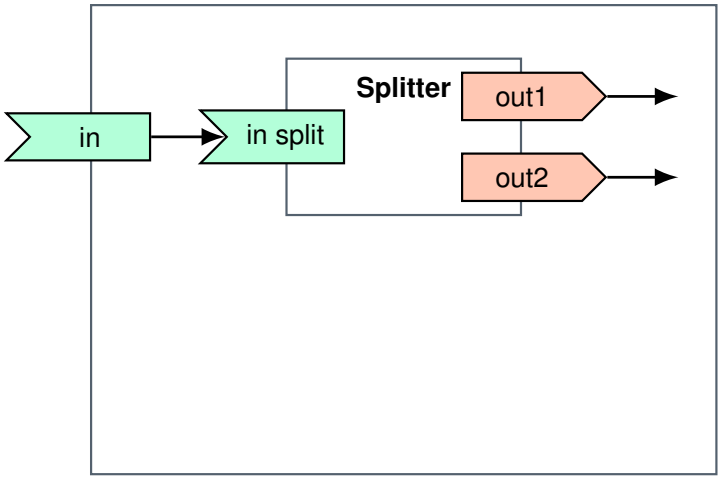


Requestable Resources

Complex Ports

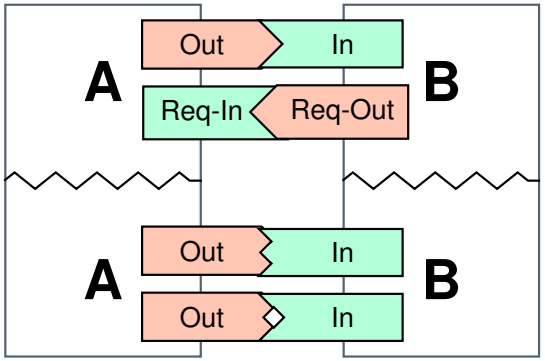


Requestable Resources

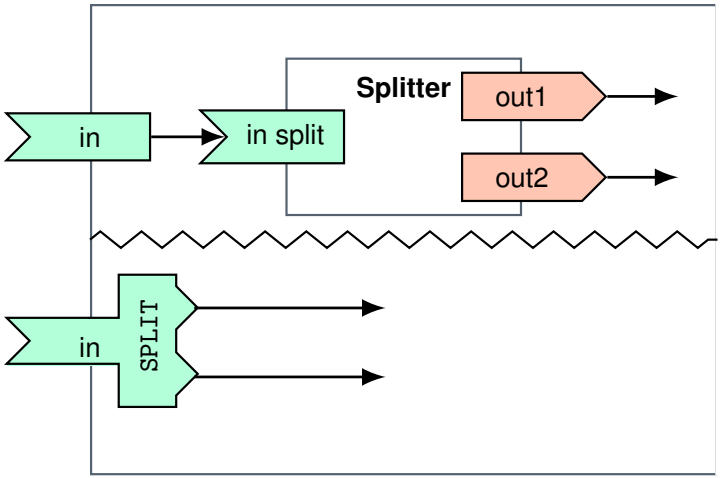


combined input & splitter

Complex Ports



Requestable Resources



combined input & splitter