

Classic DEVS

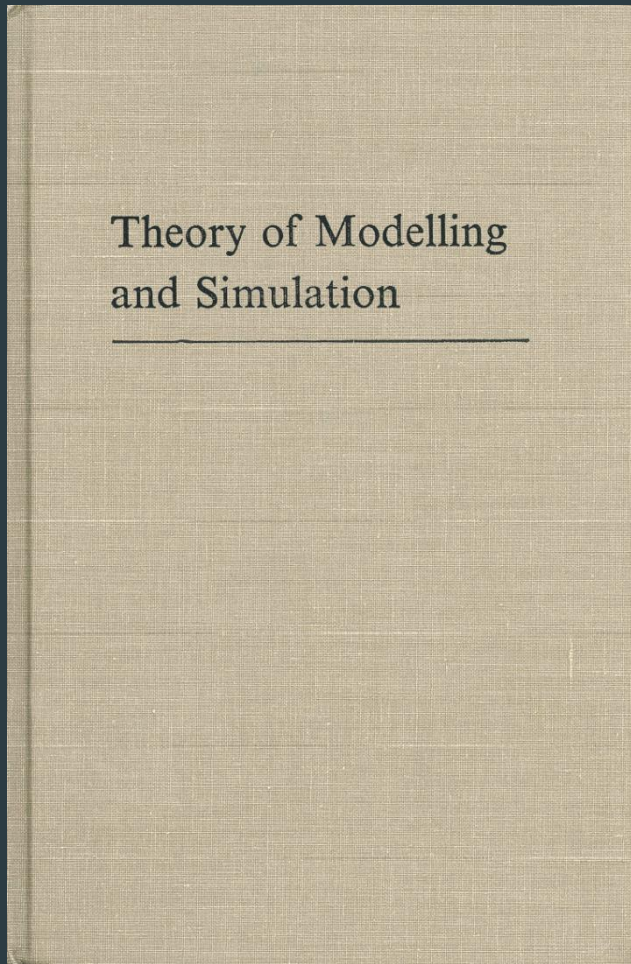
An Introduction Using PythonPDEVS

Yentl Van Tendeloo, Hans Vangheluwe

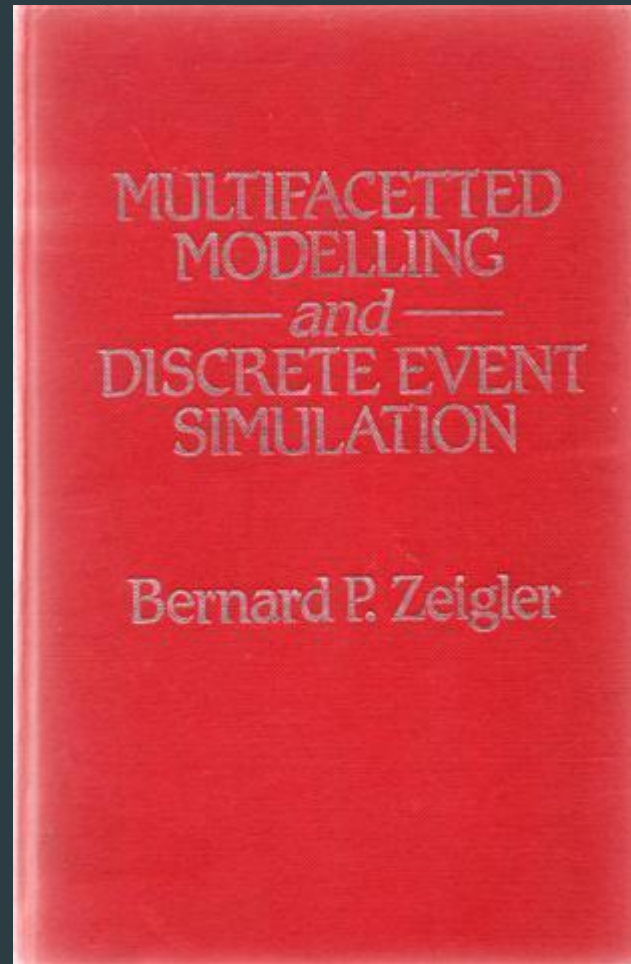


McGill

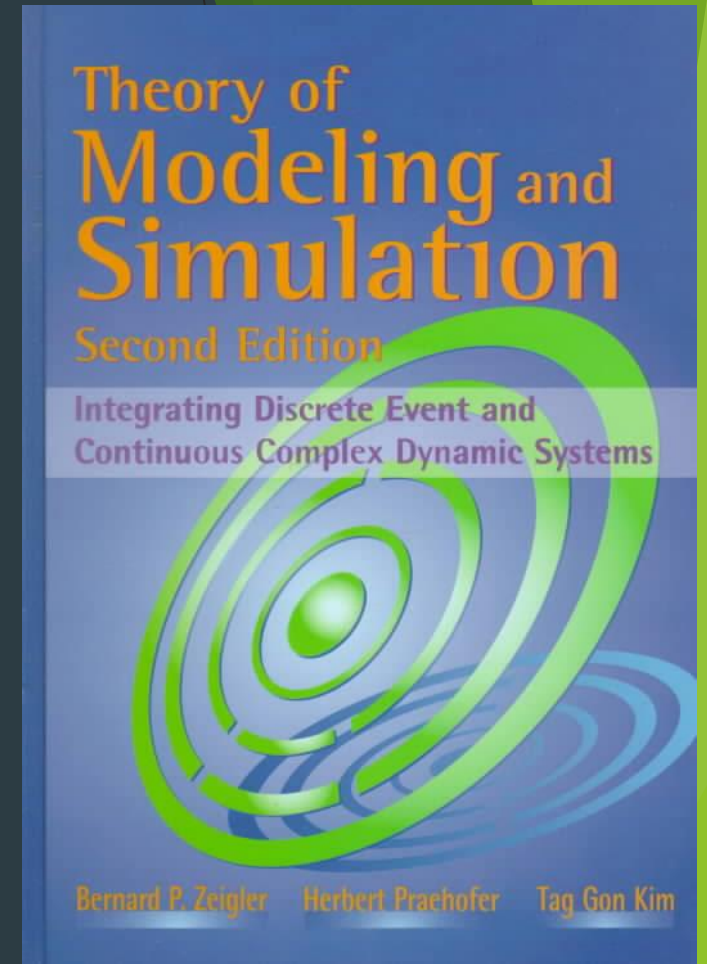
Introduction



Bernard P. Zeigler.
Theory Of Modelling And Simulation.
1st ed. Wiley, 1976.



Bernard P. Zeigler.
*Multifaceted Modelling and
Discrete Event Simulation.*
1st ed. Academic Press, 1984.



Bernard P. Zeigler, Herbert Praehofer,
and Tag Gon Kim.
Theory Of Modelling And Simulation.
2nd ed. Academic Press, 2000.

An overview of PythonPDEVS

Yentl Van Tendeloo¹

Hans Vangheluwe^{1,2}

¹ University of Antwerp, Belgium

² McGill University, Canada

Yentl.VanTendeloo@uantwerpen.be

Hans.Vangheluwe@uantwerpen.be

Yentl Van Tendeloo and Hans Vangheluwe.
An Overview of PythonPDEVS.
In Proceedings of Journées DEVS
Francophones (JDF), pages 59-66, 2016.

Methodology

An evaluation of DEVS simulation tools

Yentl Van Tendeloo^{1*} and Hans Vangheluwe^{1,2,3*}

Abstract

DEVS is a popular formalism for modeling complex dynamic systems using a discrete-event abstraction. Owing to its popularity, and the simplicity of the simulation kernel, a number of tools have been constructed by academia and industry. However, each of these tools has distinct design goals and a specific programming language implementation. Consequently, each supports a specific set of formalisms, combined with a specific set of features. Performance differs significantly between different tools. We provide an overview of the current state of eight different DEVS simulation tools: ADEVS, CD++, DEVS-Suite, MS4 Me, PowerDEVS, PythonPDEVS, VLE, and X-S-Y. We compare supported formalisms, compliance, features, and performance. This paper aims to help modelers in deciding which tools to use to solve their specific problems. It further aims to help tool builders, by showing the aspects of their tools that could be extended in future tool versions.

Simulation



Simulation: Transactions of the Society for
Modeling and Simulation International
1-19

© The Author(s) 2016

DOI: 10.1177/0037549716678330

sim.sagepub.com

SAGE

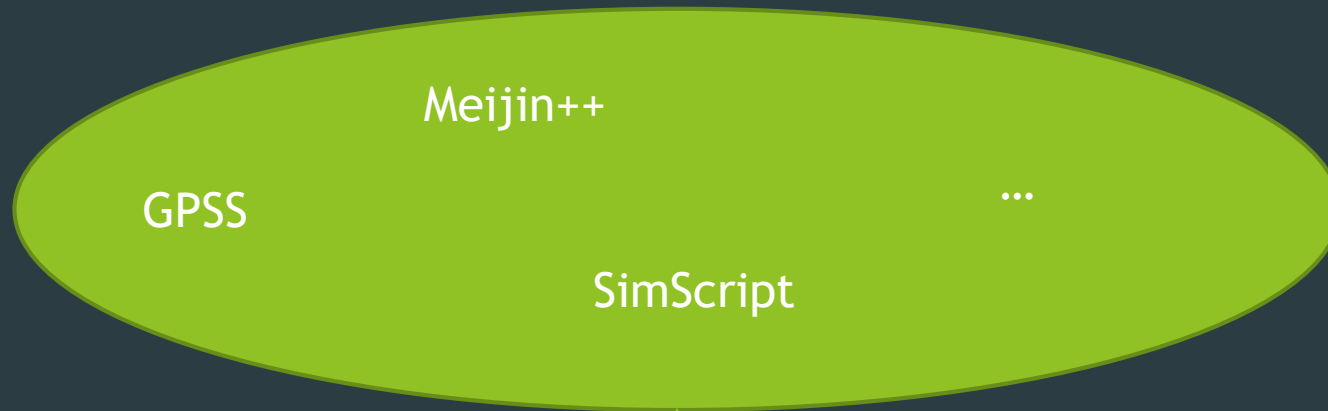


Yentl Van Tendeloo and Hans Vangheluwe.
An Evaluation of DEVS Simulation Tools.
Simulation: Transactions of the Society
for Modeling and Simulation International.
2017, 93(2): 103-121



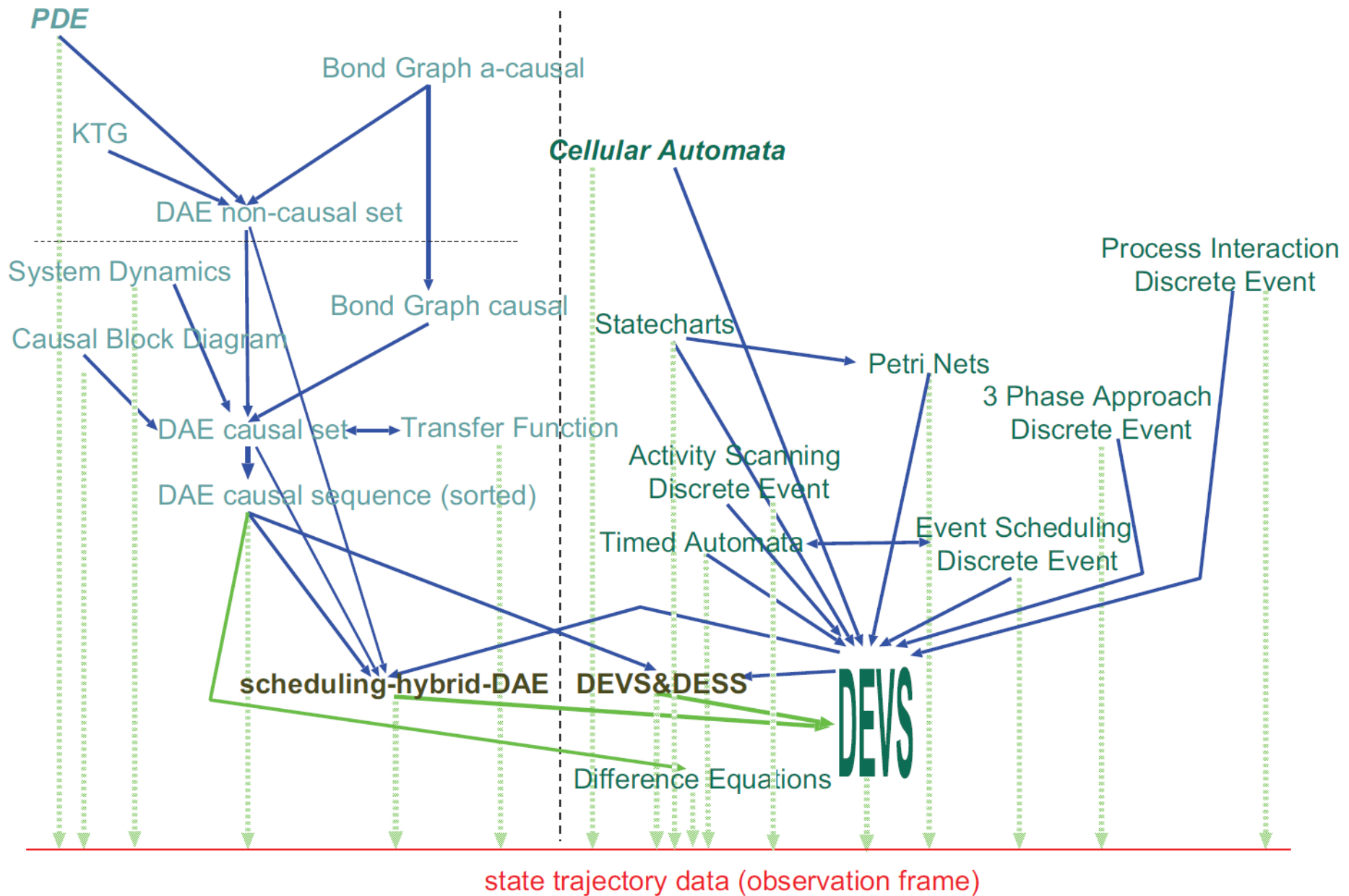
Our presentation uses initialized DEVS models, which contain an initial state. The initial state was left implicit in the original DEVS specification.

Sequential Discrete Event Language

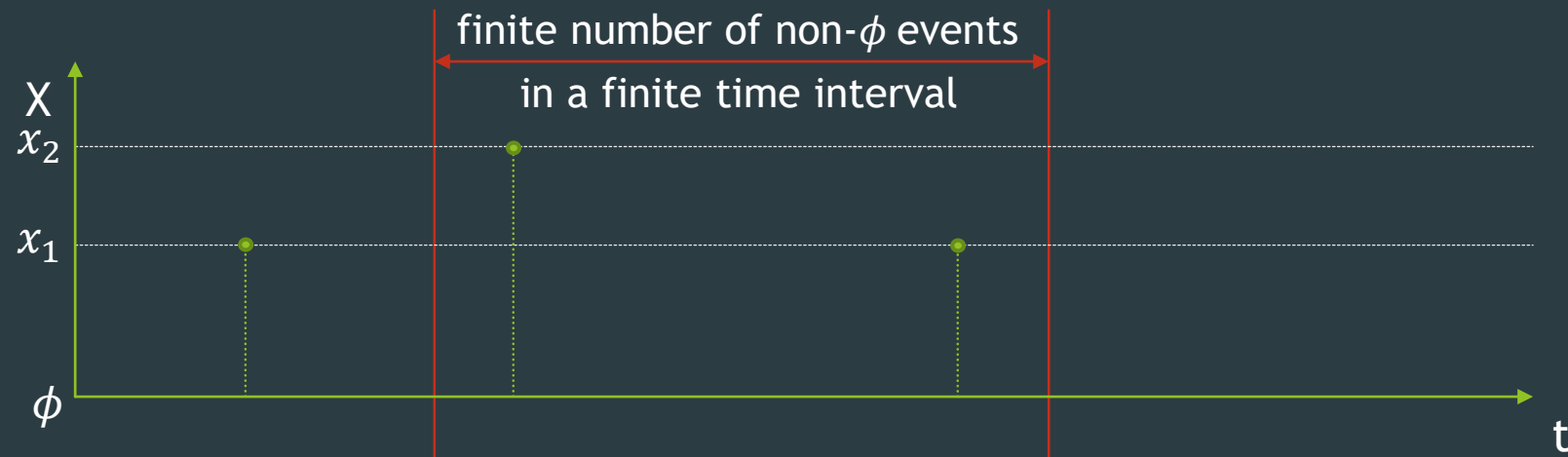


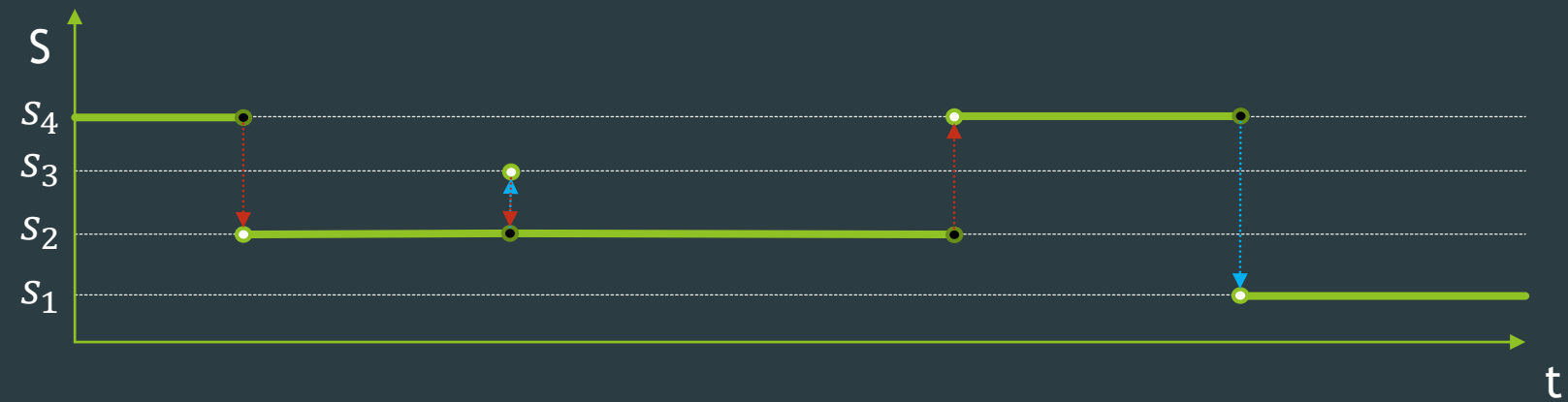
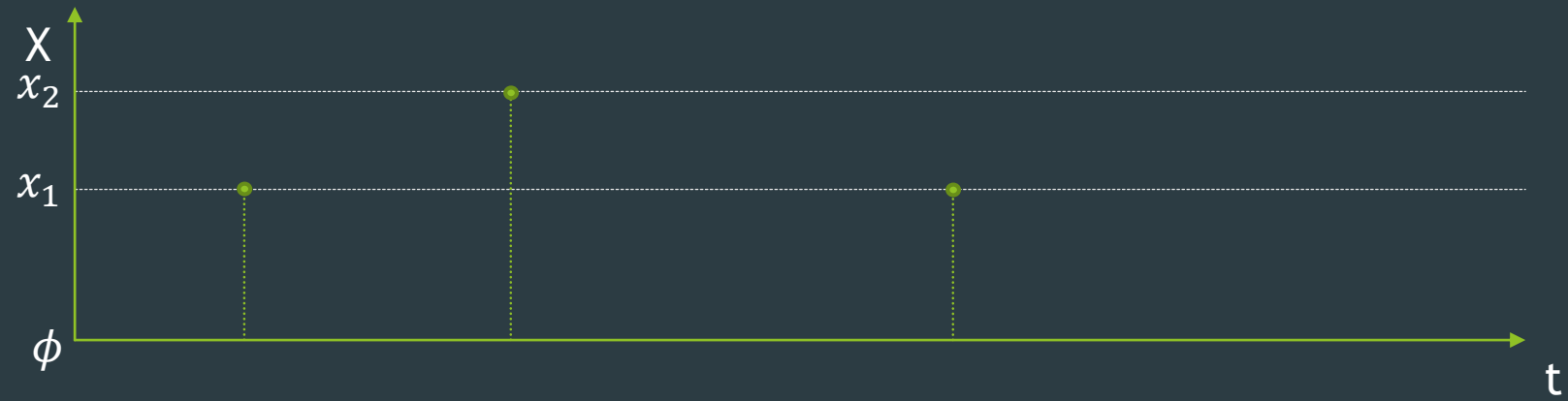
DEVS

= modular simulation assembly language



Vangheluwe, Hans. DEVS as a common denominator for multi-formalism hybrid systems modelling. In proceedings of the International Symposium on Computer-Aided Control System Design, pp. 129-134. 2000.

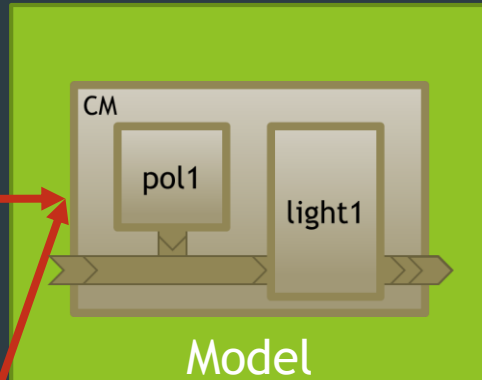
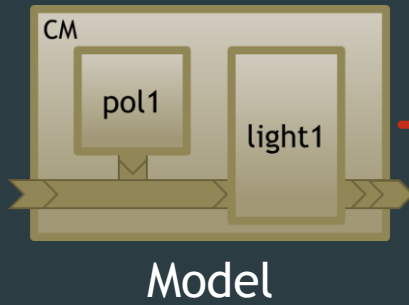






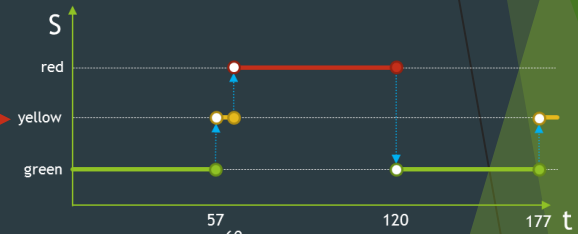
Experimentation

Simulation



Solver
Simulator

```
delayred = 60s  
delayyellow = 3s  
delaygreen = 57s  
qinit,light1 = (green, 0)  
qinit,pol1 = (idle, 280)  
condtermination = (tsim ≥ tend)  
tend = 24h
```



Concrete Syntax



simple_experiment.py

```
from pypdevs.simulator import Simulator

from mymodel import MyModel

model = MyModel(\
    q_init_pol1 = ("idle", 280),
    q_init_light1 = ("green", 0),
    delay_red = 60,
    delay_yellow = 3,
    delay_green = 57
)
simulator = Simulator(model)

simulator.setTerminationTime(24*60*60)
simulator.setClassicDEVS()
simulator.setVerbose()

simulator.simulate()
```

___ Current Time: 0.00 _____

INITIAL CONDITIONS in model <system.light>

Initial State: green

Next scheduled internal transition at time 57.00

INITIAL CONDITIONS in model <system.policeman>

Initial State: idle

Next scheduled internal transition at time 20.00

__ Current Time: 20.00 _____

EXTERNAL TRANSITION in model <system.light>

Input Port Configuration:

port <interrupt>:
toManual

New State: going_manual

Next scheduled internal transition at time 20.0

INTERNAL TRANSITION in model <system.policeman>

New State: working

Output Port Configuration:

port <output>:
go_to_work

Next scheduled internal transition at time 3620.00

__ Current Time: 20.00 _____

INTERNAL TRANSITION in model <system.light>

Output Port Configuration:

port <observer>:

turn_off

New State: manual

Next scheduled internal transition at time inf

__ Current Time: 3620.00 _____

EXTERNAL TRANSITION in model <system.light>

Input Port Configuration:

port <interrupt>:

toAuto

New State: going_auto

Next scheduled internal transition at time 3620.00

INTERNAL TRANSITION in model <system.policeman>

New State: idle

Output Port Configuration:

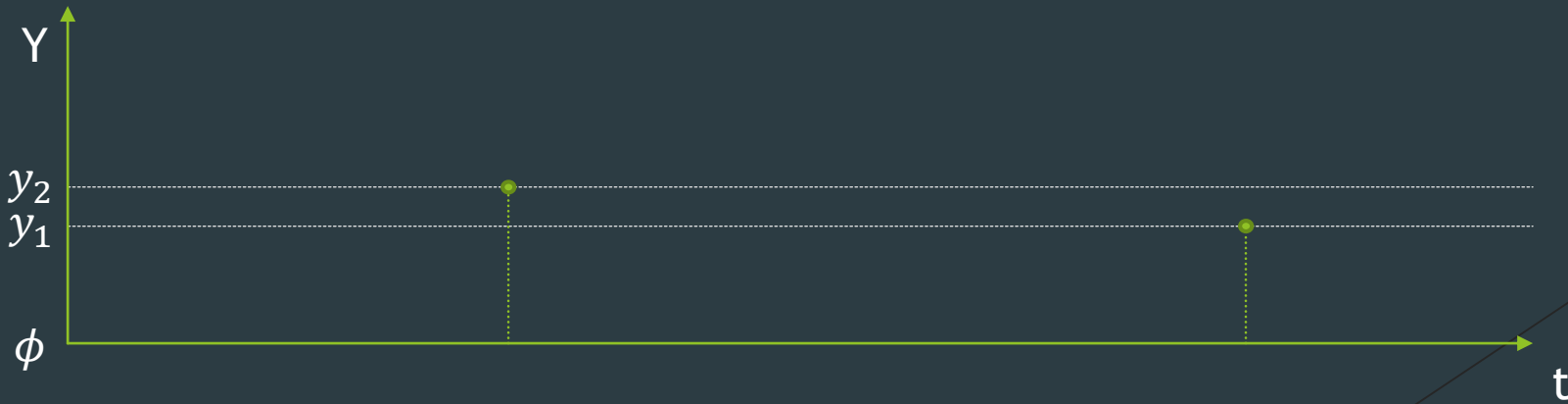
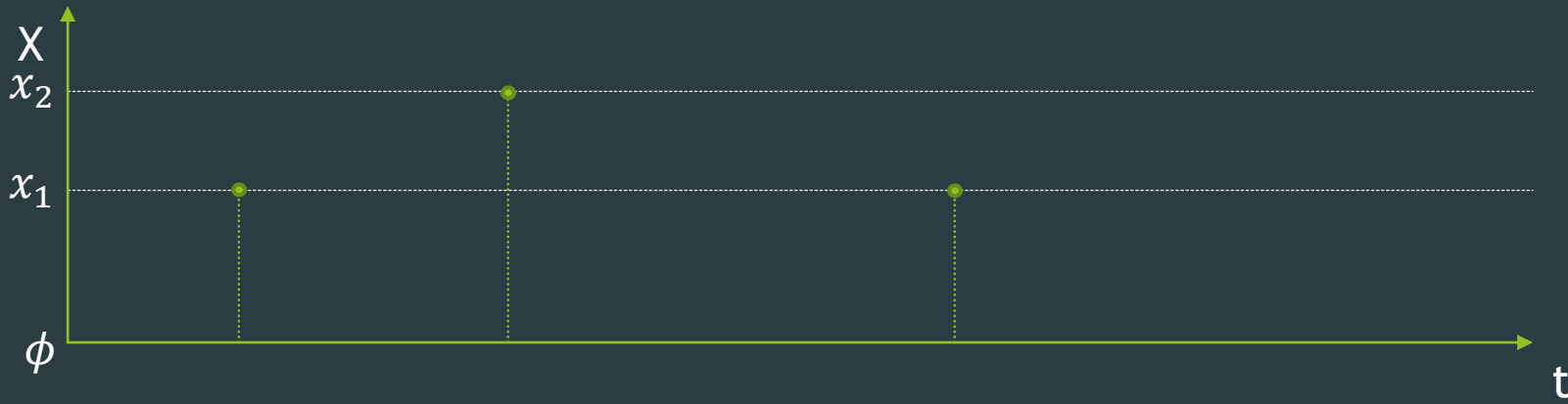
port <output>:

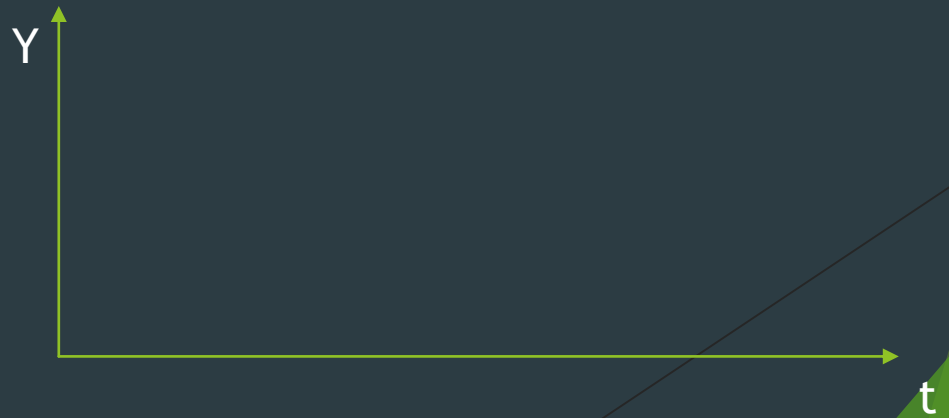
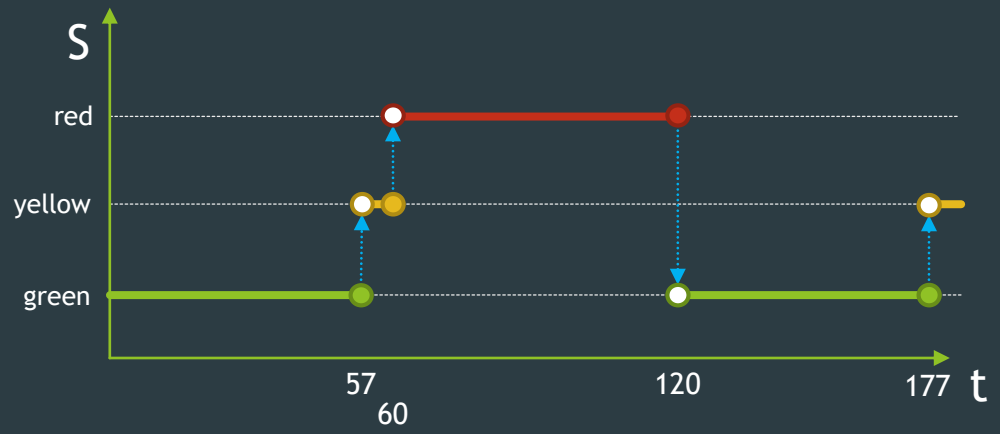
take_break

Next scheduled internal transition at time 3920.00

Atomic Models









Autonomous (no input)

$$M = \langle S, \delta_{int}, ta \rangle$$

S : set of sequential states

$$S = \{\text{red, yellow, green}\}$$

$$\delta_{int} : S \rightarrow S$$

$$\delta_{int} = \{\text{red} \rightarrow \text{green}, \\ \text{green} \rightarrow \text{yellow}, \\ \text{yellow} \rightarrow \text{red}\}$$

$$ta : S \rightarrow \mathbb{R}_{0,+\infty}^+$$

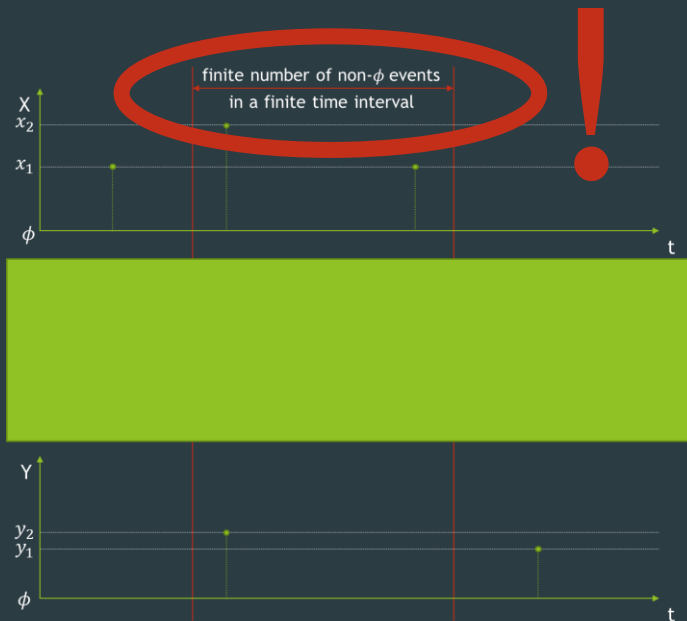
$$ta = \{\text{red} \rightarrow \text{delay}_{\text{red}}, \\ \text{green} \rightarrow \text{delay}_{\text{green}}, \\ \text{yellow} \rightarrow \text{delay}_{\text{yellow}}\}$$

Time Advance: corner cases

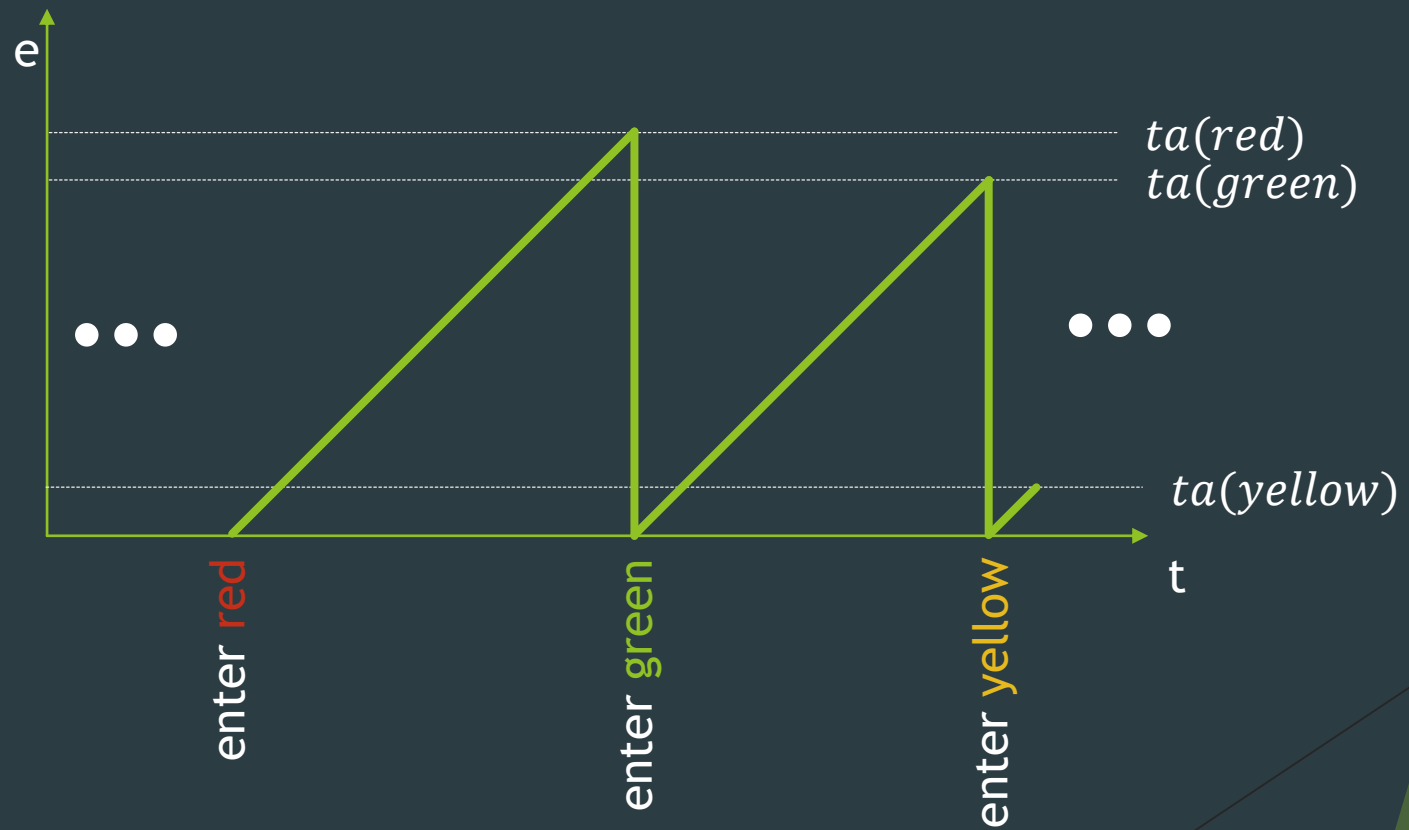
$$ta : S \rightarrow \mathbb{R}_{0,+\infty}^+$$

$ta(s_i) = 0$
transient states

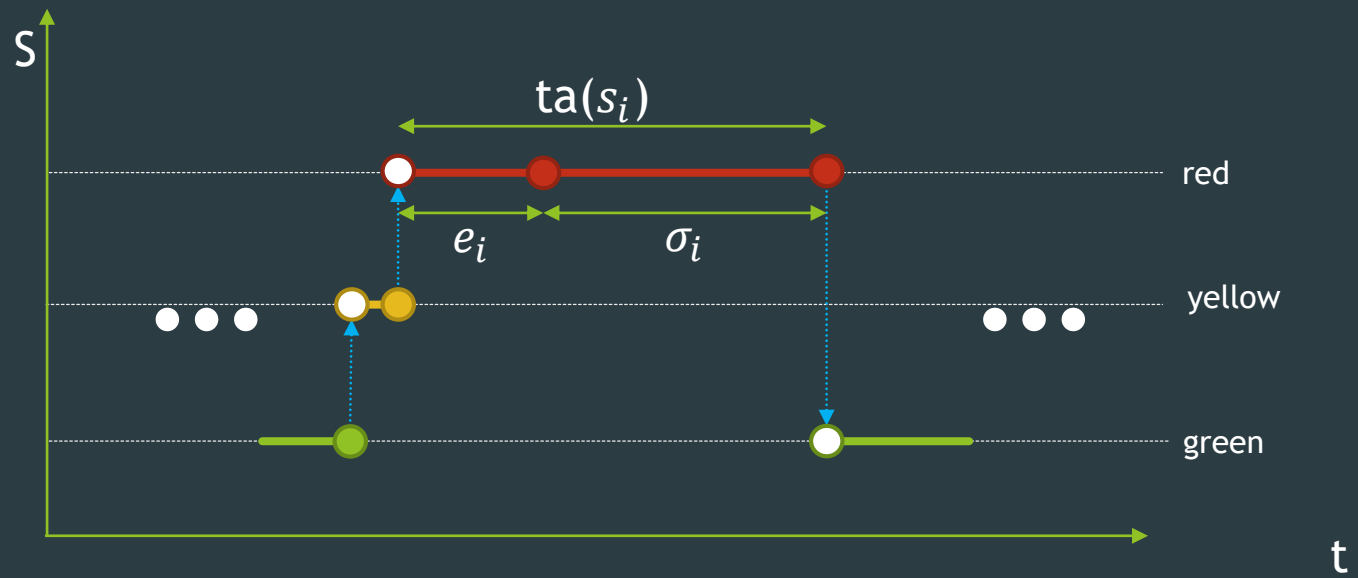
$ta(s_i) = +\infty$
passive states



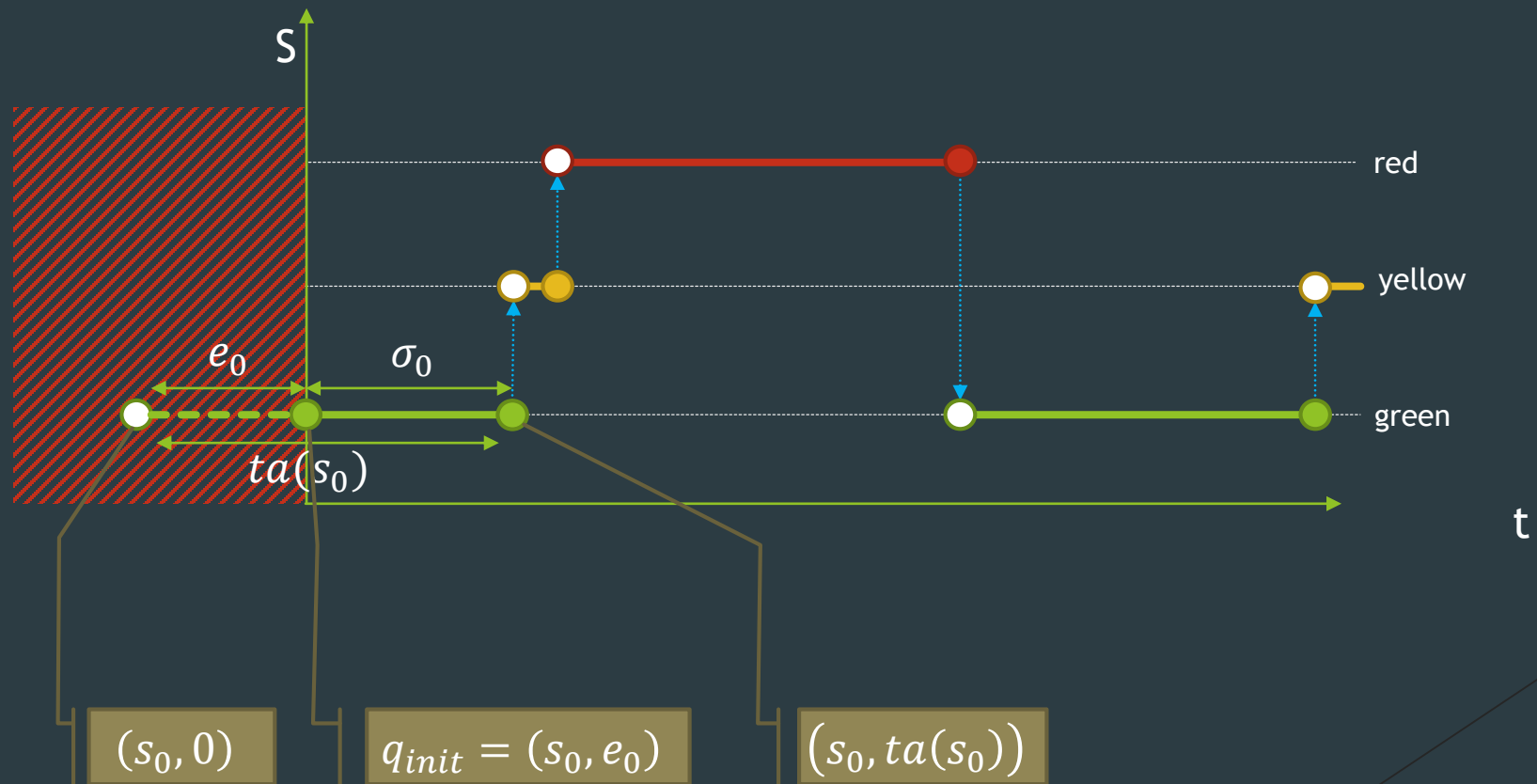
Elapsed time



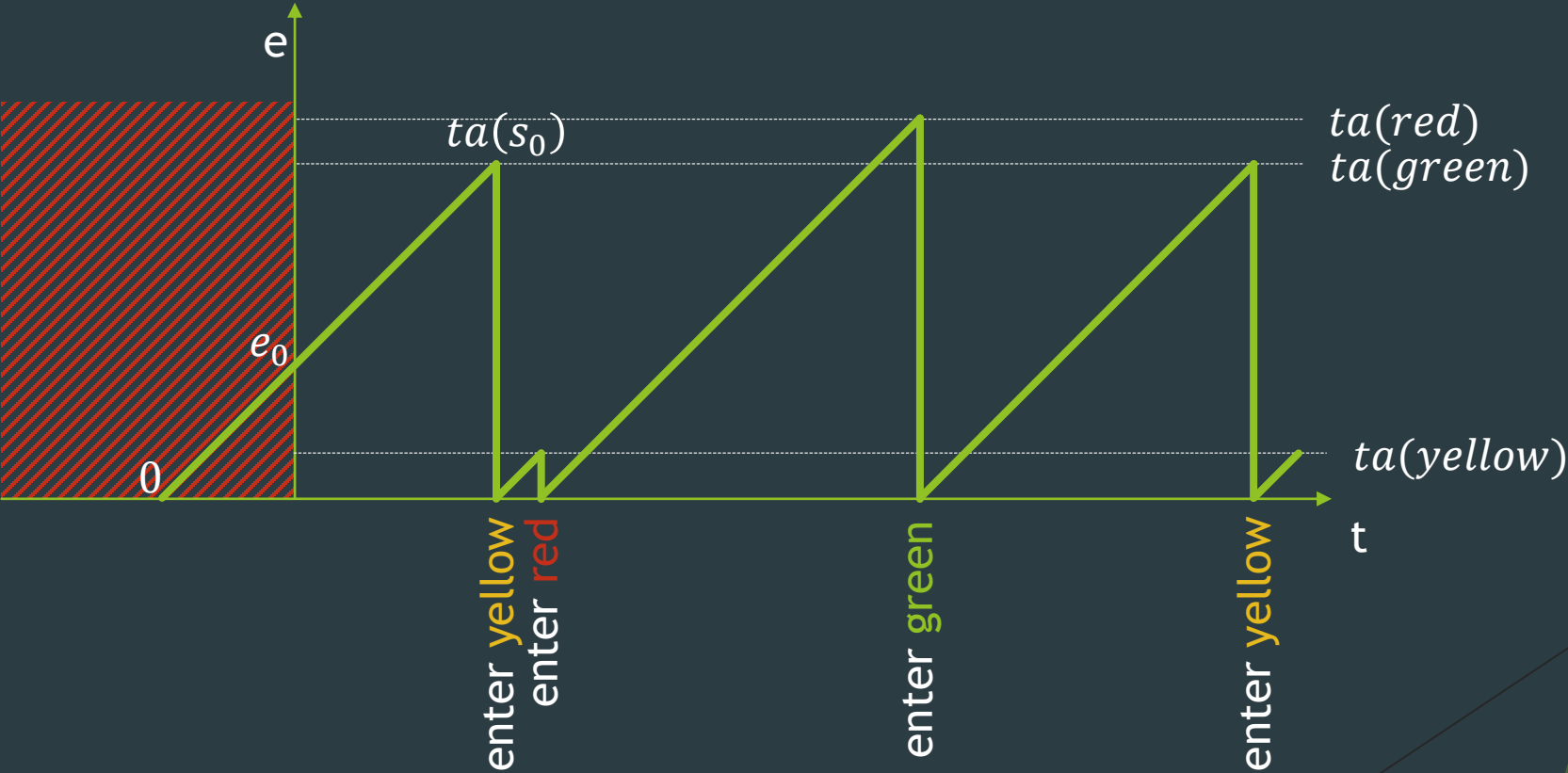
Elapsed time

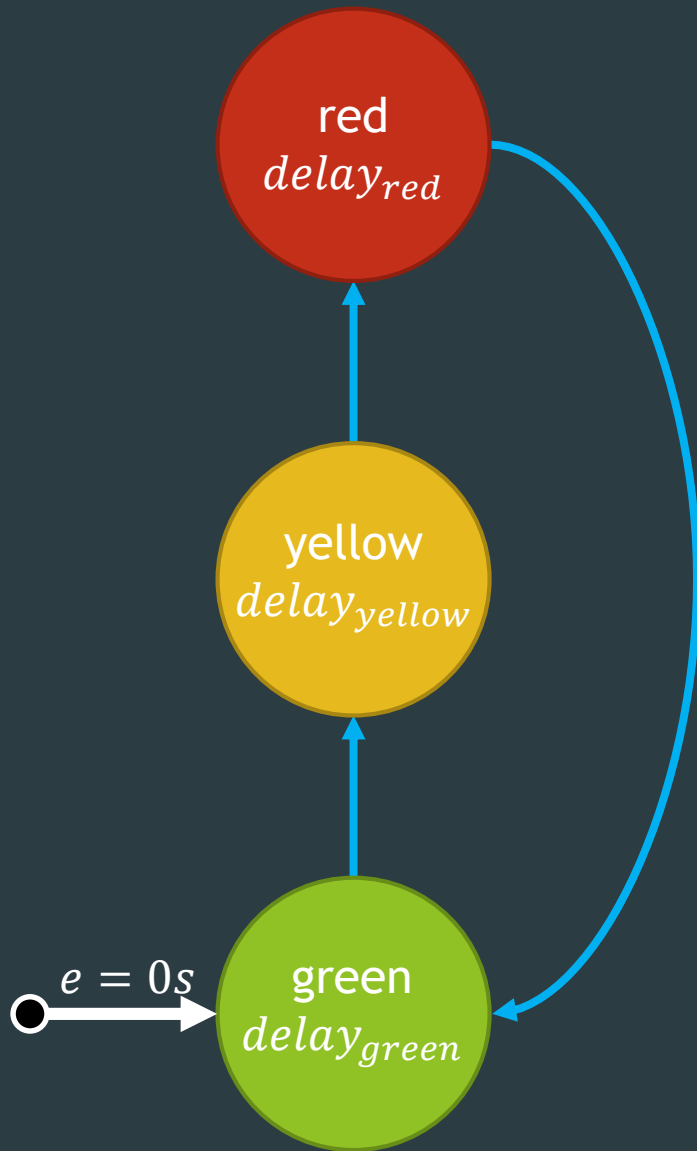


Initialization of Initial State



Elapsed time





Autonomous (no output)

$$M = \langle S, q_{init}, \delta_{int}, ta \rangle$$

S : set of sequential states

$$S = \{\text{red}, \text{yellow}, \text{green}\}$$

$$\delta_{int} : S \rightarrow S$$

$$\delta_{int} = \{\text{red} \rightarrow \text{green}, \\ \text{green} \rightarrow \text{yellow}, \\ \text{yellow} \rightarrow \text{red}\}$$

$$ta : S \rightarrow \mathbb{R}_{0,+\infty}^+$$

$$ta = \{\text{red} \rightarrow \text{delay}_{\text{red}}, \\ \text{green} \rightarrow \text{delay}_{\text{green}}, \\ \text{yellow} \rightarrow \text{delay}_{\text{yellow}}\}$$

q_{init} : Q - set of total states

$$Q = \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\}$$

$$q_{init} = (\text{green}, 0)$$

Abstract Syntax

```
S = {red, yellow, green}
 $\delta_{int} = \{ \text{red} \rightarrow \text{green},$ 
              $\text{green} \rightarrow \text{yellow},$ 
              $\text{yellow} \rightarrow \text{red} \}$ 
 $ta = \{ \text{red} \rightarrow \text{delay}_{red},$ 
         $\text{green} \rightarrow \text{delay}_{green},$ 
         $\text{yellow} \rightarrow \text{delay}_{yellow} \}$ 
 $q_{init} = (\text{green}, 0)$ 
```

Operational Semantics

```
time = 0
current_state = initial_state
last_time = -initial_elapsed
while not termination_condition():
    time = last_time + ta(current_state)
    current_state =  $\delta_{int}$ (current_state)
    last_time = time
```

Concrete Syntax

 atomic_int.py

```
from pypdevs.DEVS import *

class TrafficLightAutonomous(AtomicDEVS):
    def __init__(self, q_init, delay_green,
                 delay_yellow, delay_red):
        AtomicDEVS.__init__(self, "light")
        self.state, self.elapsed = q_init
        self.delay_green = delay_green
        self.delay_yellow = delay_yellow
        self.delay_red = delay_red

    def intTransition(self):
        state = self.state
        return {"red": "green",
              "yellow": "red",
              "green": "yellow"}[state]

    def timeAdvance(self):
        state = self.state
        return {"red": self.delay_red,
              "yellow": self.delay_yellow,
              "green": self.delay_green}[state]
```

___ Current Time: 0.00 _____

INITIAL CONDITIONS in model <light>

Initial State: green

Next scheduled internal transition at time 57.00

___ Current Time: 57.00 _____

INTERNAL TRANSITION in model <light>

New State: yellow

Output Port Configuration:

Next scheduled internal transition at time 60.00

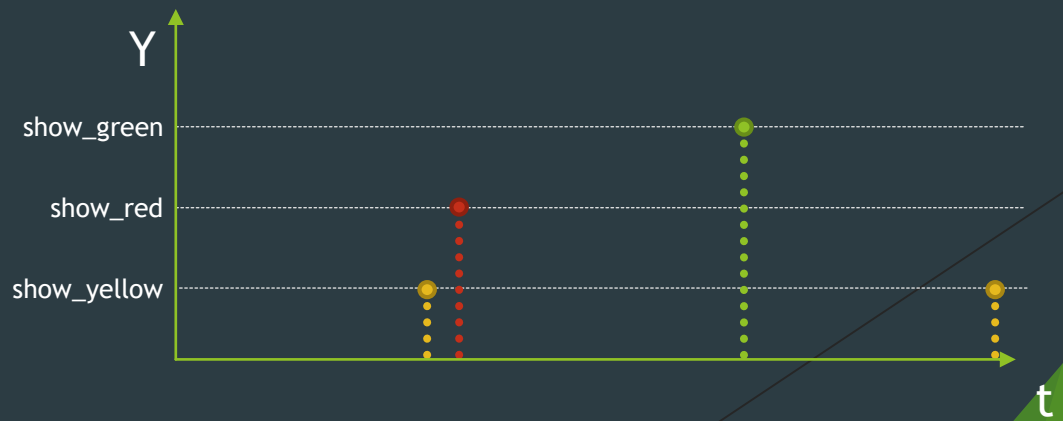
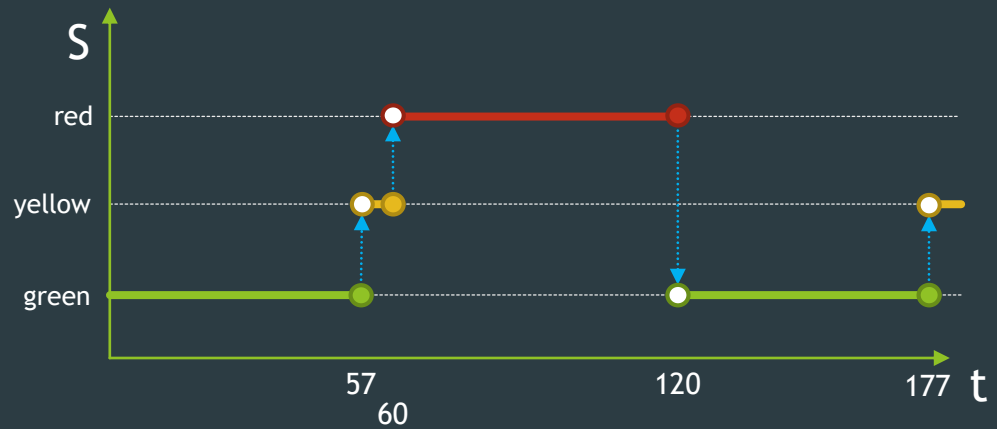
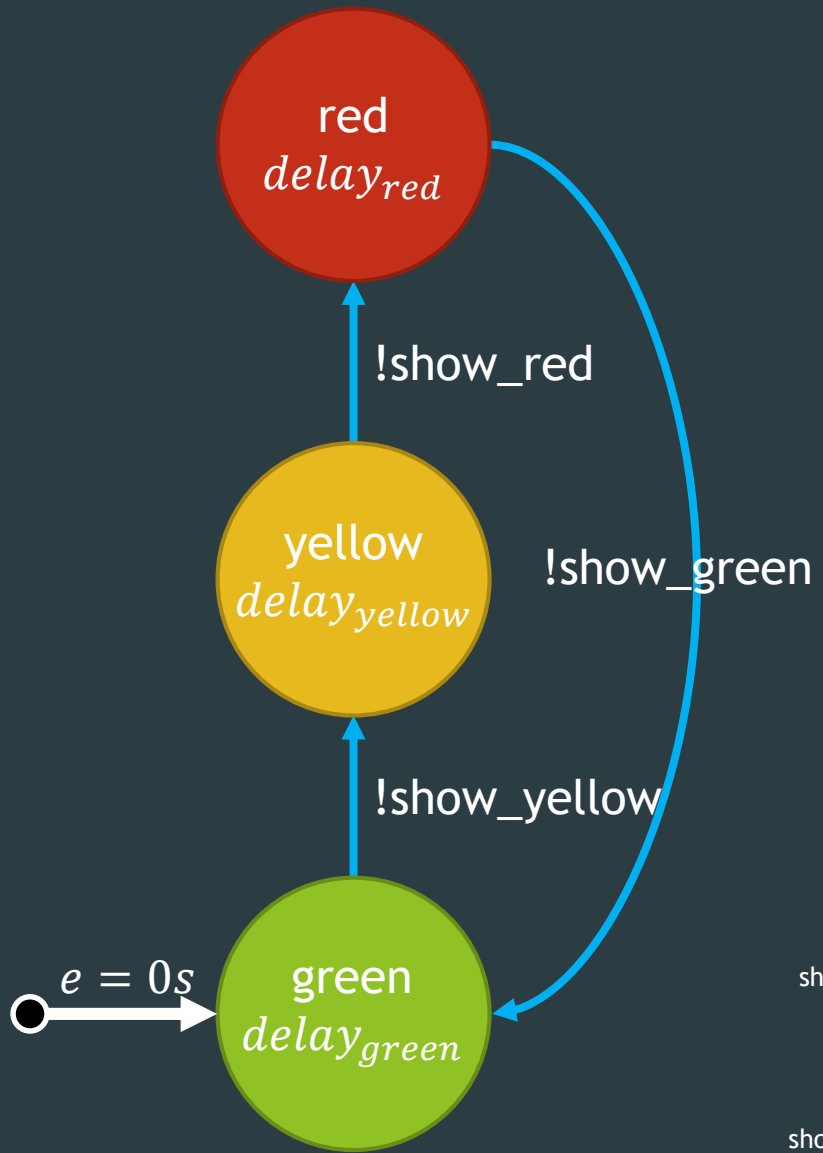
___ Current Time: 60.00 _____

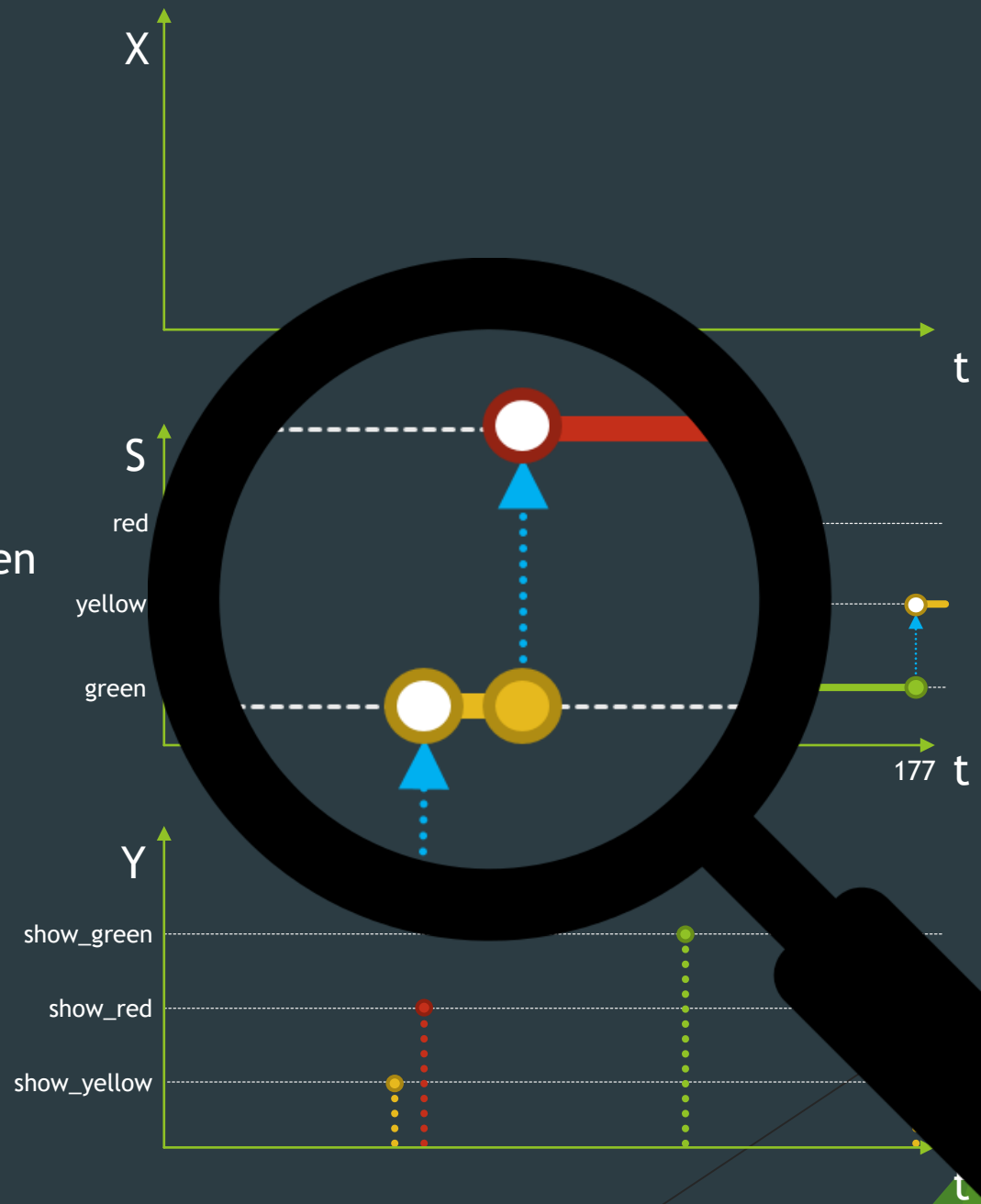
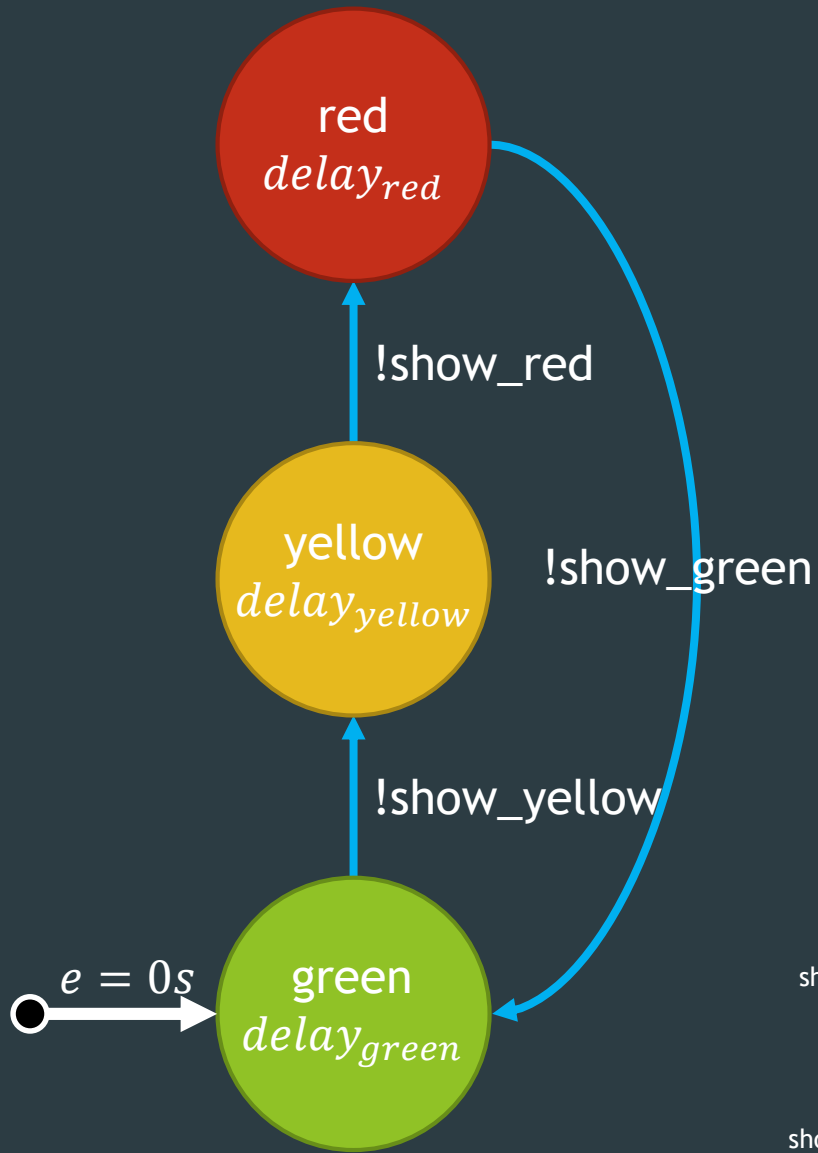
INTERNAL TRANSITION in model <light>

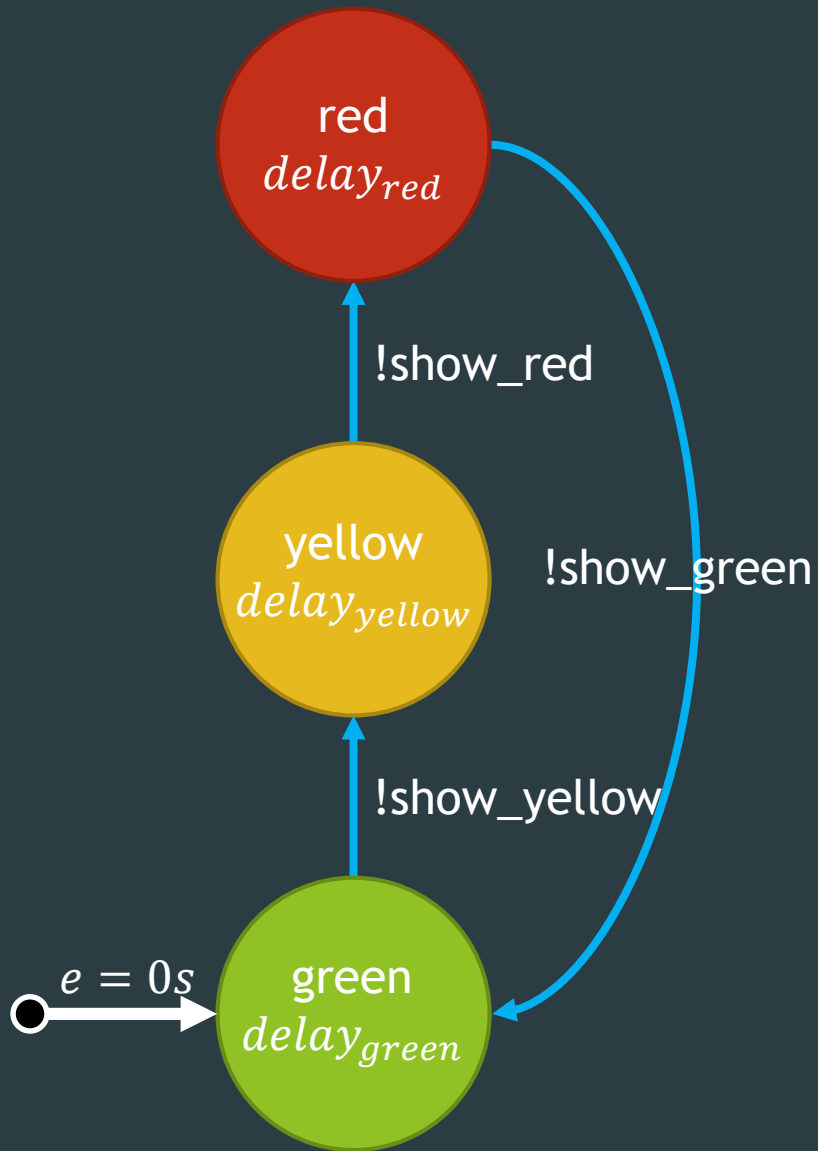
New State: red

Output Port Configuration:

Next scheduled internal transition at time 120.00







Autonomous (with output)

$$M = \langle Y, S, q_{init}, \delta_{int}, \lambda, ta \rangle$$

$$S = \{\text{red, yellow, green}\}$$

$$\delta_{int} = \{ \text{red} \rightarrow \text{green}, \\ \text{green} \rightarrow \text{yellow}, \\ \text{yellow} \rightarrow \text{red} \}$$

$$q_{init} = (\text{green}, 0)$$

$$ta = \{ \text{red} \rightarrow \text{delay}_{red}, \\ \text{green} \rightarrow \text{delay}_{green}, \\ \text{yellow} \rightarrow \text{delay}_{yellow} \}$$

Y : set of output events

$$Y = \{ \text{"show_red"}, \text{"show_green"}, \text{"show_yellow"} \}$$

$$\lambda : S \rightarrow Y \cup \{ \phi \}$$

$$\lambda = \{ \text{green} \rightarrow \text{"show_yellow"}, \\ \text{yellow} \rightarrow \text{"show_red"}, \\ \text{red} \rightarrow \text{"show_green"} \}$$

Abstract Syntax

```
S = {red, yellow, green}
qinit = (green, 0)
δint = { red → green,
         green → yellow,
         yellow → red}
ta = {red → delayred,
      green → delaygreen,
      yellow → delayyellow}
Y = {"show_red",
     "show_green",
     "show_yellow"}
λ = {green → "show_yellow",
     yellow → "show_red",
     red → "show_green"}
```

Operational Semantics

```
time = 0
current_state = initial_state
last_time = -initial_elapsed
while not termination_condition():
    time = last_time + ta(current_state)
    output(λ(current_state))
    current_state = δint(current_state)
    last_time = time
```

Concrete Syntax

 atomic_out.py

```
from pypdevs.DEVS import *

class TrafficLightWithOutput(AtomicDEVS):
    def __init__(self, ...):
        AtomicDEVS.__init__(self, "light")
        self.observe = self.addOutPort("observer")
        ...
    ...

    def outputFnc(self):
        state = self.state
        if state == "red":
            return {self.observe: "show_green"}
        elif state == "yellow":
            return {self.observe: "show_red"}
        elif state == "green":
            return {self.observe: "show_yellow"}
```

__ Current Time: 0.00 _____

INITIAL CONDITIONS in model <light>

Initial State: green

Next scheduled internal transition at time 57.00

__ Current Time: 57.00 _____

INTERNAL TRANSITION in model <light>

New State: yellow

Output Port Configuration:

port <observer>:

show_yellow

Next scheduled internal transition at time 60.00

__ Current Time: 60.00 _____

INTERNAL TRANSITION in model <light>

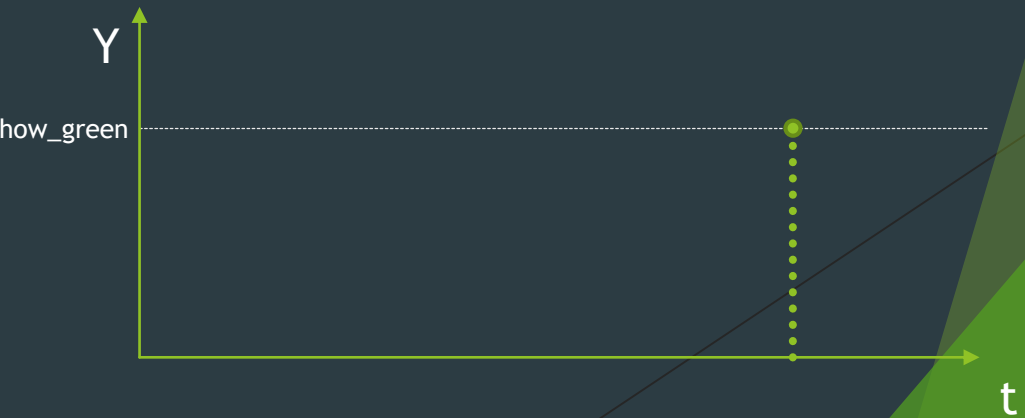
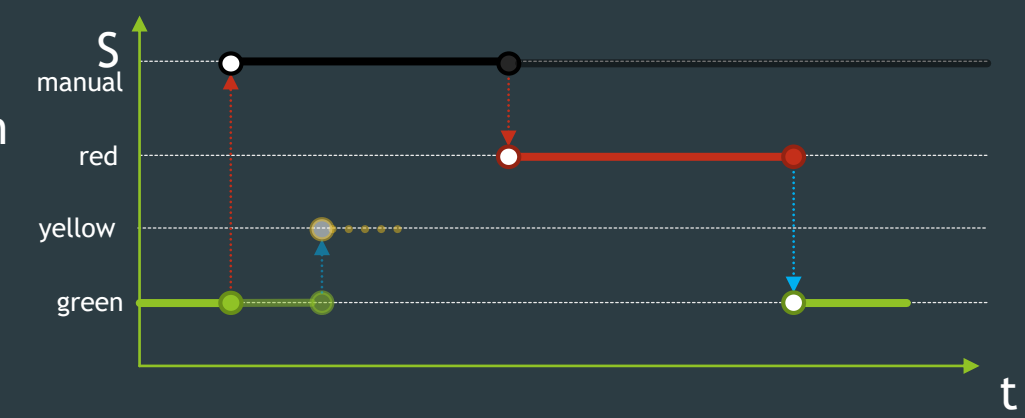
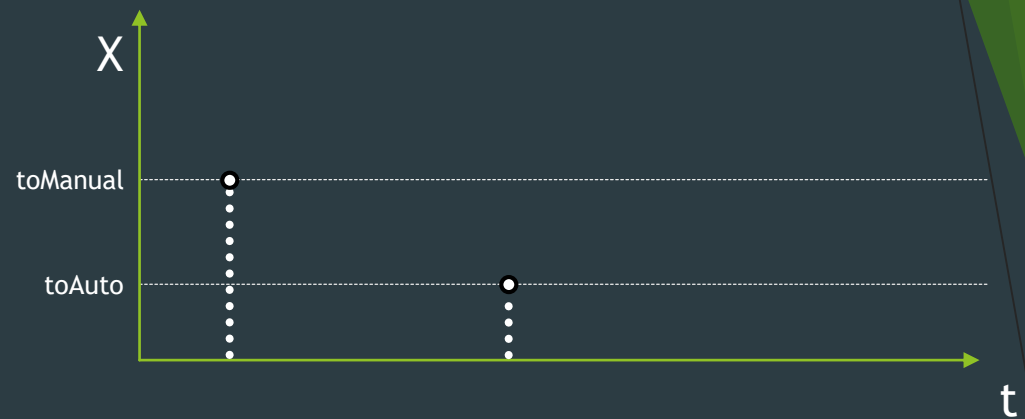
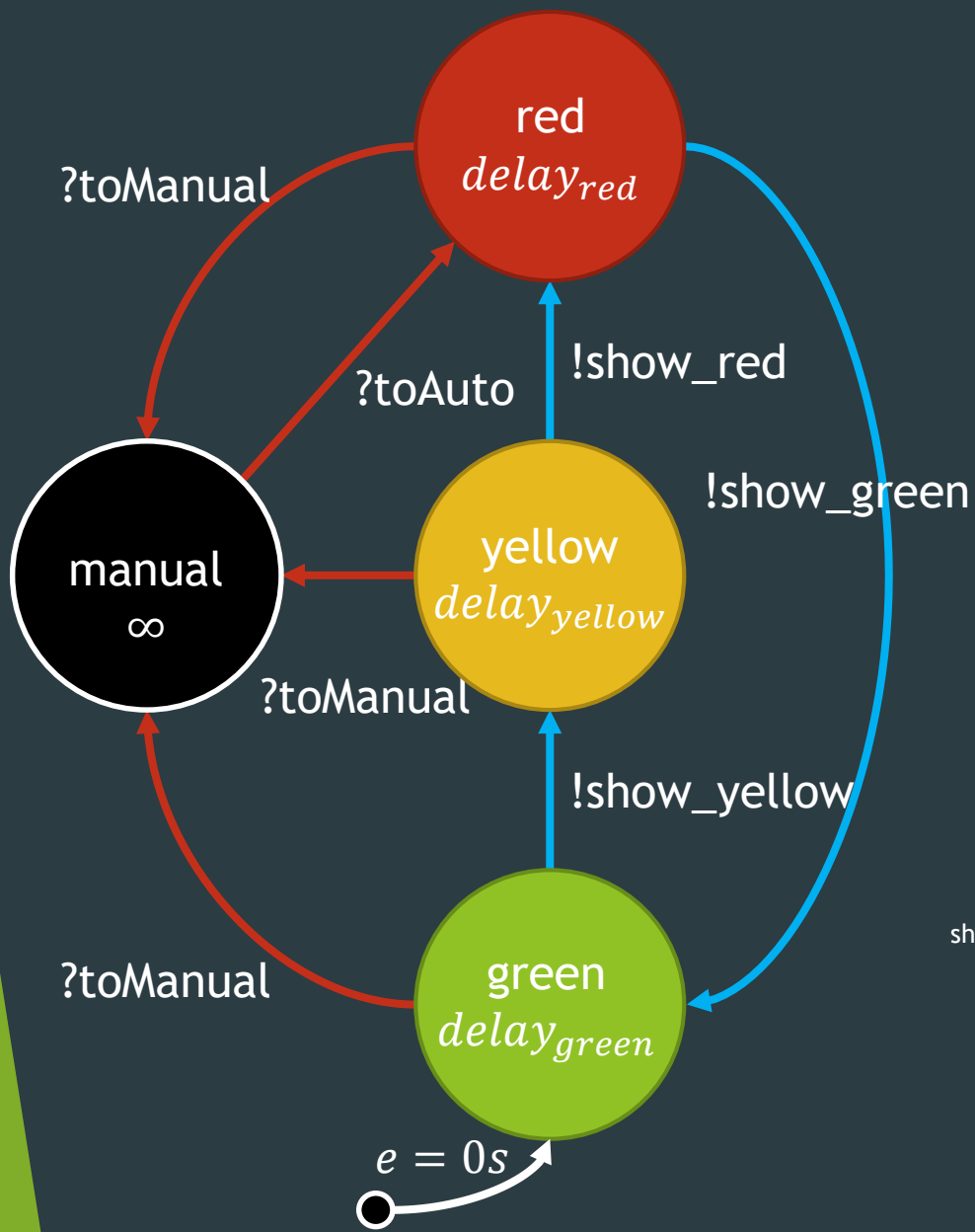
New State: red

Output Port Configuration:

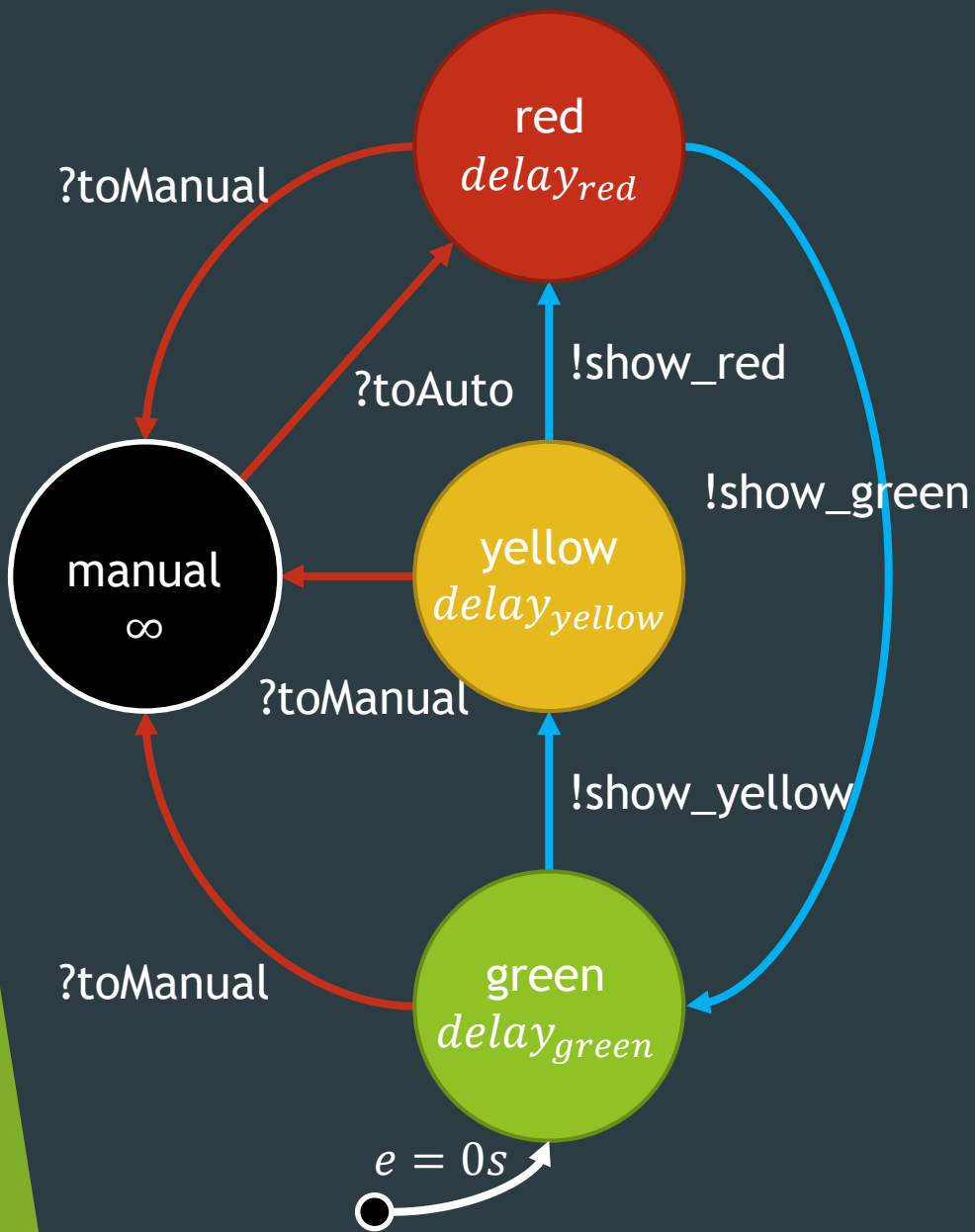
port <observer>:

show_red

Next scheduled internal transition at time 120.00



Reactive



$$M = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

$$Y = \{\text{"show_red"}, \text{"show_green"}, \text{"show_yellow"}\}$$

$$S = \{\text{red}, \text{yellow}, \text{green}, \text{manual}\}$$

$$q_{init} = (\text{green}, 0)$$

$$\delta_{int} = \{\text{red} \rightarrow \text{green}, \\ \text{green} \rightarrow \text{yellow}, \\ \text{yellow} \rightarrow \text{red}\}$$

$$\lambda = \{\text{green} \rightarrow \text{"show_yellow"}, \\ \text{yellow} \rightarrow \text{"show_red"}, \\ \text{red} \rightarrow \text{"show_green"}\}$$

$$ta = \{\text{red} \rightarrow \text{delay}_{red}, \\ \text{green} \rightarrow \text{delay}_{green}, \\ \text{yellow} \rightarrow \text{delay}_{yellow}, \\ \text{manual} \rightarrow +\infty\}$$

X : set of input events

$$X = \{\text{"toAuto"}, \text{"toManual"}\}$$

$$\delta_{ext} : Q \times X \rightarrow S$$

$$Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$$

$$\delta_{ext} = \{((*, *), \text{"toManual"}) \rightarrow \text{"manual"}, \\ ((\text{"manual"}, *), \text{"toAuto"}) \rightarrow \text{"red"}\}$$

Abstract Syntax

```
Y = {"show_red", "show_green", "show_yellow"}
S = {red, yellow, green, manual}
qinit = (green, 0)
δint = {red → green,
        green → yellow,
        yellow → red}
λ = {green → "show_yellow",
     yellow → "show_red",
     red → "show_green"}
ta = {red → delayred,
     green → delaygreen,
     yellow → delayyellow,
     manual → ∞}
X = {"toAuto", "toManual"}
δext = {( (*, *), "toManual") → manual,
         ( (manual, *), "toAuto") → red}
```

Operational Semantics

```
time = 0
current_state = initial_state
last_time = -initial_elapsed
while not termination_condition():
    next_time = last_time + ta(current_state)
    if time_next_ev <= next_time:
        e = time_next_ev - last_time
        time = time_next_ev
        current_state = δext((current_state, e), next_ev)
    else:
        time = next_time
        output(λ(current_state))
        current_state = δint(current_state)
        last_time = time
```

Abstract Syntax

```
 $Y = \{\text{"show\_red"}, \text{"show\_green"}, \text{"show\_yellow"}\}$   
 $S = \{\text{red}, \text{yellow}, \text{green}, \text{manual}\}$   
 $q_{init} = (\text{green}, 0)$   
 $\delta_{int} = \{\text{red} \rightarrow \text{green},$   
           $\text{green} \rightarrow \text{yellow},$   
           $\text{yellow} \rightarrow \text{red}\}$   
 $\lambda = \{\text{green} \rightarrow \text{"show\_yellow"},$   
         $\text{yellow} \rightarrow \text{"show\_red"},$   
         $\text{red} \rightarrow \text{"show\_green"}\}$   
 $ta = \{\text{red} \rightarrow \text{delay}_{red},$   
        $\text{green} \rightarrow \text{delay}_{green},$   
        $\text{yellow} \rightarrow \text{delay}_{yellow},$   
        $\text{manual} \rightarrow \infty\}$   
 $X = \{\text{"toAuto"}, \text{"toManual"}\}$   
 $\delta_{ext} = \{(\text{*, *}, \text{"toManual"}) \rightarrow \text{manual},$   
           $(\text{manual}, \text{*}), \text{"toAuto"} \rightarrow \text{red}\}$ 
```

Concrete Syntax

 atomic_ext.py

```
from pypdevs.DEVS import *  
  
class TrafficLight(AtomicDEVS):  
    def __init__(self, ...):  
        AtomicDEVS.__init__(self, "light")  
        self.interrupt = self.addInPort("interrupt")  
        ...  
    ...  
  
    def extTransition(self, inputs):  
        inp = inputs[self.interrupt]  
        if inp == "toManual":  
            return "manual"  
        elif inp == "toAuto":  
            if self.state == "manual":  
                return "red"
```

__ Current Time: 0.00 _____

INITIAL CONDITIONS in model <light>

Initial State: green

Next scheduled internal transition at time 57.00

__ Current Time: 57.00 _____

INTERNAL TRANSITION in model <light>

New State: yellow

Output Port Configuration:

port <observer>:

show_yellow

Next scheduled internal transition at time 60.00

__ Current Time: 60.00 _____

INTERNAL TRANSITION in model <light>

New State: red

Output Port Configuration:

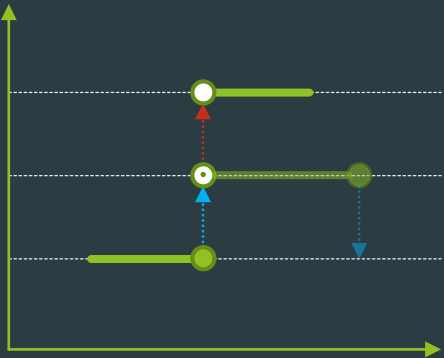
port <observer>:

show_red

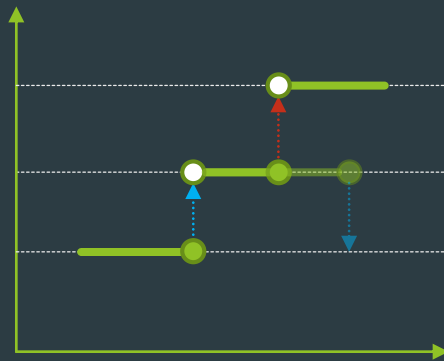
Next scheduled internal transition at time 120.00

$$Q = \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\}$$

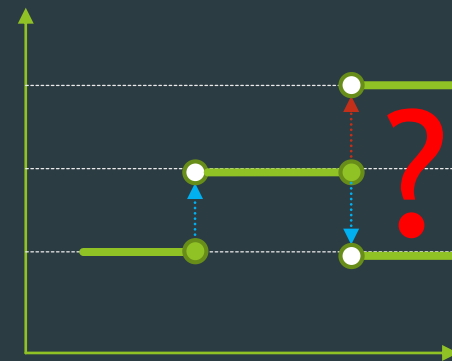
$e = 0$

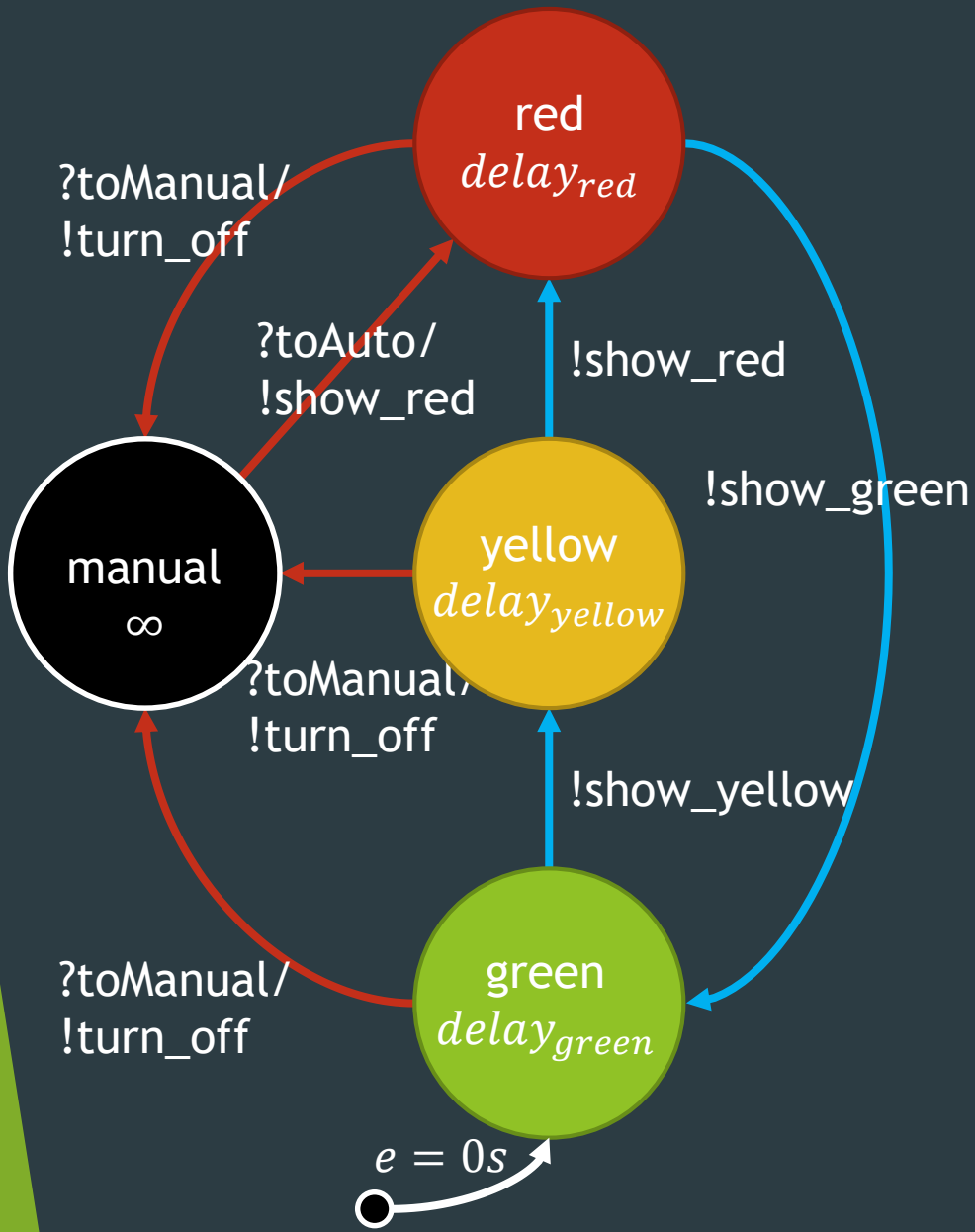


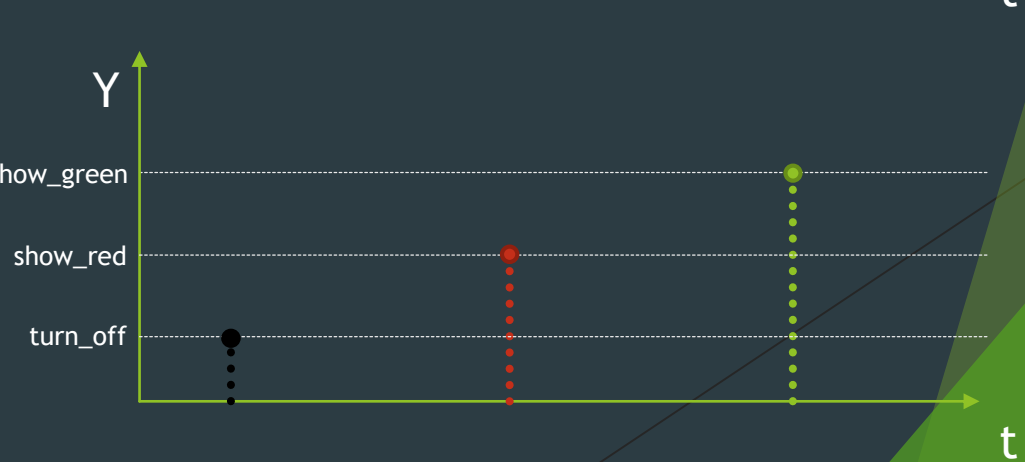
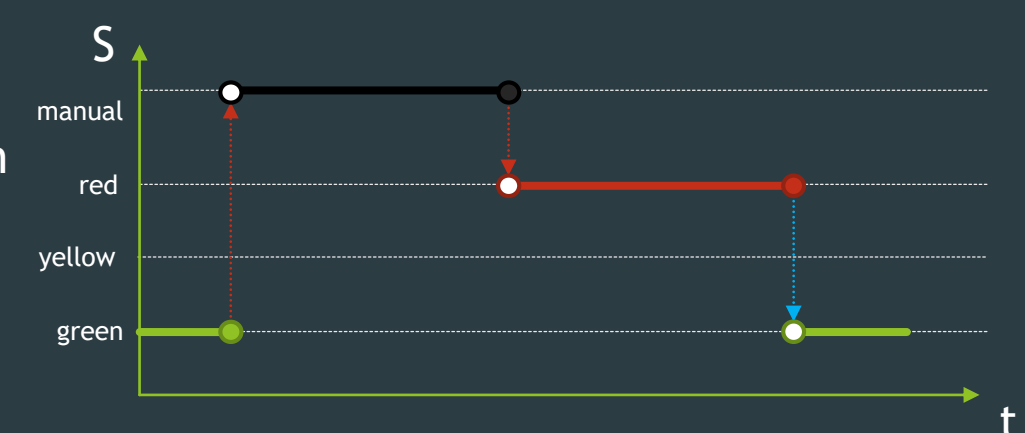
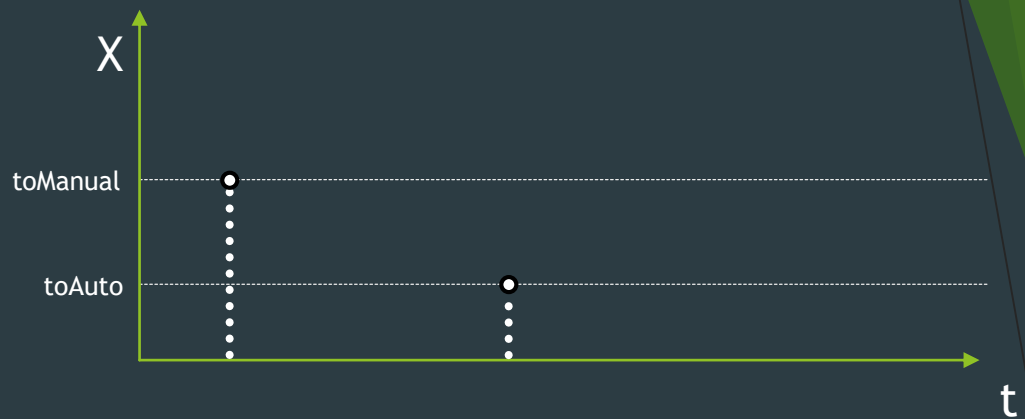
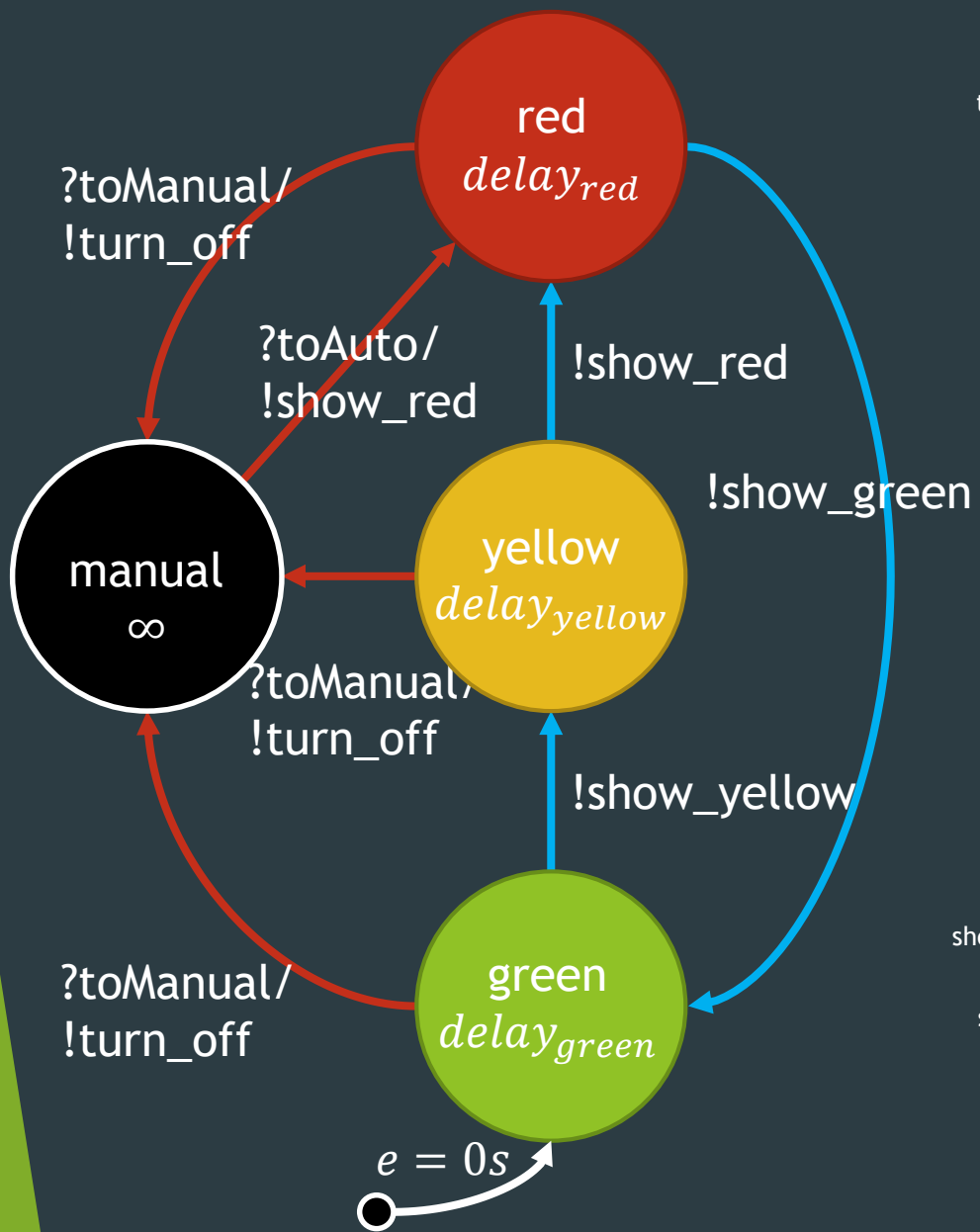
$0 < e < ta(s)$

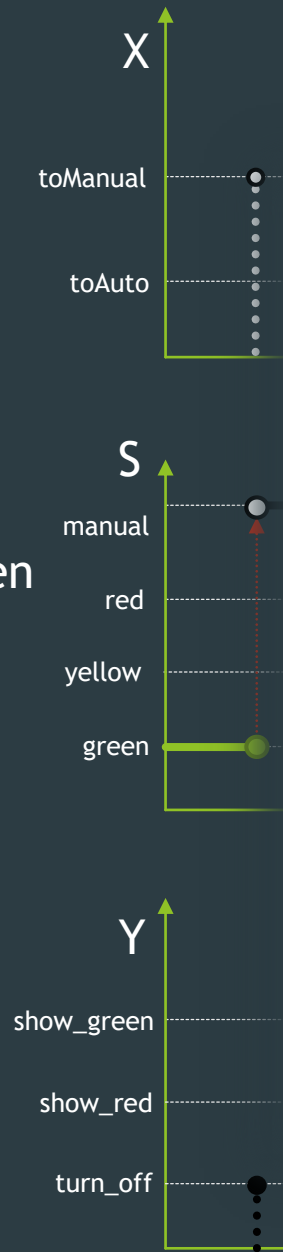
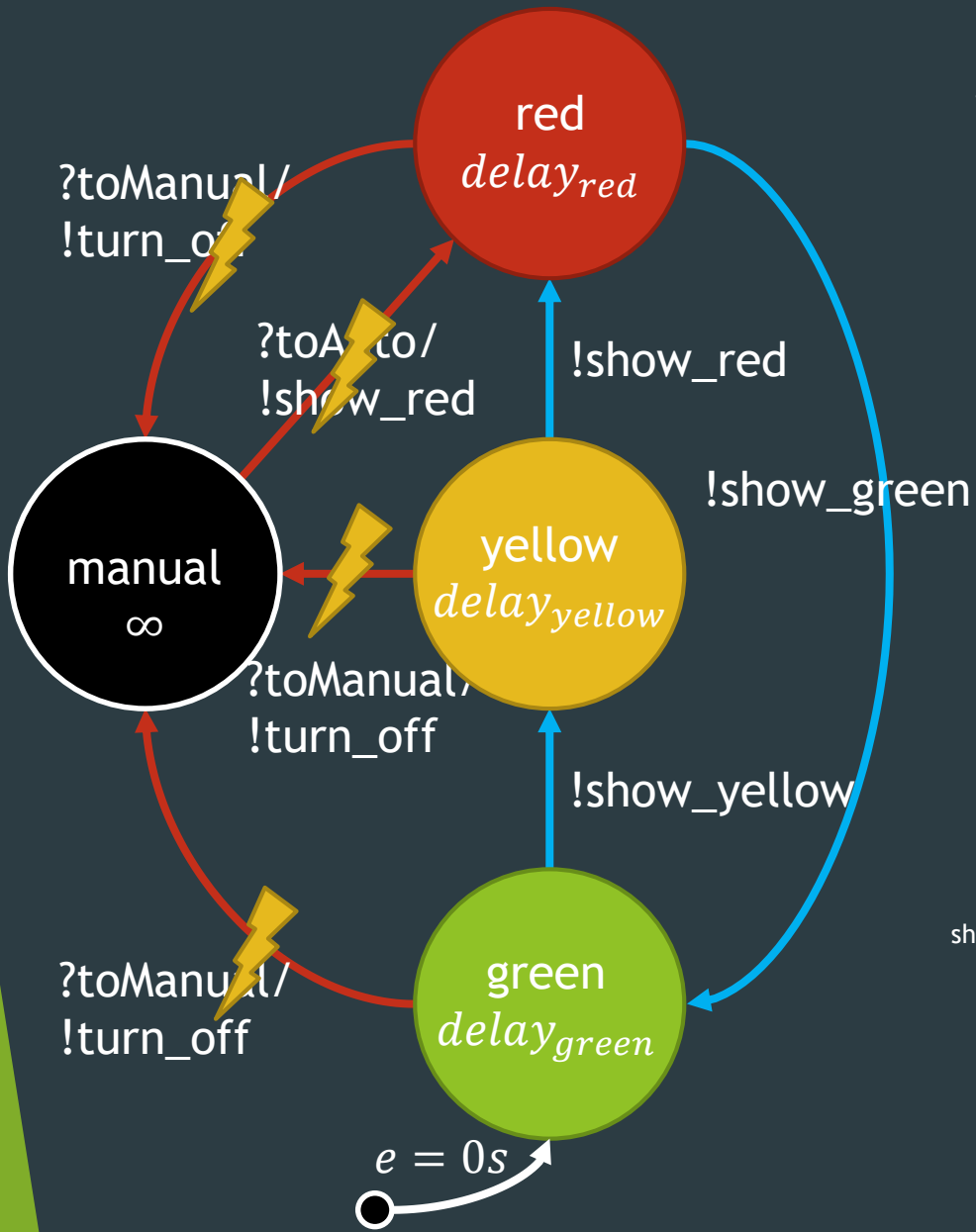


$e = ta(s)$





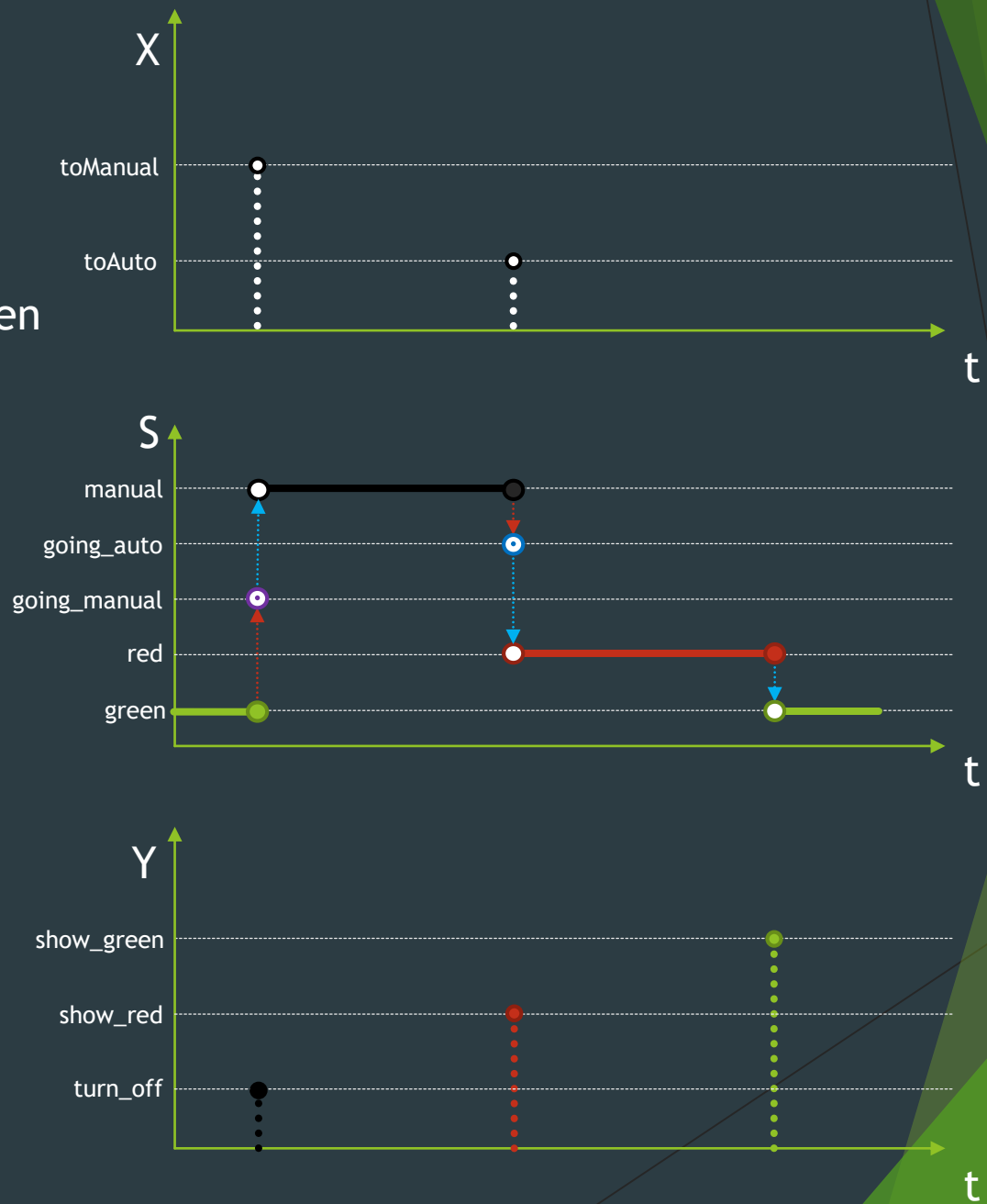
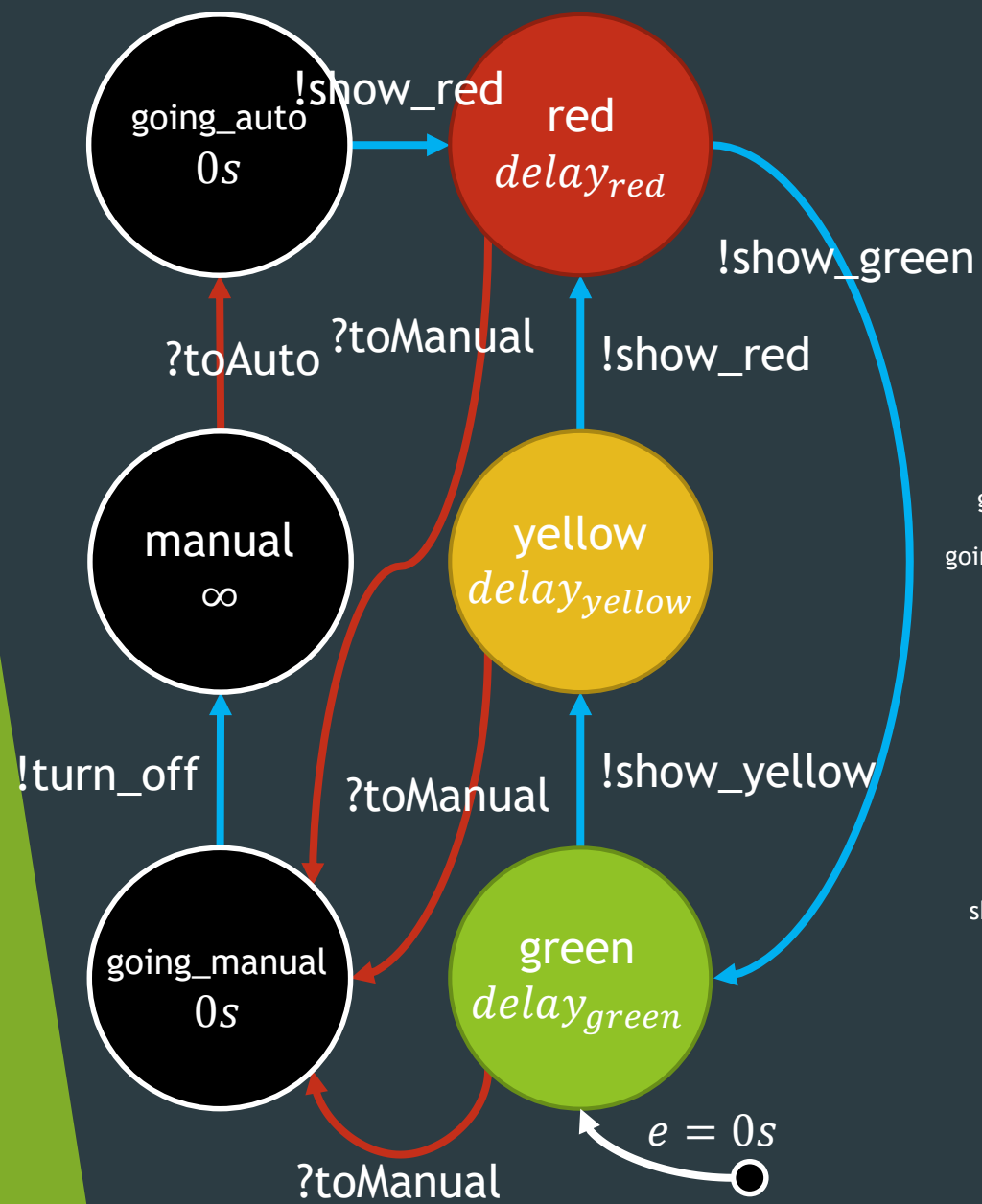




```

time = 0
current_state = initial_state
last_time = -initial_elapsed
while not termination_condition():
    next_time = last_time + ta(current_state)
    if time_next_ev <= next_time:
        e = time_next_ev - last_time
        time = time_next_ev
        current_state =  $\delta_{ext}((current\_state, e), next\_ev)$ 
    else:
        time = next_time
        output( $\lambda(current\_state)$ )
        current_state =  $\delta_{int}(current\_state)$ 
        last_time = time
  
```

t



Full Atomic DEVS Specification

$$M = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

X : set of input events

Y : set of output events

S : set of sequential states

$q_{init} : Q$

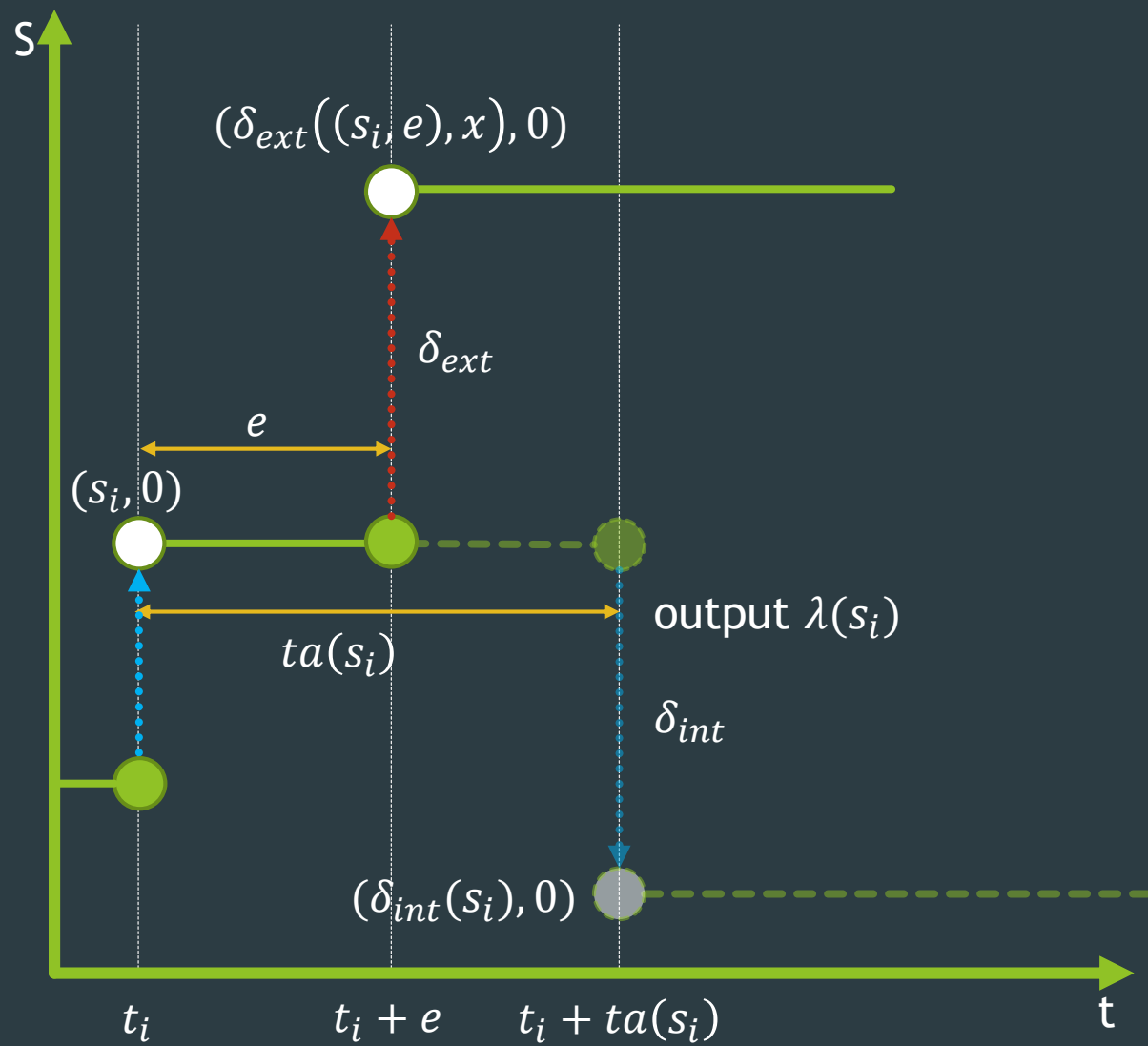
$$Q = \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\}$$

$\delta_{int} : S \rightarrow S$

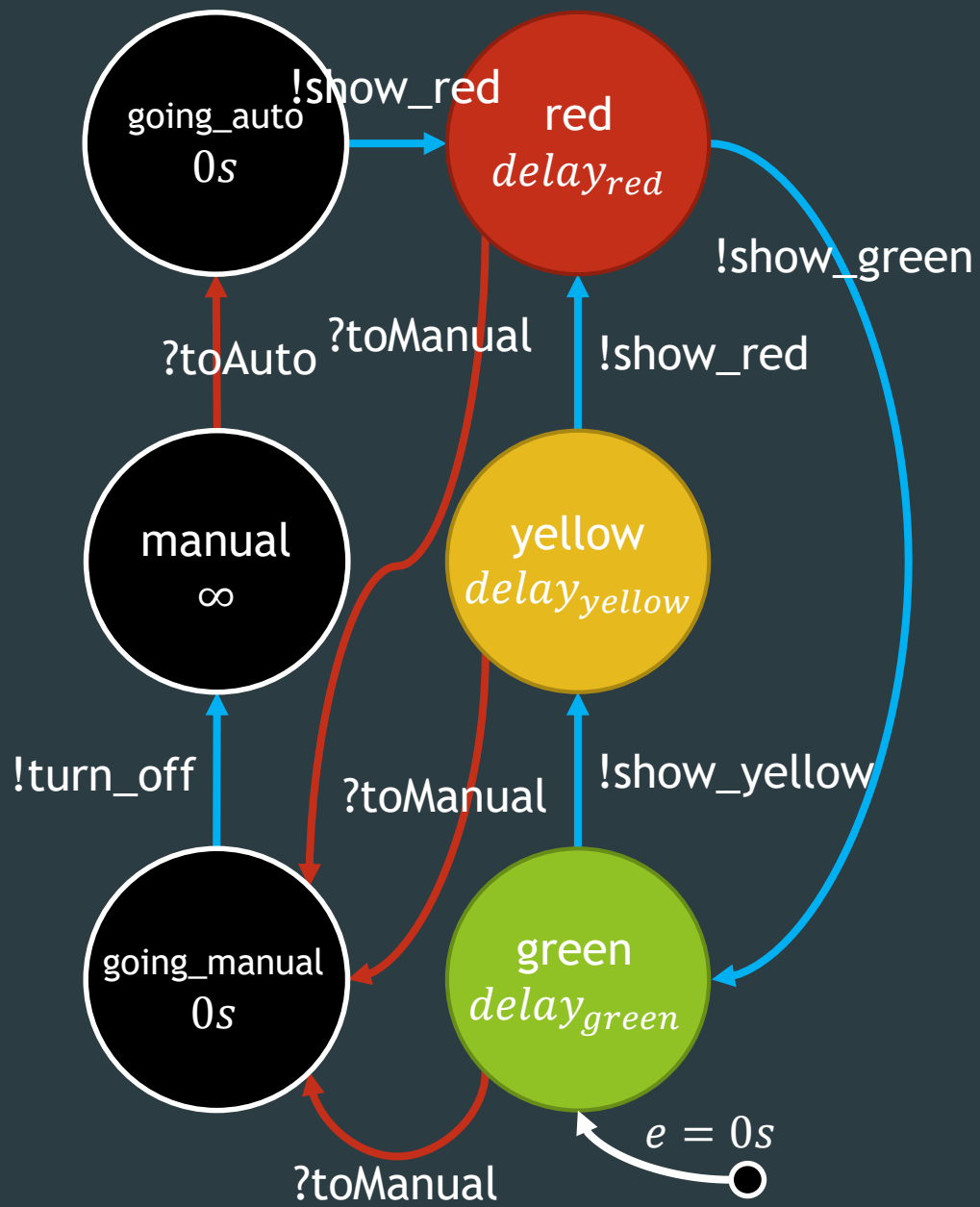
$\delta_{ext} : Q \times X \rightarrow S$

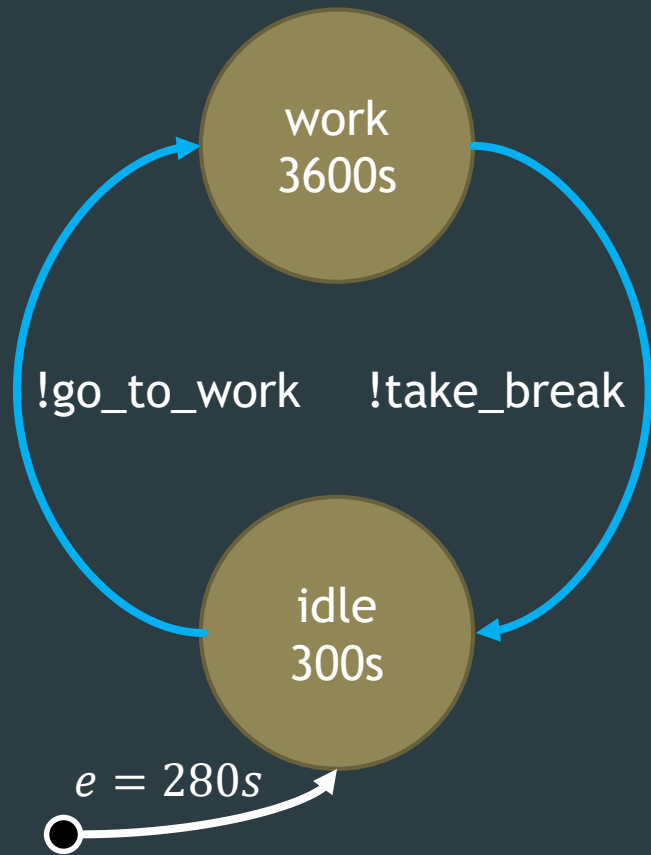
$\lambda : S \rightarrow Y \cup \{\phi\}$

$ta : S \rightarrow \mathbb{R}_{0,+\infty}^+$



Coupled Models

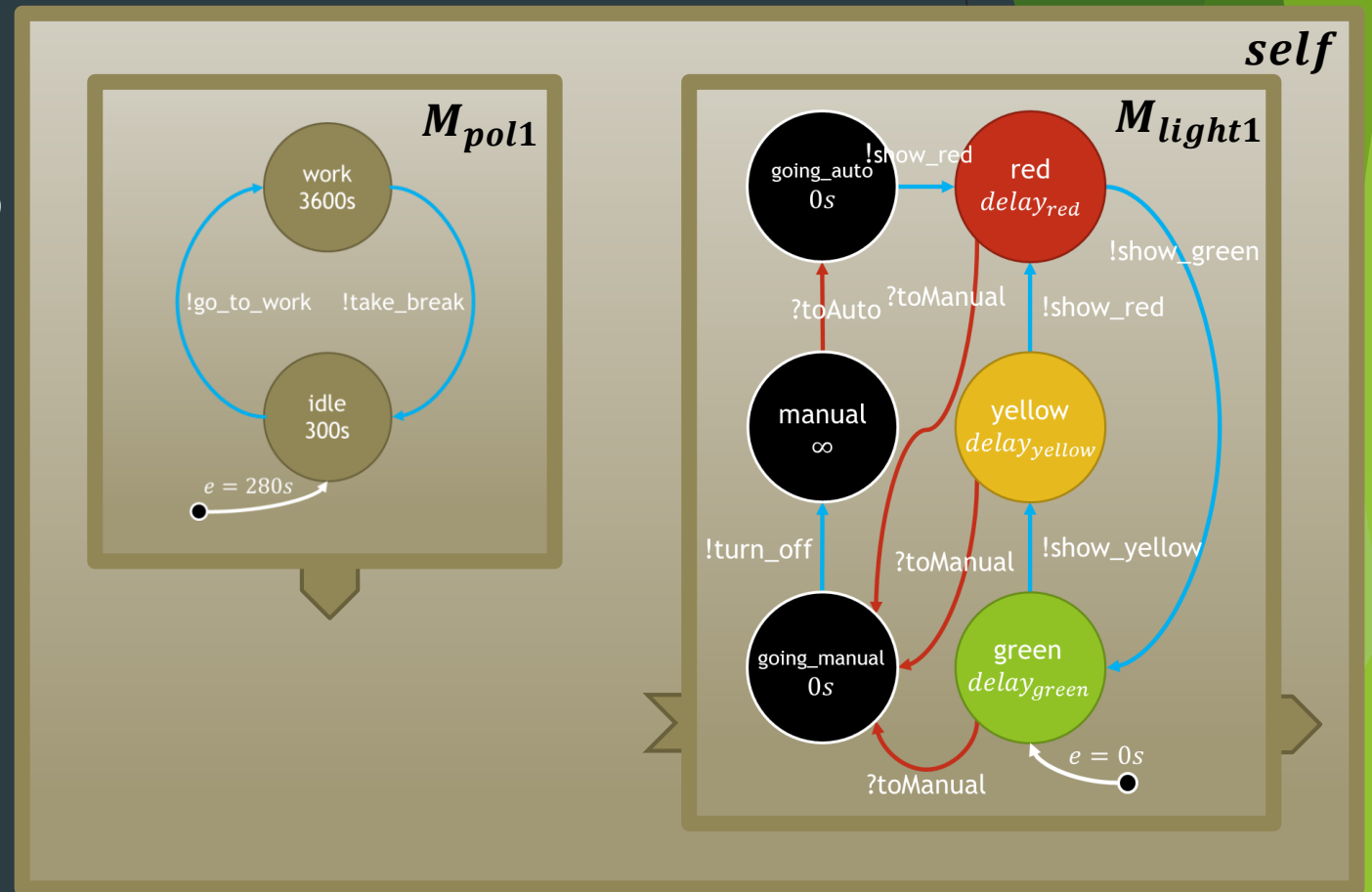




$$C = \langle D, MS \rangle$$

$$MS = \{M_i | i \in D\}$$

$$M_i = \langle X_i, Y_i, S_i, q_{init,i}, \delta_{int,i}, \delta_{ext,i}, \lambda_i, ta_i \rangle, \forall i \in D$$

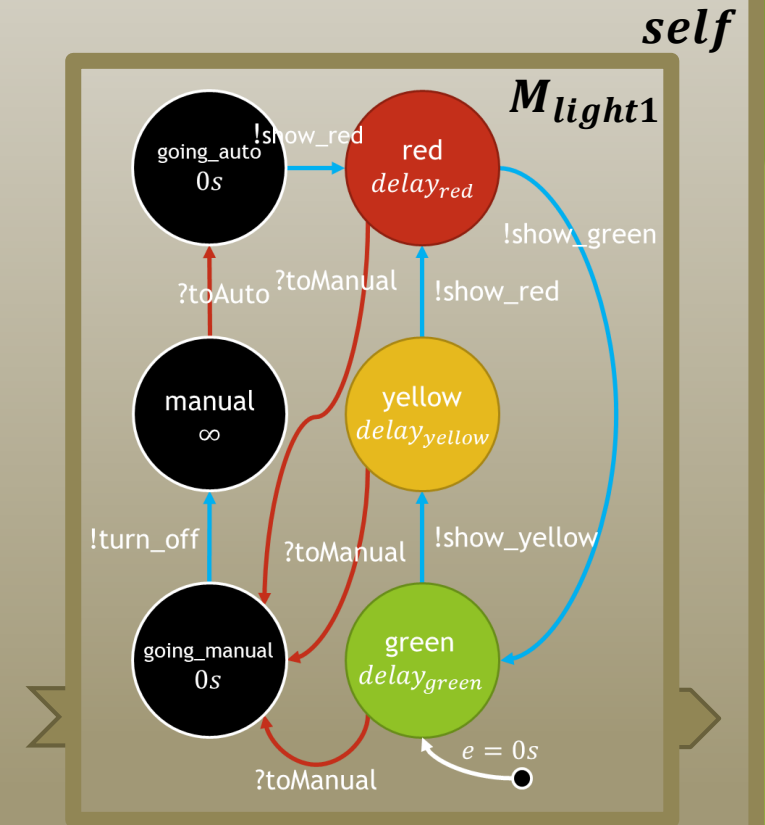
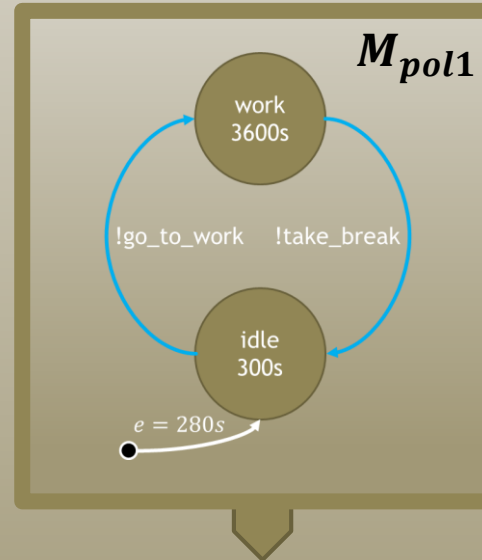


$$D = \{light_1, pol_1\}$$

$$C = \langle D, MS \rangle$$

$$MS = \{M_i | i \in D\}$$

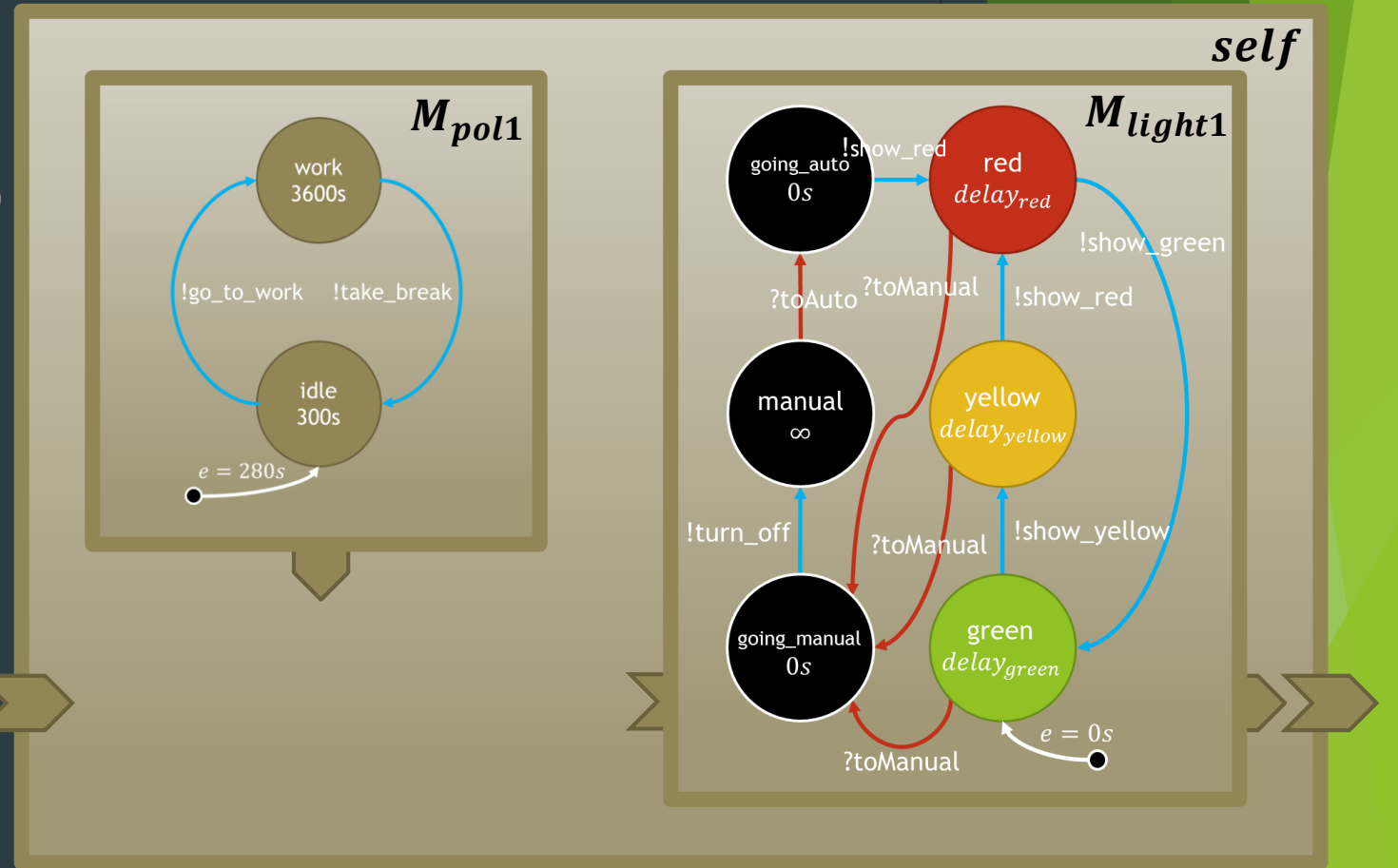
$$M_i = \langle X_i, Y_i, S_i, q_{init,i}, \delta_{int,i}, \delta_{ext,i}, \lambda_i, ta_i \rangle, \forall i \in D$$



$$C = \langle X_{self}, Y_{self}, D, MS \rangle$$

$$MS = \{M_i | i \in D\}$$

$$M_i = \langle X_i, Y_i, S_i, q_{init,i}, \delta_{int,i}, \delta_{ext,i}, \lambda_i, ta_i \rangle, \forall i \in D$$



$$C = \langle X_{self}, Y_{self}, D, MS, IS \rangle$$

$$MS = \{M_i | i \in D\}$$

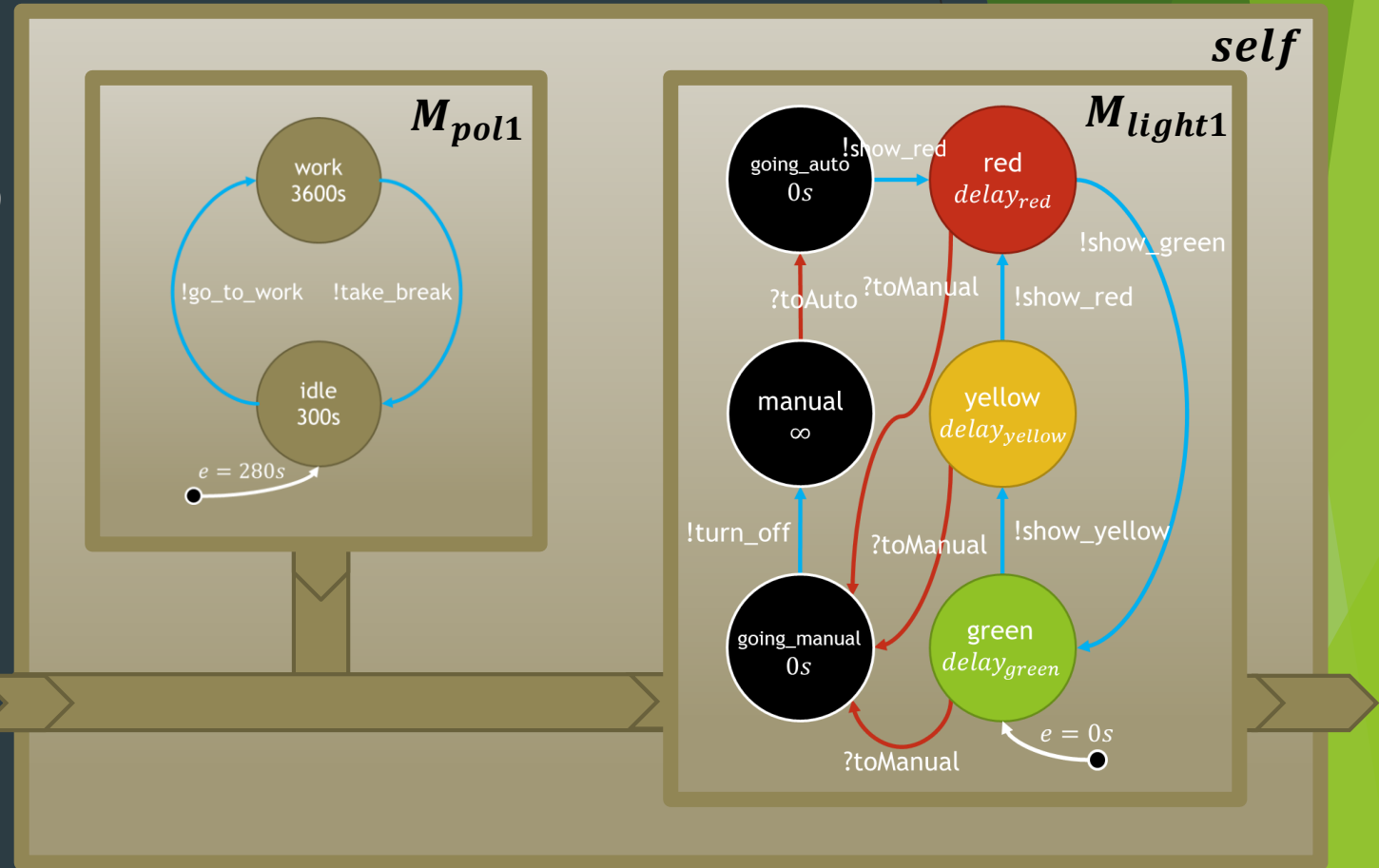
$$M_i = \langle X_i, Y_i, S_i, q_{init,i}, \delta_{int,i}, \delta_{ext,i}, \lambda_i, ta_i \rangle, \forall i \in D$$

$$IS = \{I_i | i \in D \cup \{self\}\}$$

$$\forall i \in D \cup \{self\} : I_i \subseteq D \cup \{self\}$$

$$\forall i \in D \cup \{self\} : i \notin I_i$$

I_i : Influences of i



$$C = \langle X_{self}, Y_{self}, D, MS, IS \rangle$$

$$MS = \{M_i | i \in D\}$$

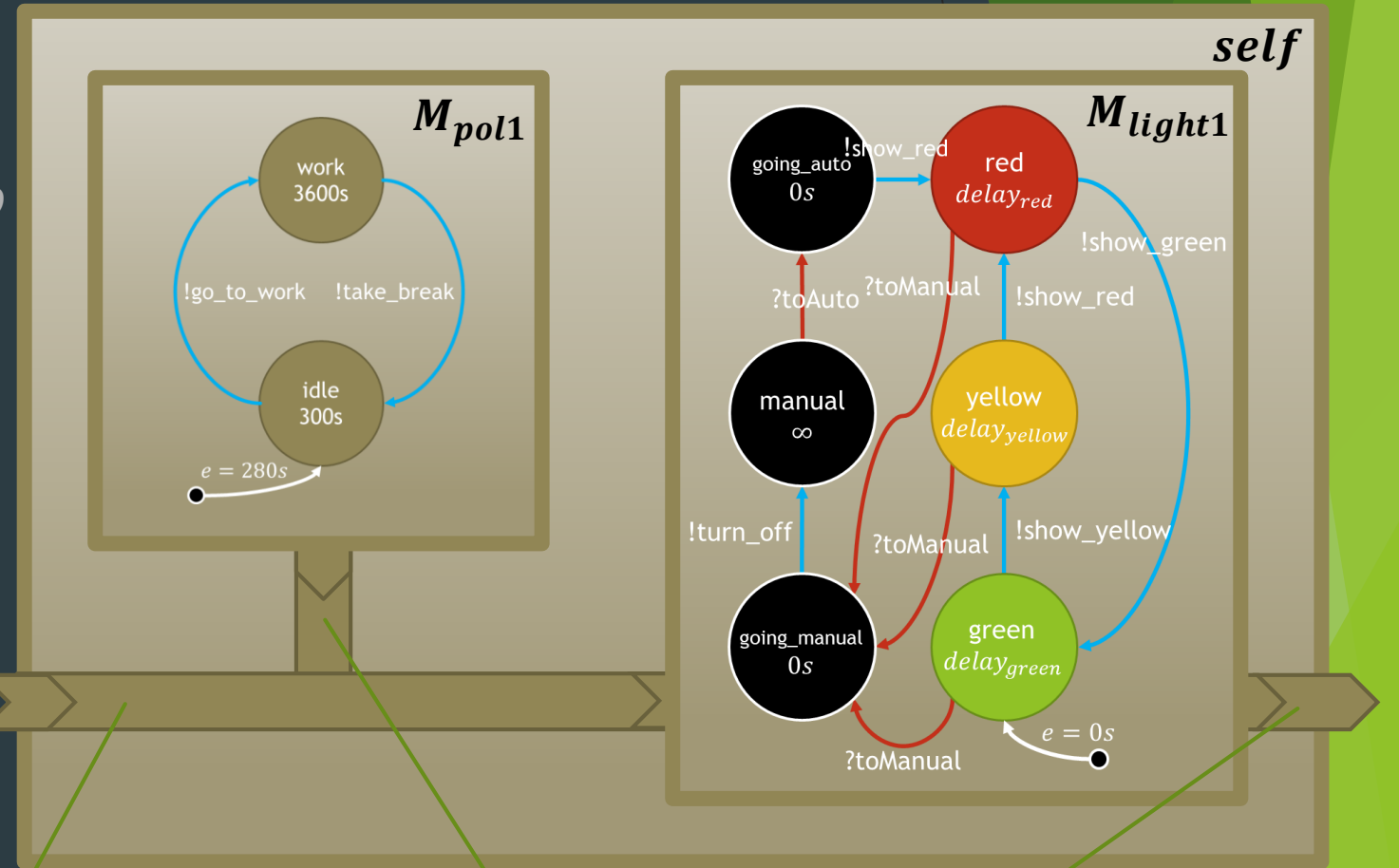
$$M_i = \langle X_i, Y_i, S_i, q_{init,i}, \delta_{int,i}, \delta_{ext,i}, \lambda_i, ta_i \rangle, \forall i \in D$$

$$IS = \{I_i | i \in D \cup \{self\}\}$$

$$\forall i \in D \cup \{self\} : I_i \subseteq D \cup \{self\}$$

$$\forall i \in D \cup \{self\} : i \notin I_i$$

I_i : Influences of i



$$I_{self} = \{light1\}$$

$$I_{pol1} = \{light1\}$$

$$I_{light1} = \{self\}$$

$$Y_{pol1} = \{go_to_work, take_break\}$$

$$C = \langle X_{self}, Y_{self}, D, MS, IS, ZS \rangle$$

$$MS = \{M_i | i \in D\}$$

$$M_i = \langle X_i, Y_i, S_i, q_{init,i}, \delta_{int,i}, \delta_{ext,i}, \lambda_i, ta_i \rangle, \forall i \in D$$

$$IS = \{I_i | i \in D \cup \{self\}\}$$

$$\forall i \in D \cup \{self\} : I_i \subseteq D \cup \{self\}$$

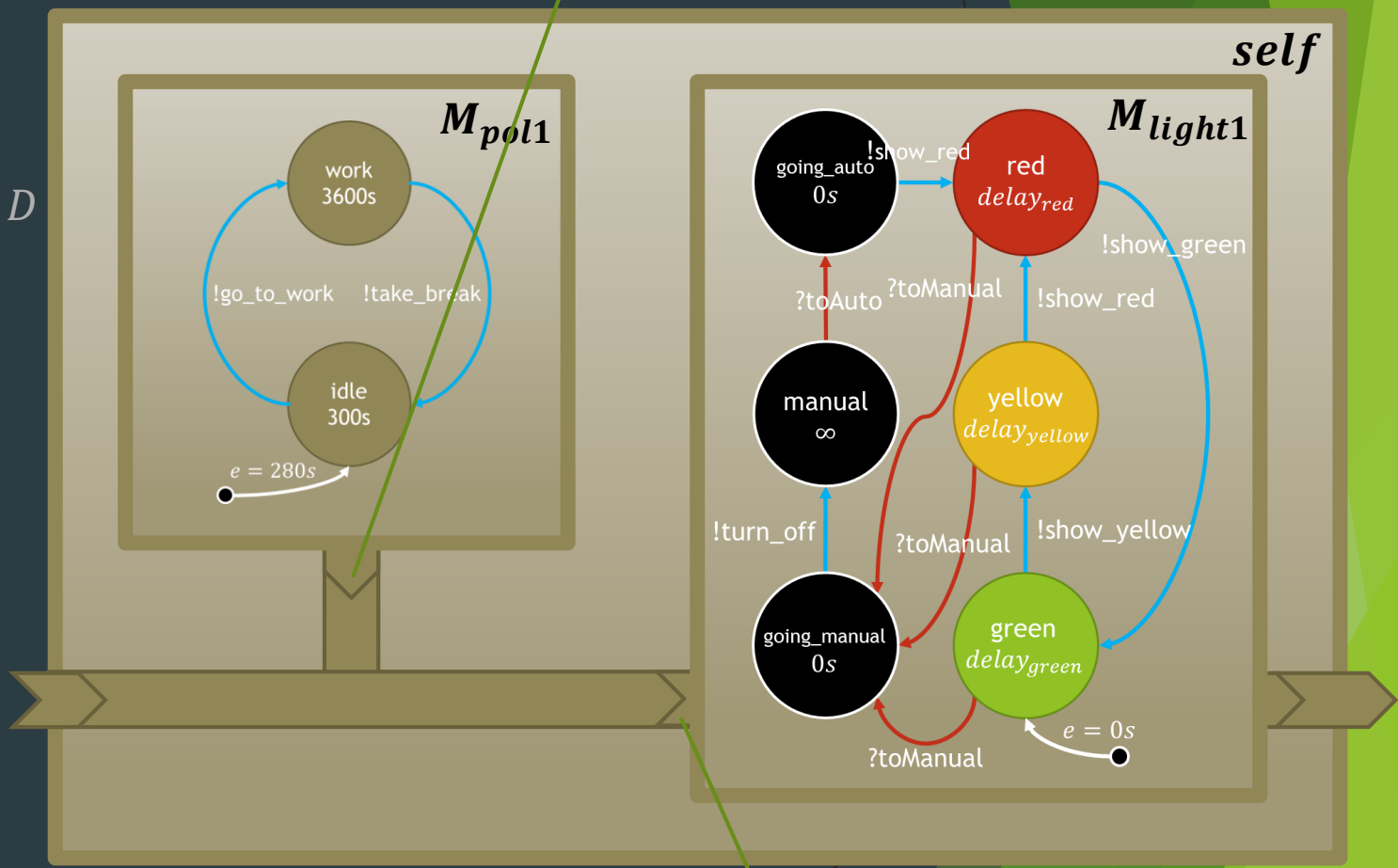
$$\forall i \in D \cup \{self\} : i \notin I_i$$

$$ZS = \{Z_{i,j} | i \in D \cup \{self\}, j \in I_i\}$$

$$Z_{self,j} : X_{self} \rightarrow X_j, \forall j \in D$$

$$Z_{i,self} : Y_i \rightarrow Y_{self}, \forall i \in D$$

$$Z_{i,j} : Y_i \rightarrow X_j, \forall i, j \in D$$



$$X_{light1} = \{toManual, toAuto\}$$

$$C = \langle X_{self}, Y_{self}, D, MS, IS, ZS \rangle$$

$$MS = \{M_i | i \in D\}$$

$$M_i = \langle X_i, Y_i, S_i, q_{init,i}, \delta_{int,i}, \delta_{ext,i}, \lambda_i, ta_i \rangle, \forall i \in D$$

$$IS = \{I_i | i \in D \cup \{self\}\}$$

$$\forall i \in D \cup \{self\} : I_i \subseteq D \cup \{self\}$$

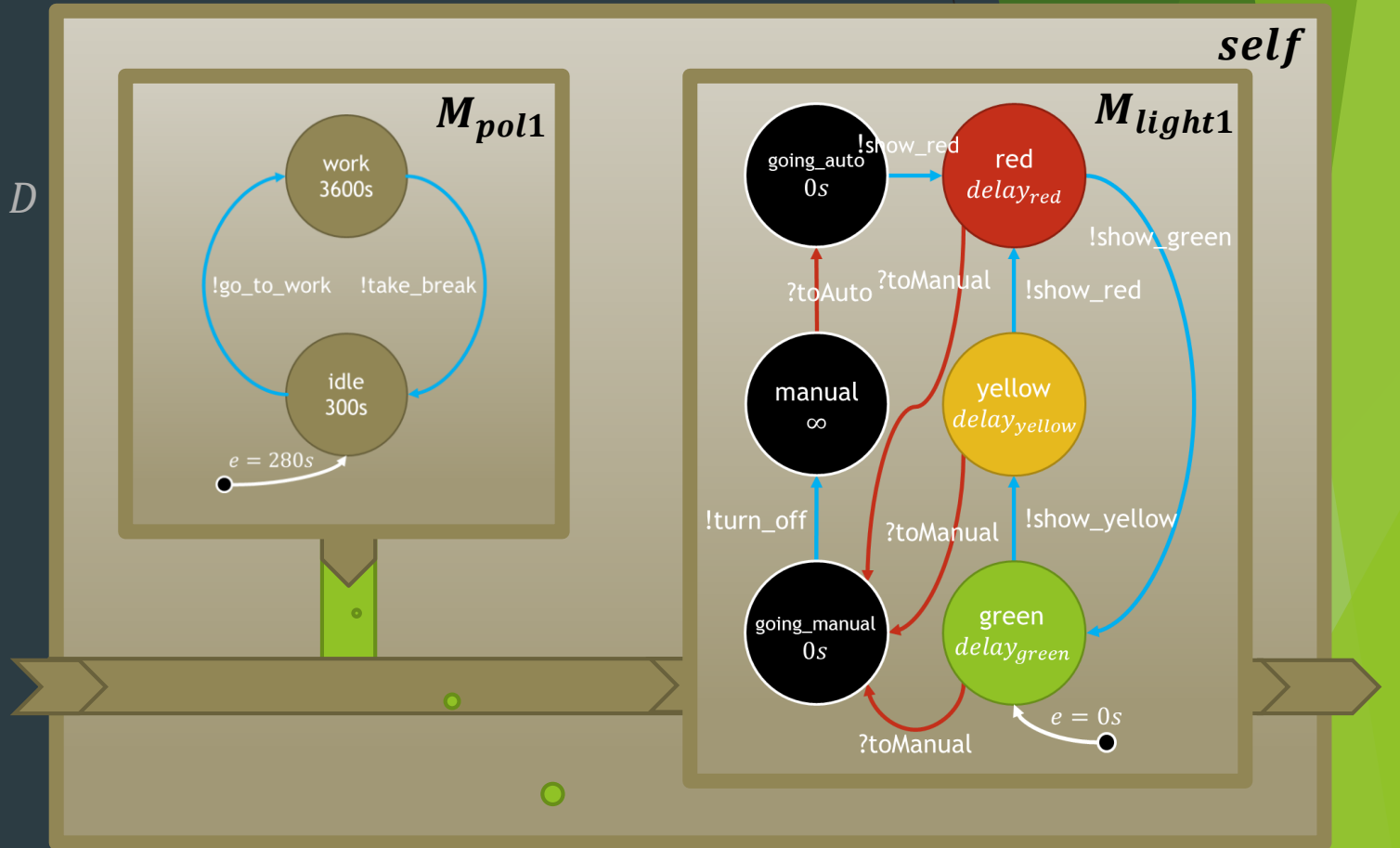
$$\forall i \in D \cup \{self\} : i \notin I_i$$

$$ZS = \{Z_{i,j} | i \in D \cup \{self\}, j \in I_i\}$$

$$Z_{self,j} : X_{self} \rightarrow X_j, \forall j \in D$$

$$Z_{i,self} : Y_i \rightarrow Y_{self}, \forall i \in D$$

$$Z_{i,j} : Y_i \rightarrow X_j, \forall i, j \in D$$



$$C = \langle X_{self}, Y_{self}, D, MS, IS, ZS, select \rangle$$

$$MS = \{M_i | i \in D\}$$

$$M_i = \langle X_i, Y_i, S_i, q_{init,i}, \delta_{int,i}, \delta_{ext,i}, \lambda_i, ta_i \rangle, \forall i \in D$$

$$IS = \{I_i | i \in D \cup \{self\}\}$$

$$\forall i \in D \cup \{self\} : I_i \subseteq D \cup \{self\}$$

$$\forall i \in D \cup \{self\} : i \notin I_i$$

$$ZS = \{Z_{i,j} | i \in D \cup \{self\}, j \in I_i\}$$

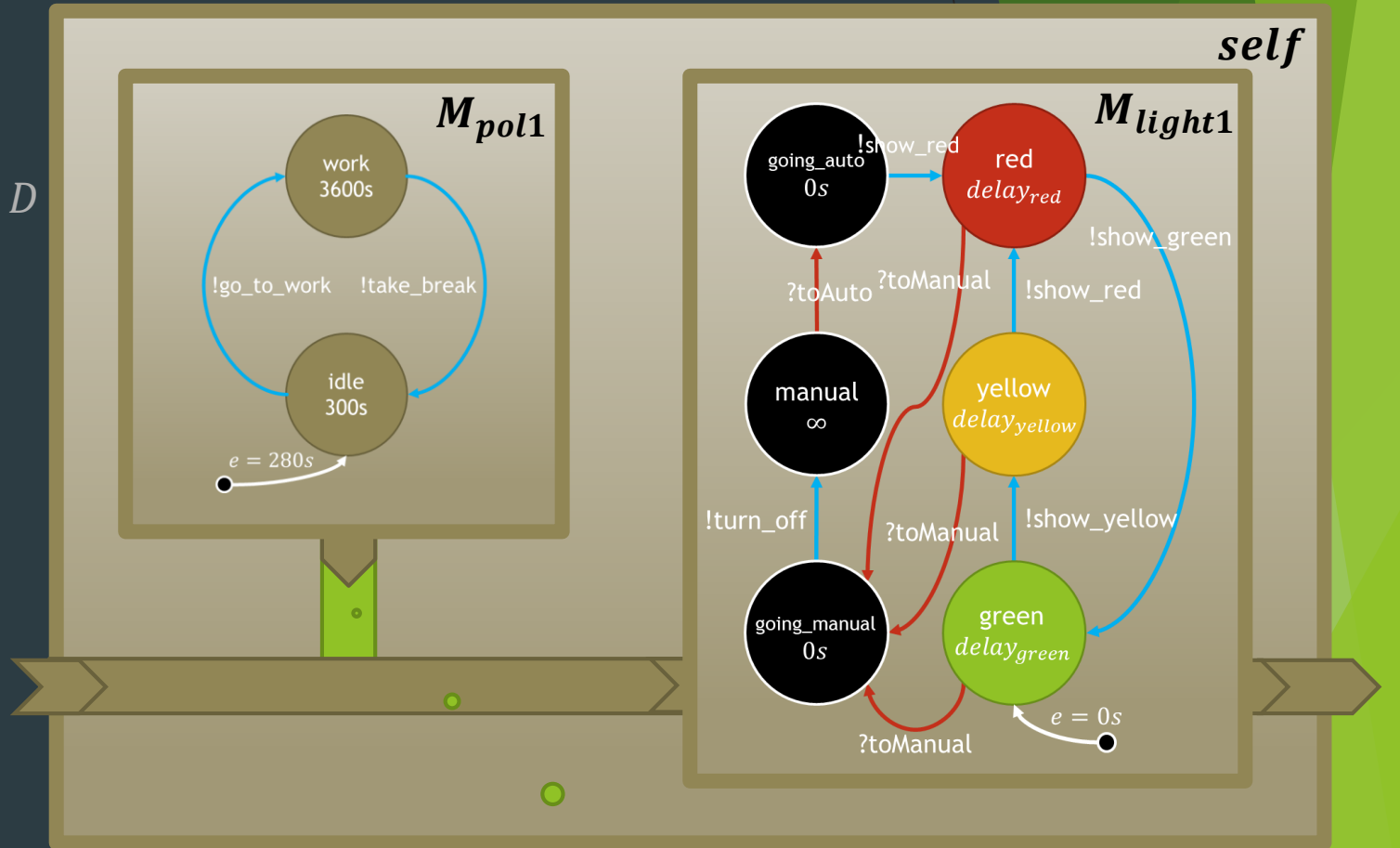
$$Z_{self,j} : X_{self} \rightarrow X_j, \forall j \in D$$

$$Z_{i,self} : Y_i \rightarrow Y_{self}, \forall i \in D$$

$$Z_{i,j} : Y_i \rightarrow X_j, \forall i, j \in D$$

$$select : 2^D \rightarrow D$$

$$\forall E \subseteq D, E \neq \emptyset : select(E) \in E$$



```
from pypdevs.DEVS import *

from trafficlight import TrafficLight
from policeman import Policeman

def translate(in_evt):
    mapping = {"take_break": "toAuto",
              "go_to_work": "toManual"}
    return mapping[in_evt]

class TrafficLightSystem(CoupledDEVS):
    def __init__(self):
        CoupledDEVS.__init__(self, "system")
        self.light = self.addSubModel(TrafficLight())
        self.police = self.addSubModel(Policeman())
        self.connectPorts(self.police.out, self.light.interrupt, translate)

    def select(self, immlist):
        if self.police in immlist:
            return self.police
        else:
            return self.light
```

___ Current Time: 0.00 _____

INITIAL CONDITIONS in model <system.light>

Initial State: green

Next scheduled internal transition at time 57.00

INITIAL CONDITIONS in model <system.policeman>

Initial State: idle

Next scheduled internal transition at time 20.00

__ Current Time: 20.00 _____

EXTERNAL TRANSITION in model <system.light>

Input Port Configuration:

port <interrupt>:
toManual

New State: going_manual

Next scheduled internal transition at time 20.0

INTERNAL TRANSITION in model <system.policeman>

New State: working

Output Port Configuration:

port <output>:
go_to_work

Next scheduled internal transition at time 3620.00

__ Current Time: 20.00 _____

INTERNAL TRANSITION in model <system.light>

Output Port Configuration:

port <observer>:

turn_off

New State: manual

Next scheduled internal transition at time inf

__ Current Time: 3620.00 _____

EXTERNAL TRANSITION in model <system.light>

Input Port Configuration:

port <interrupt>:

toAuto

New State: going_auto

Next scheduled internal transition at time 3620.00

INTERNAL TRANSITION in model <system.policeman>

New State: idle

Output Port Configuration:

port <output>:

take_break

Next scheduled internal transition at time 3920.00

__ Current Time: 3620.00 _____

INTERNAL TRANSITION in model <system.light>

Output Port Configuration:

port <observer>:

show_red

New State: red

Next scheduled internal transition at time 3680.00

__ Current Time: 3620.00 _____

EXTERNAL TRANSITION in model <system.light>

Input Port Configuration:

port <interrupt>:

toAuto

New State: going_auto

Next scheduled internal transition at time 3620.00

INTERNAL TRANSITION in model <system.policeman>

New State: idle

Output Port Configuration:

port <output>:

take_break

Next scheduled internal transition at time 3920.00

Current Time: 3920.00

CONFLICT between models:

<system.light>

* <system.policeman>

EXTERNAL TRANSITION in model <system.light>

Input Port Configuration:

port <interrupt>:

toManual

New State: going_manual

Next scheduled internal transition at time 3920.00

INTERNAL TRANSITION in model <system.policeman>

New State: work

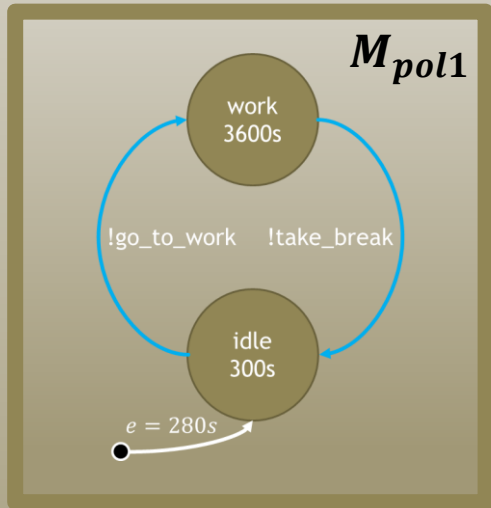
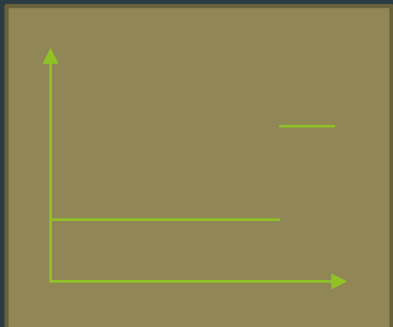
Output Port Configuration:

port <output>:

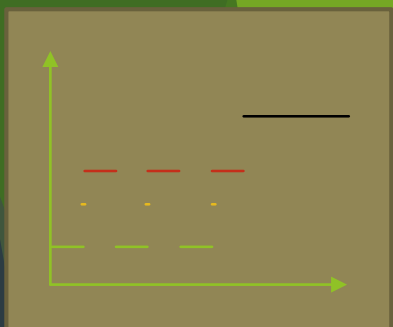
go_to_work

Next scheduled internal transition at time 7520.00

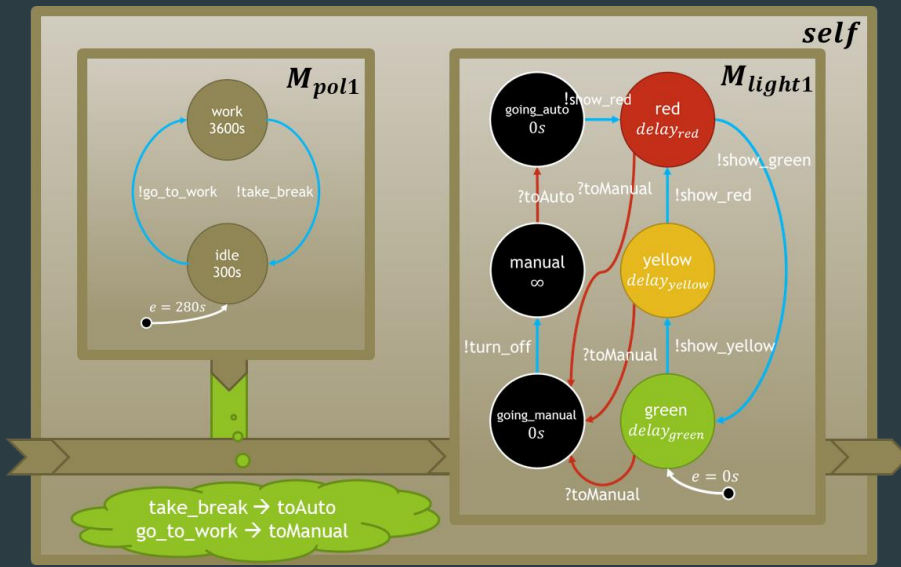
Closure under Coupling



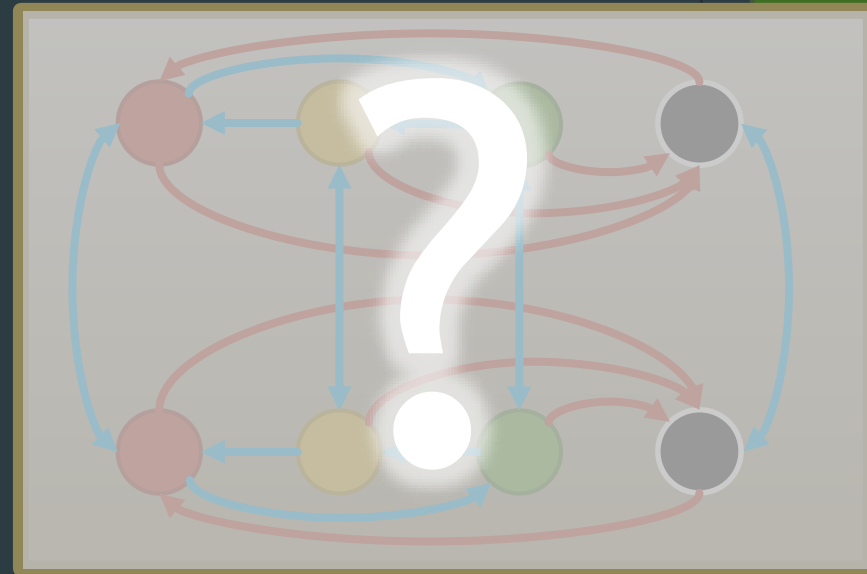
take_break → toAuto
go_to_work → toManual



?



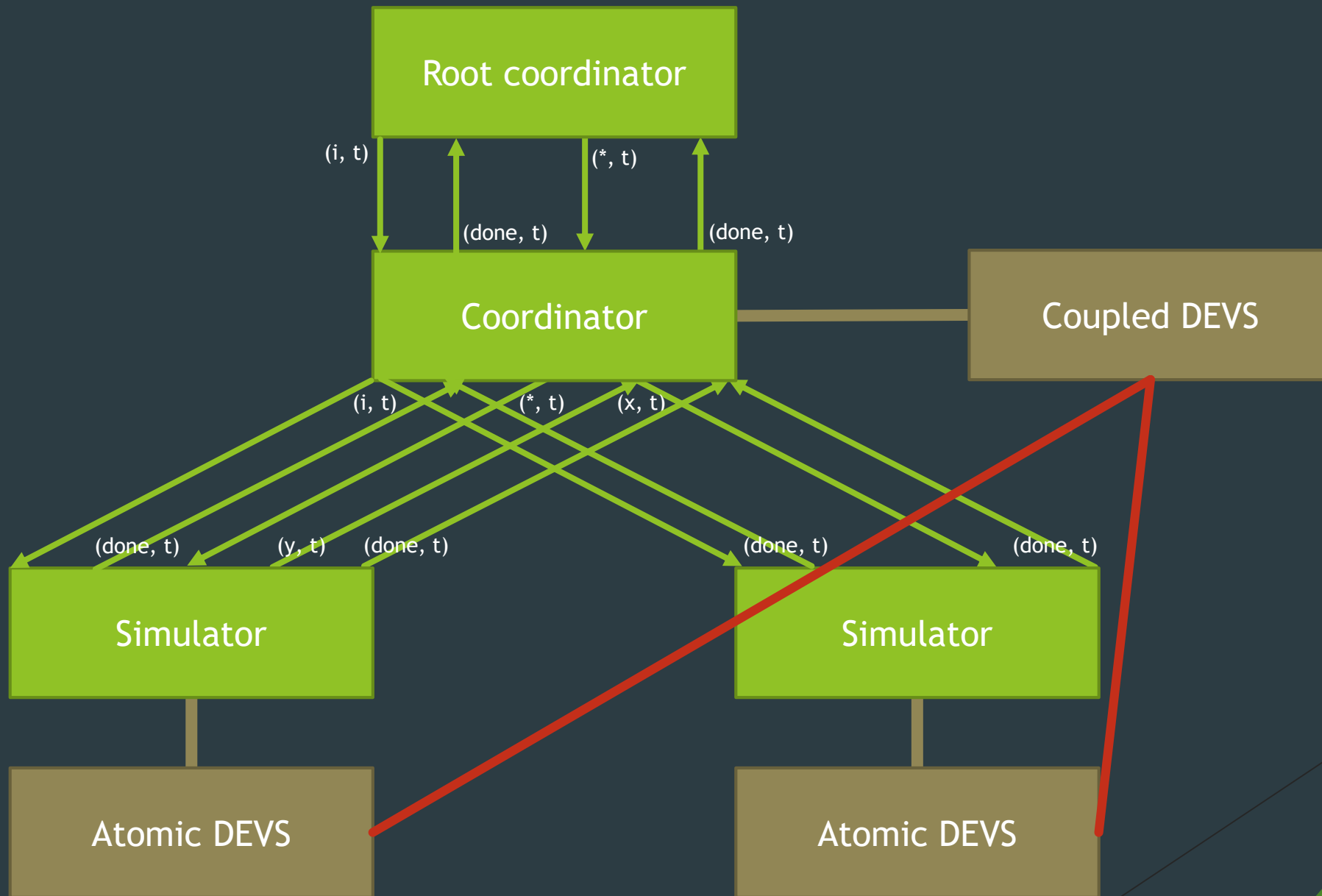
flatten



$$CM = \langle X_{self}, Y_{self}, D, MS, IS, ZS \rangle$$

$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

Hierarchical Simulator



DEVS Semantics

	Operational Semantics	Denotational Semantics
Atomic DEVS	Abstract Simulator	[1]
Coupled DEVS	Hierarchical Simulator	Closure under Coupling

[1] Ashvin Radiya and Robert G. Sargent. A logic-based foundation of discrete event modeling and simulation. ACM Transactions on Modeling and Computer Simulation, 1(1):3-51, 1994.

Conclusions

- ▶ Atomic DEVS
- ▶ Coupled DEVS
- ▶ Closure under coupling
- ▶ Abstract Simulator

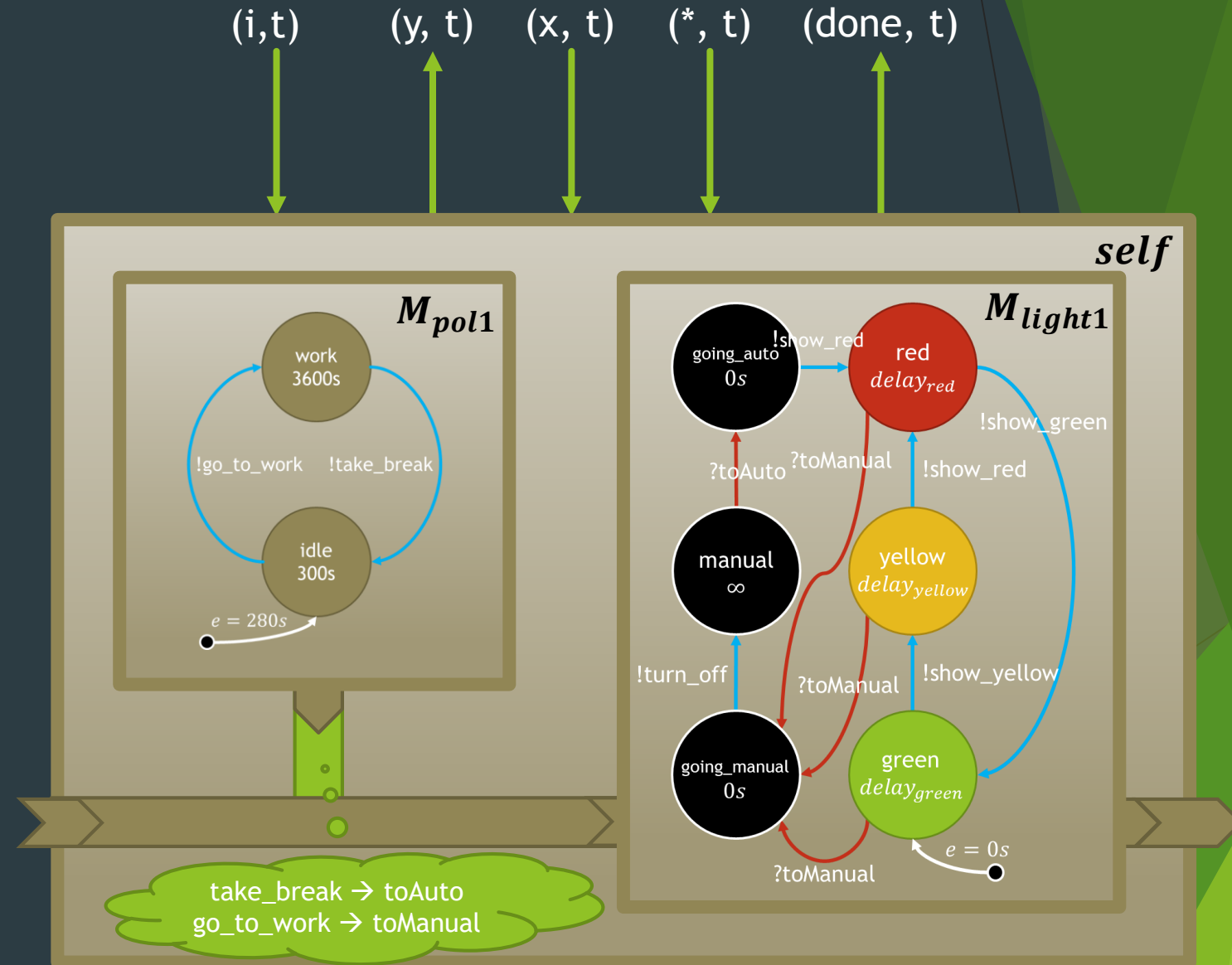


Table Of Contents

Examples

- Generator
- Simple queue
- Coupling
- Simulation
- Tracing
- Termination
- Simulation time

Previous topic

[Differences from PyDEVs](#)

Next topic

[Examples for Parallel DEVs](#)

This Page

[Show Source](#)

Quick search

Enter search terms or a module, class or function name.

Examples

A small *trafficModel* and corresponding *trafficExperiment* file is included in the *examples* folder of the PyPDEVs distribution. This (completely working) example is slightly too big to use as a first introduction to PyPDEVs and therefore this page will start with a very simple example.

For this, we will first introduce a simplified queue model, which will be used as the basis of all our examples. The complete model can be downloaded: [queue_example_classic.py](#).

This section should provide you with all necessary information to get you started with creating your very own PyPDEVs simulation. More advanced features are presented in the next section.

Generator

Somewhat simpler than a queue even, is a generator. It will simply create a message to send after a certain delay and then it will stop doing anything.

Informally, this would result in a DEVs specification as:

- Time advance function returns the waiting time to generate the message, infinity after the message was created
- Output function outputs the generated message

<http://msdl.cs.mcgill.ca/projects/PythonPDEVs>

Limitations of Classic DEVS

- ▶ Parallel implementation
 - ▶ Parallel DEVS [1]
- ▶ Select function is artificial
 - ▶ Parallel DEVS [1]
- ▶ Dynamic Structure systems
 - ▶ Dynamic Structure DEVS [2]

[1] A.C.-H. Chow. Parallel DEVS: A parallel, hierarchical, modular modeling formalism and its distributed simulator. Transactions of the Society for Computer Simulation International, 13(2):55-68, 1996.

[2] F. Barros. The dynamic structure discrete event system specification formalism. Transactions of the Society for Computer Simulation International, 13(1):35-46, 1996.

Formalisms

Dynamic Structure

Real-time

Cell DEVS

Verification

Standardization

Tools

Languages

Interoperable

Performance

Algorithms

Activity

Distribution

Parallel

Model libraries

Example

Reusable

Applications



<http://msdl.cs.mcgill.ca/projects/PythonPDEVS>