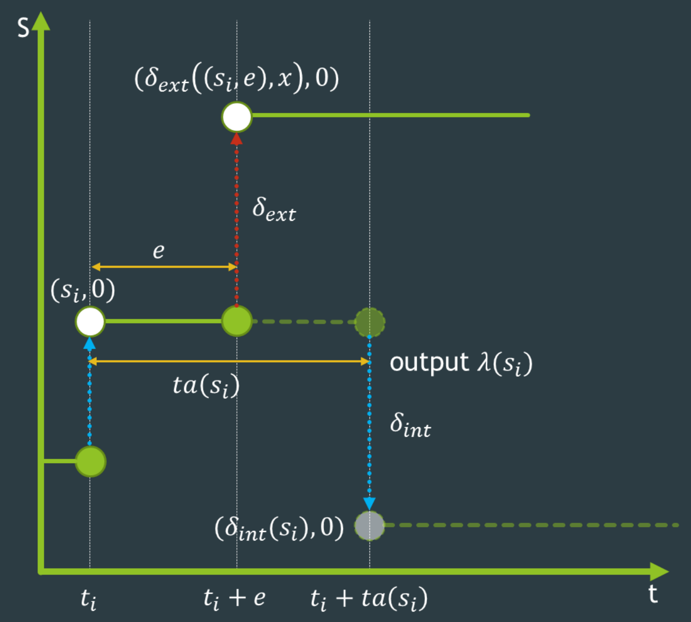


Parallel DEVS

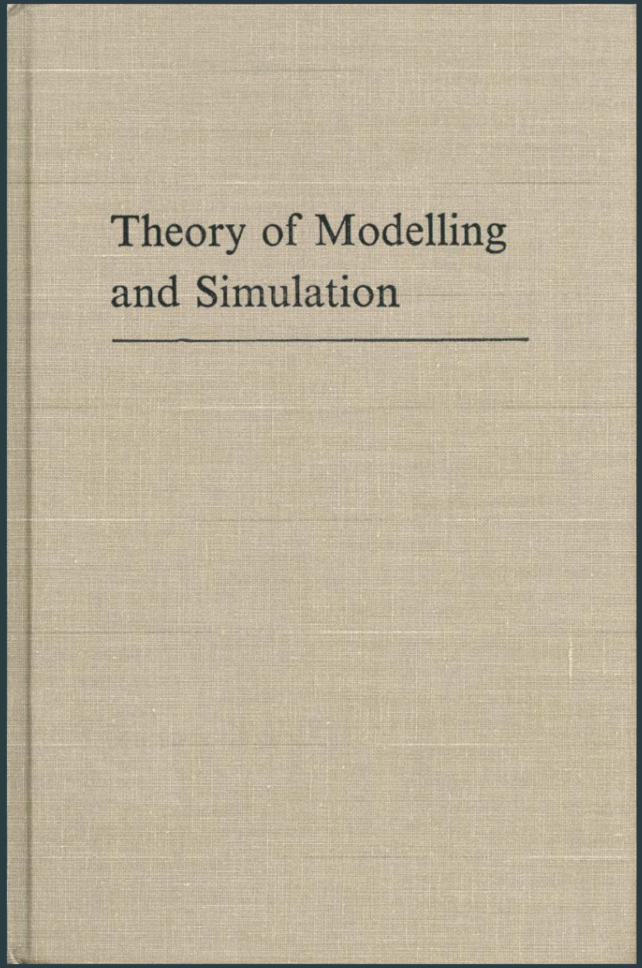
An Introduction Using PythonPDEVS

Yentl Van Tendeloo, Hans Vangheluwe

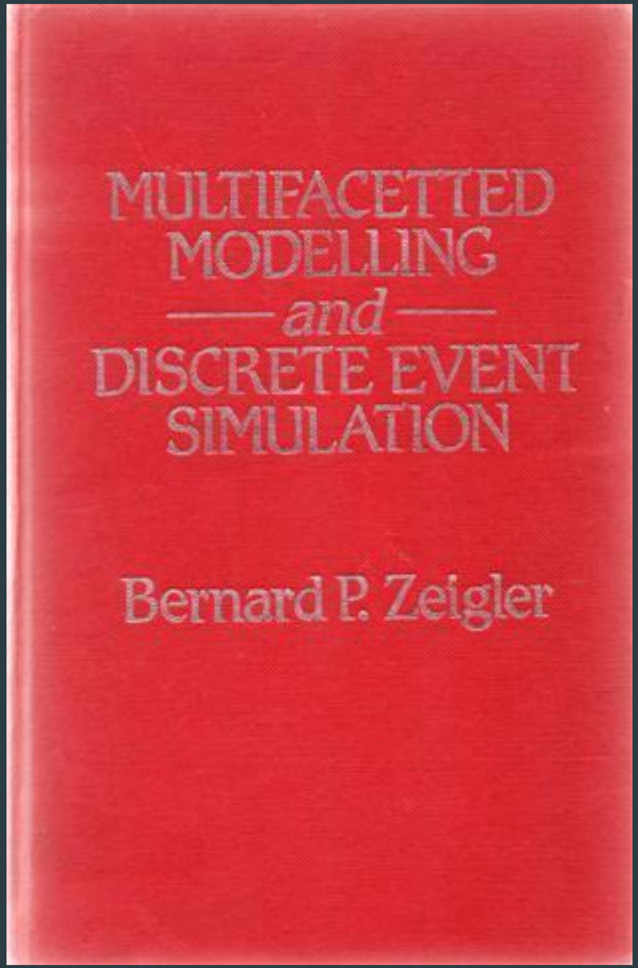
Universiteit Antwerpen

McGill

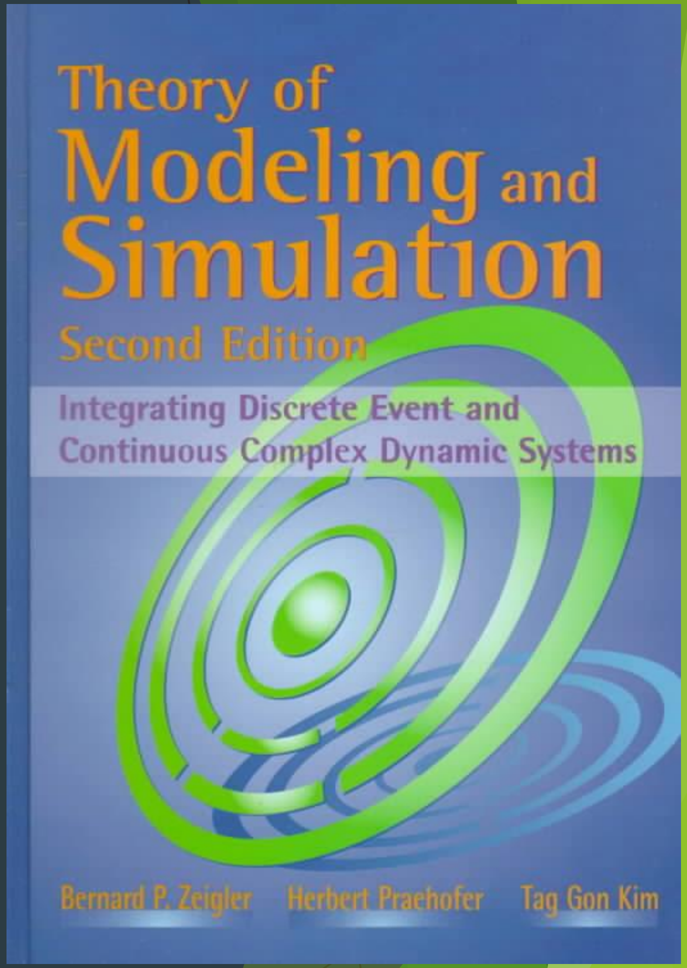
Introduction



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



Bernard P. Zeigler.
Multifacetted 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.

Parallel DEVS: A parallel, hierarchical, modular modeling formalism and its distributed simulator

A. C.-H. Chow

IBM Corporation, P.O. Box 180124, Austin, Texas 78758
Tel: 512-838-8069 FAX: 512-250-2346 E-mail: alexc@austin.ibm.com

We present a parallel, hierarchical, modular Discrete Event System Specification (P-DEVS) modeling formalism which provides a modeler with both conceptual and parallel execution benefits. The parallel formalism distinguishes between transition collisions and ordinary external events in the external transition function of DEVS models. Such separation enables us to extend the modeling capability of the collisions. The formalism also does away with the necessity for tie-breaking of simultaneously scheduled events, as embodied in the select *function. We next present a design for the parallel simulation procedures needed to prove the formalism's soundness and to serve as a reference for implementation. We then discuss a prototype implementation that affords a high degree of flexibility by mechanizing the "closure under coupling" property of the Parallel DEVS formalism and the object-oriented characteristics.*

Keywords: System specification, discrete event simulation, DEVS, object-oriented modeling and simulation, distributed/parallel simulation, modeling methodology

Abstract Simulator for the Parallel DEVS Formalism

Alex ChungHen Chow

Object Technology Products

IBM Corp.
Austin, TX 78758
alexc@austin.ibm.com

Bernard P. Zeigler
Doo Hwan Kim

Department of
Electrical and Computer Engineering
The University of Arizona
Tucson, AZ 85721
zeigler@ece.arizona.edu
dhkim@ece.arizona.edu

Alex ChungHen Chow.
Parallel DEVS: a parallel, hierarchical, modular modeling formalism and its distributed simulator.
Transactions of the Society for Computer Simulation. 1996; 13(2): 55-68.

Alex ChungHen Chow, Bernard P. Zeigler and Doo Hwan Kim.
Abstract simulator for the parallel DEVS formalism.
Fifth Annual Conference on AI, and Planning in High Autonomy Systems. 1994, pp. 157-163.

An overview of PythonPDEVS

Yentl Van Tendeloo[1] Hans Vangheluwe[1,2]
[1] University of Antwerp, Belgium
[2] McGill University, Canada

Yentl.VanTendeloo@uantwerpen.be
Hans.Vangheluwe@uantwerpen.be

Simulation

Methodology

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

DOI: 10.1177/0037549716678330
sim.sagepub.com
SAGE SCS

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.

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

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

Meijin++

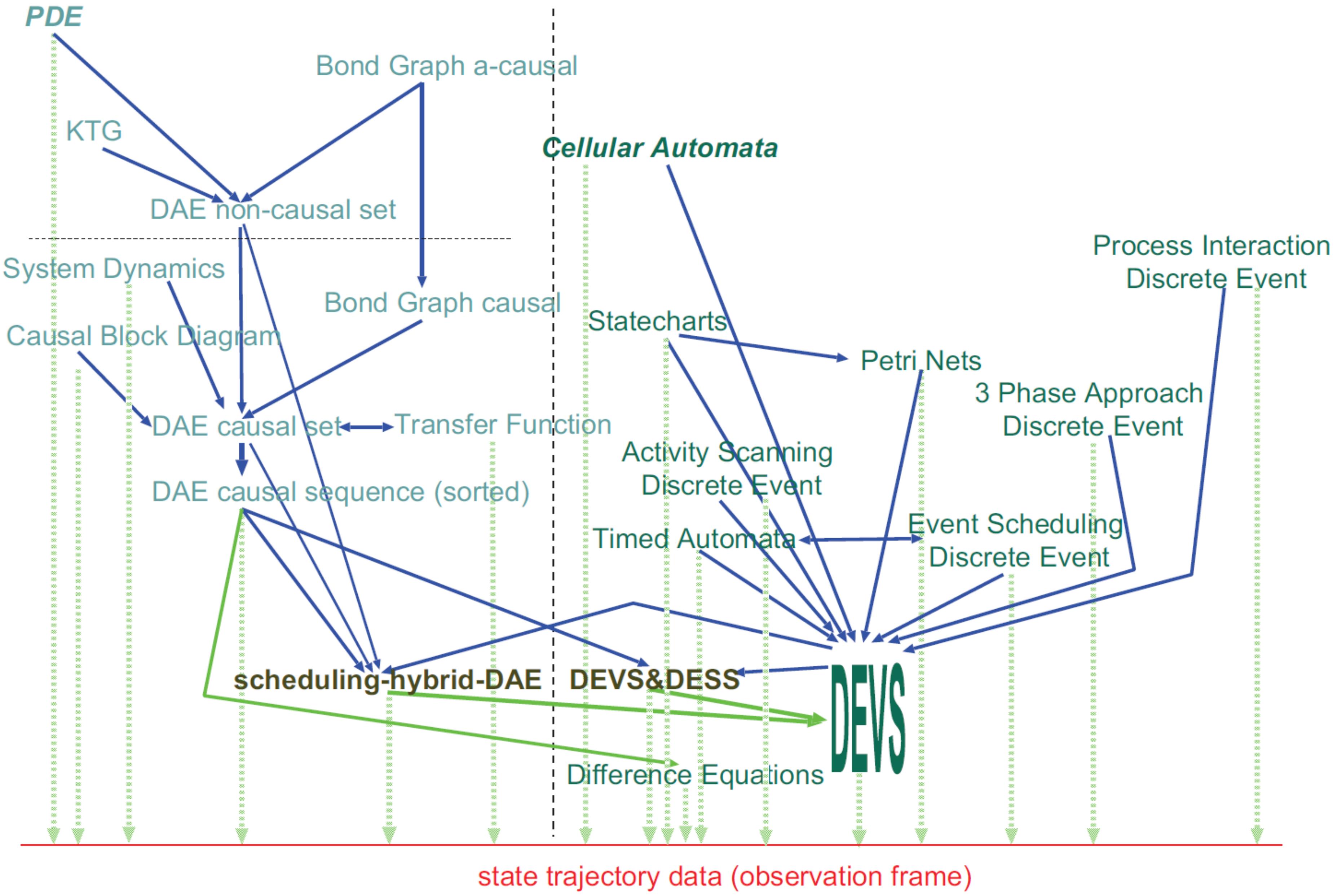
GPSS

…

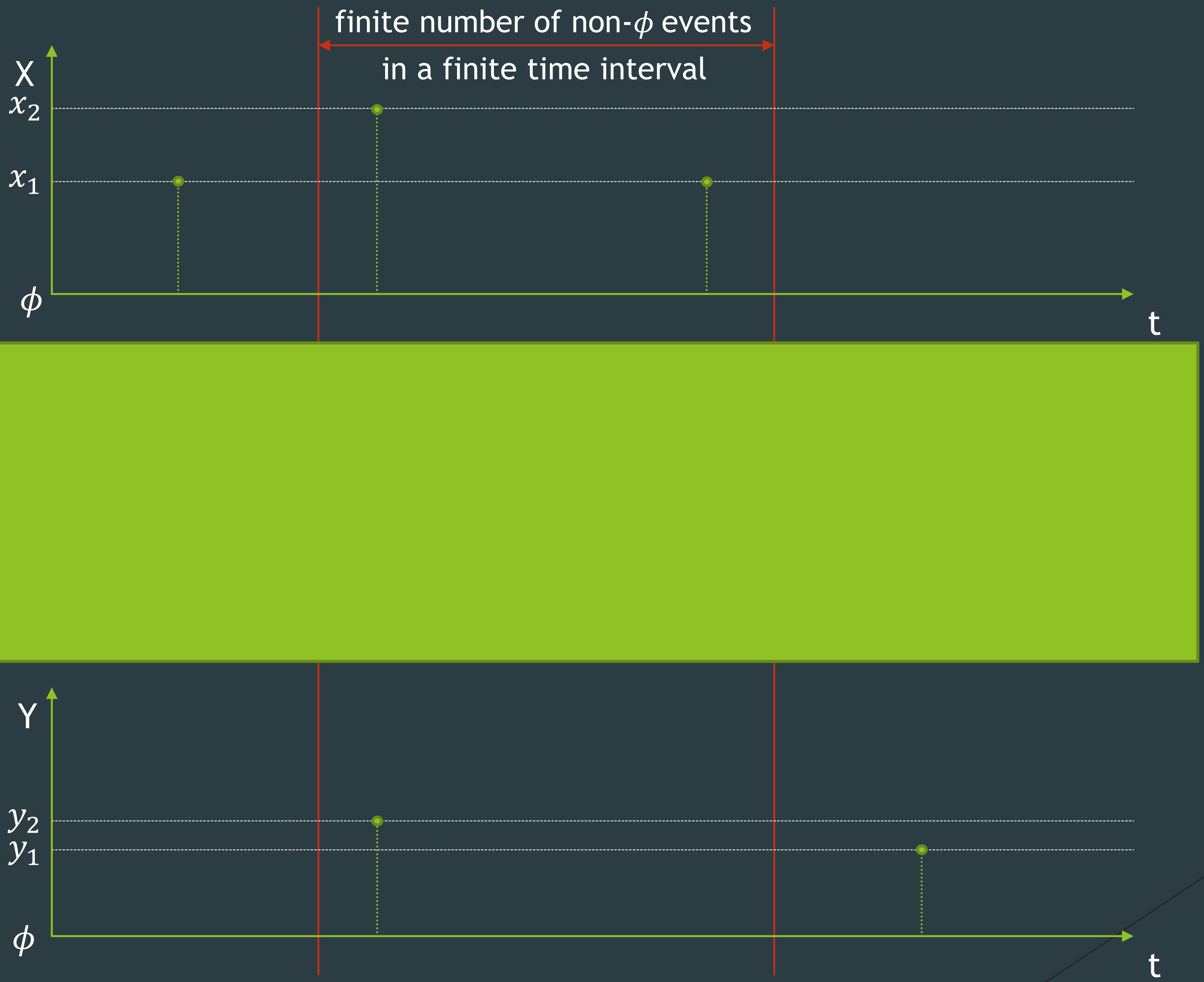
SimScript

DEVS

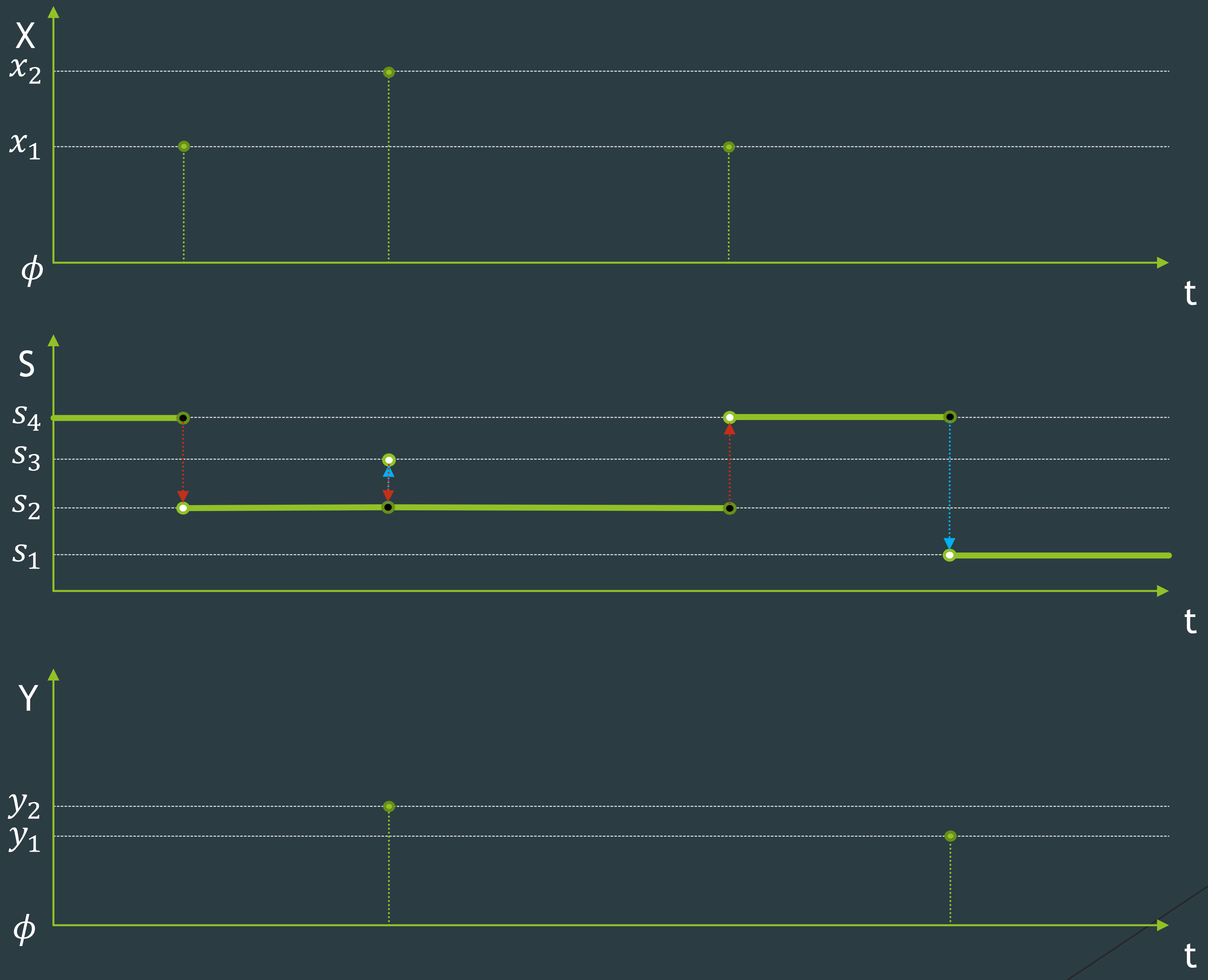
= modular simulation assembly language



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.



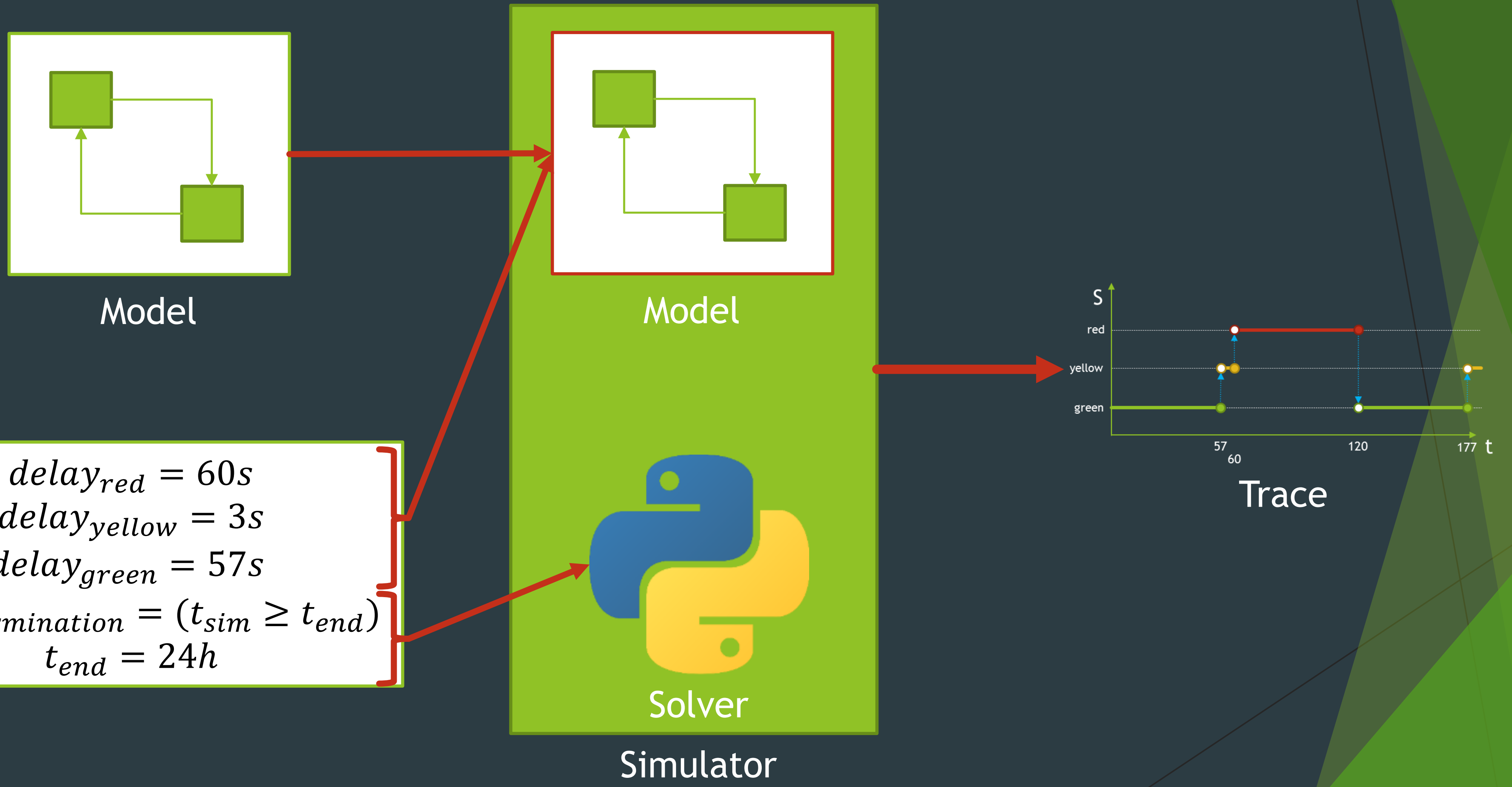
finite number of non-ϕ events
in a finite time interval
X
x_2
x_1
ϕ
t
Y
y_2
y_1
ϕ
t



X
x_2
x_1
ϕ
t
S
s_4
s_3
s_2
s_1
t
Y
y_2
y_1
ϕ
t

Experimentation

Simulation



$$delay_{red} = 60s$$
$$delay_{yellow} = 3s$$
$$delay_{green} = 57s$$
$$cond_{termination} = (t_{sim} \geq t_{end})$$
$$t_{end} = 24h$$

Concrete Syntax

simple_experiment.py

```
from pypdevs.simulator import Simulator

from mymodel import MyModel

model = MyModel()
simulator = Simulator(model)

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>

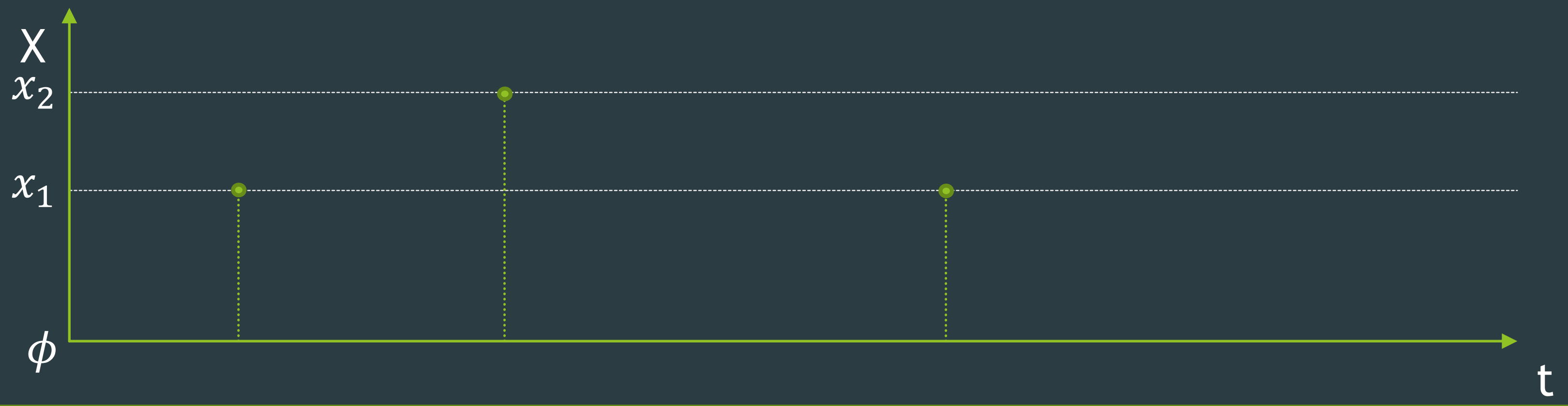
Initial State: idle

Next scheduled internal transition at time 20.00

```
__  Current Time:      20.00 __________________________________________

        EXTERNAL TRANSITION in model <system.Light>
                Input Port Configuration:
                        port <interrupt>:
                                manual
                New State: manual
                Next scheduled internal transition at time inf

        INTERNAL TRANSITION in model <system.policeman>
                New State: working
                Output Port Configuration:
                        port <output>:
                                manual
                Next scheduled internal transition at time 380.00
```

```
__  Current Time:     380.00 ____________________________________________

    EXTERNAL TRANSITION in model <system.Light>
        Input Port Configuration:
            port <interrupt>:
                auto
        New State: red
        Next scheduled internal transition at time 440.00

    INTERNAL TRANSITION in model <system.policeman>
        New State: idle
        Output Port Configuration:
            port <output>:
                auto
        Next scheduled internal transition at time 400.00
```

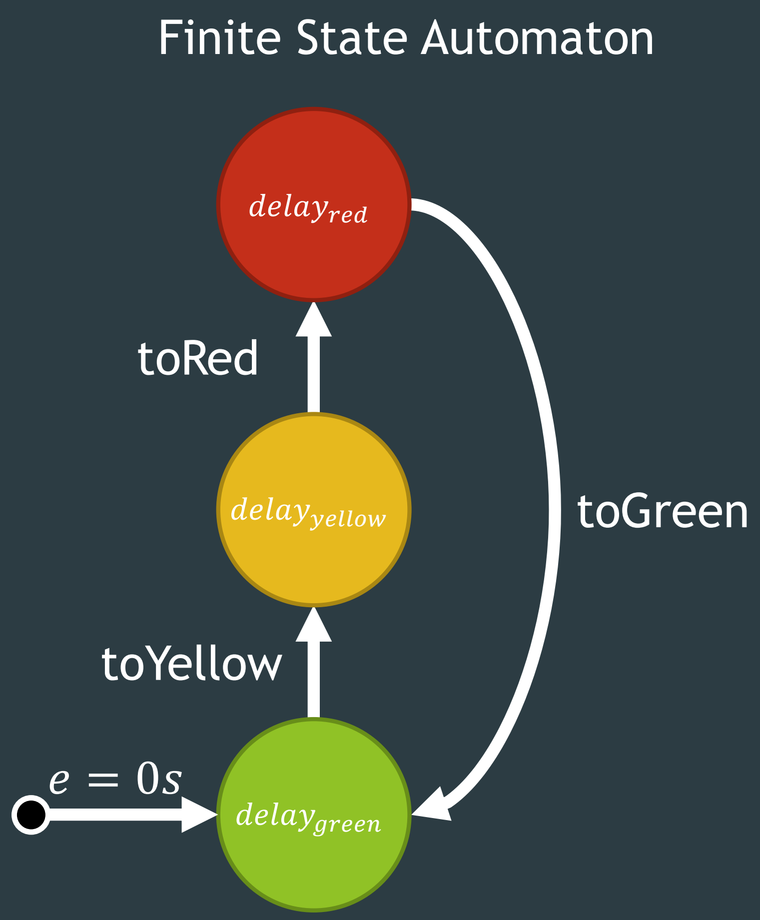
Atomic Models

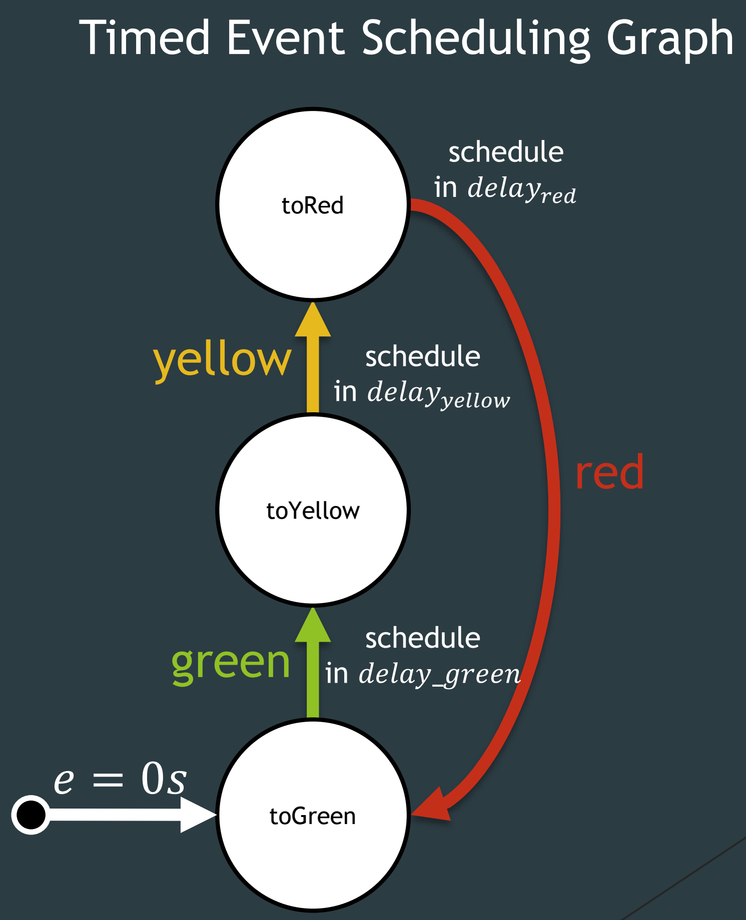

X
x_2
x_1
ϕ
t

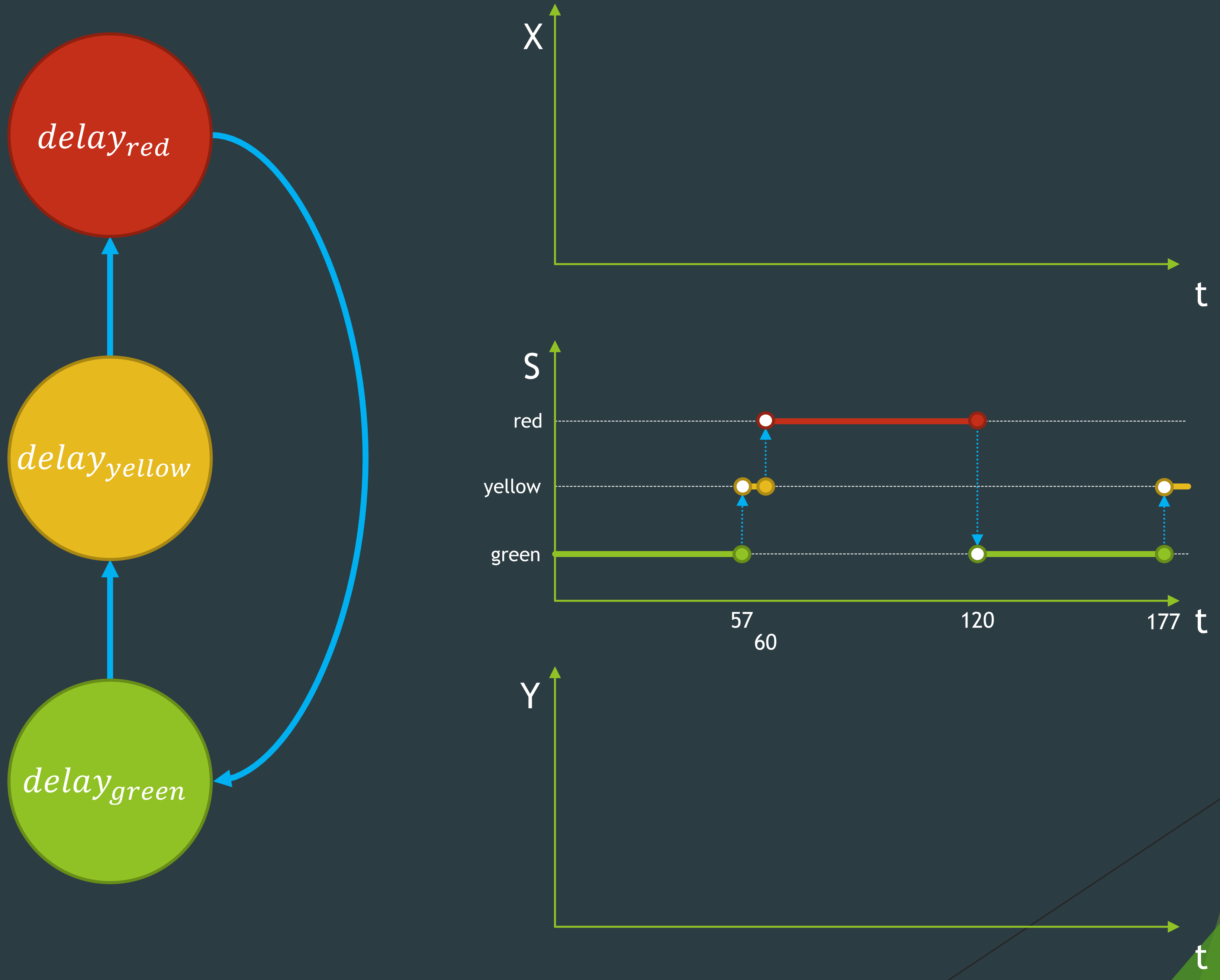
?


Y
y_2
y_1
ϕ
t

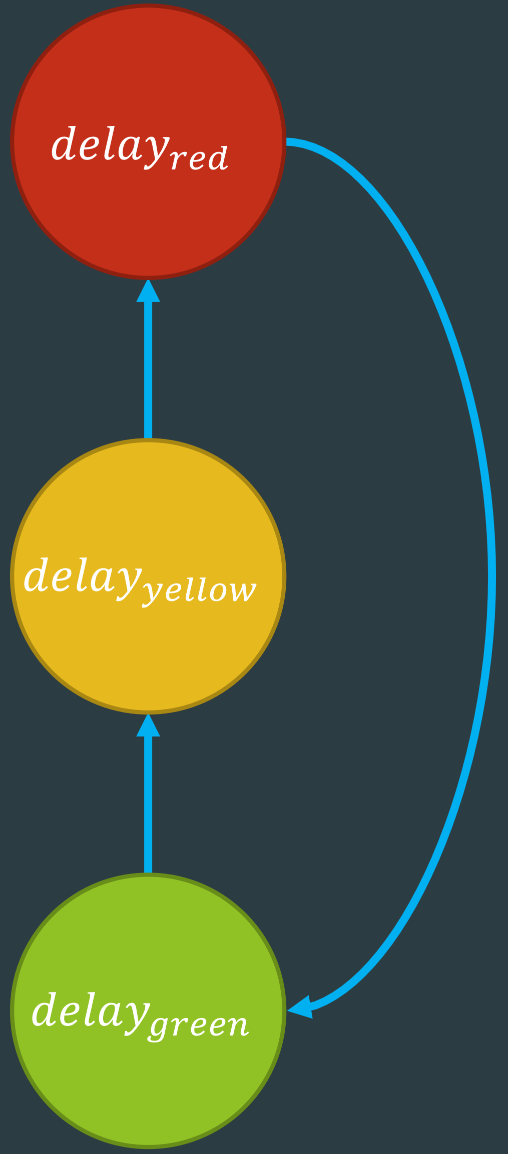
Modelling Discrete Event Behaviour







$delay_{red}$
$delay_{yellow}$
$delay_{green}$
X
t
S
red
yellow
green
57
60
120
177
t
Y
t



Autonomous (no input)

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

S : set of sequential states

S = {red, yellow, green}

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

δ_{int} = {red → green,
green → yellow,
yellow → red}

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

ta = {red → $delay_{red}$,
green → $delay_{green}$,
yellow → $delay_{yellow}$}

Time Advance: corner cases

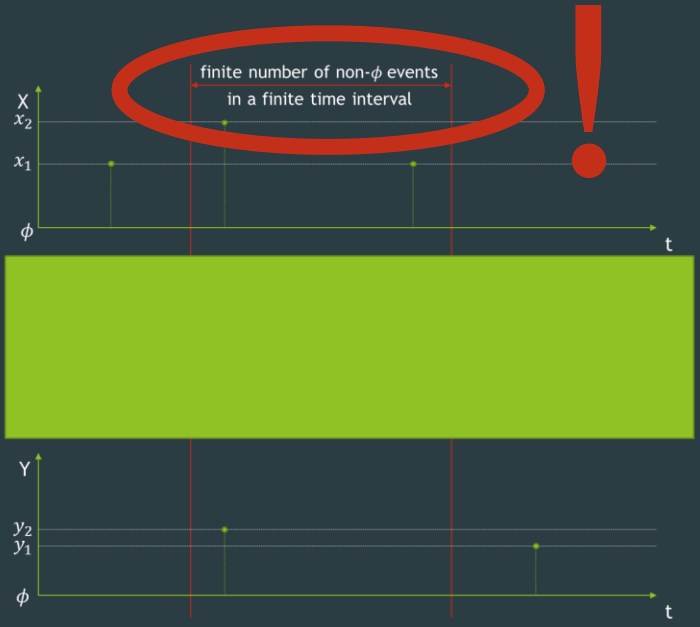
$$ta : S \rightarrow \mathbb{R}_{0,+\infty}$$

$ta(s_i) = 0$

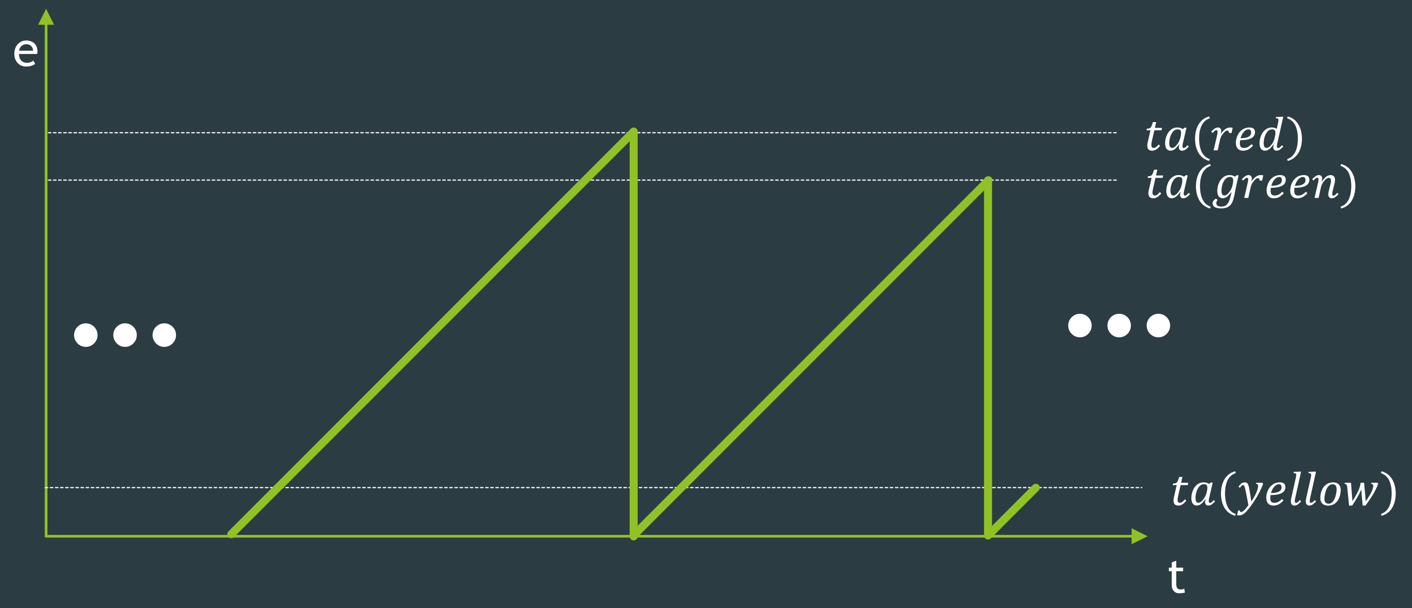
transient states

$ta(s_i) = +\infty$

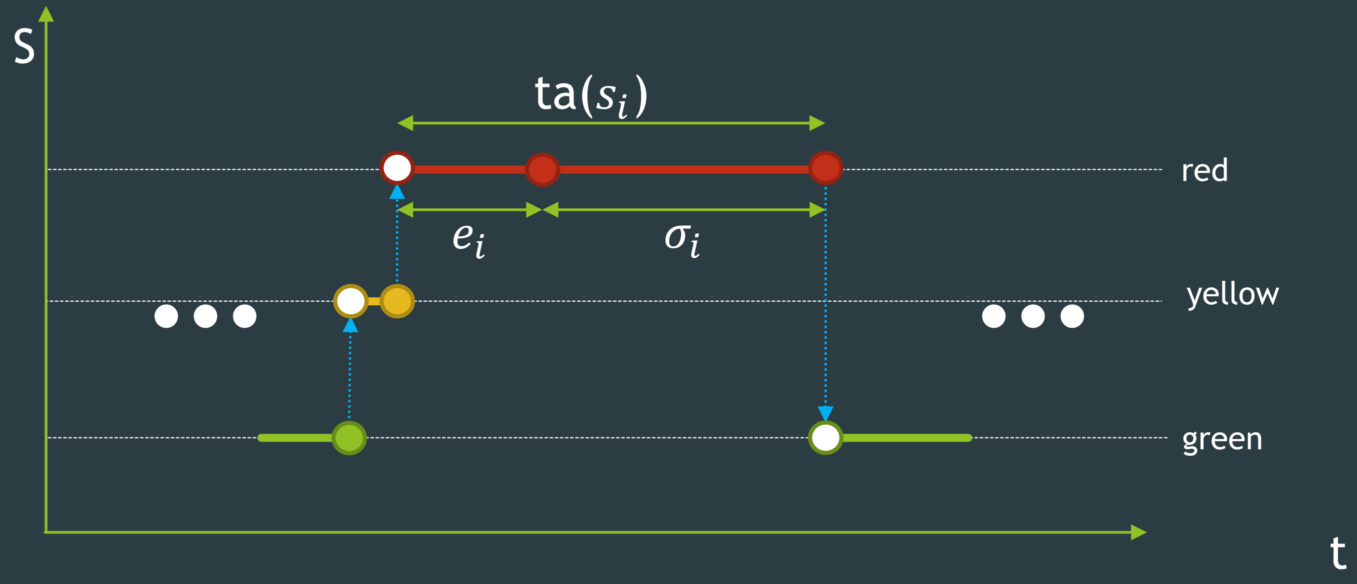
passive states



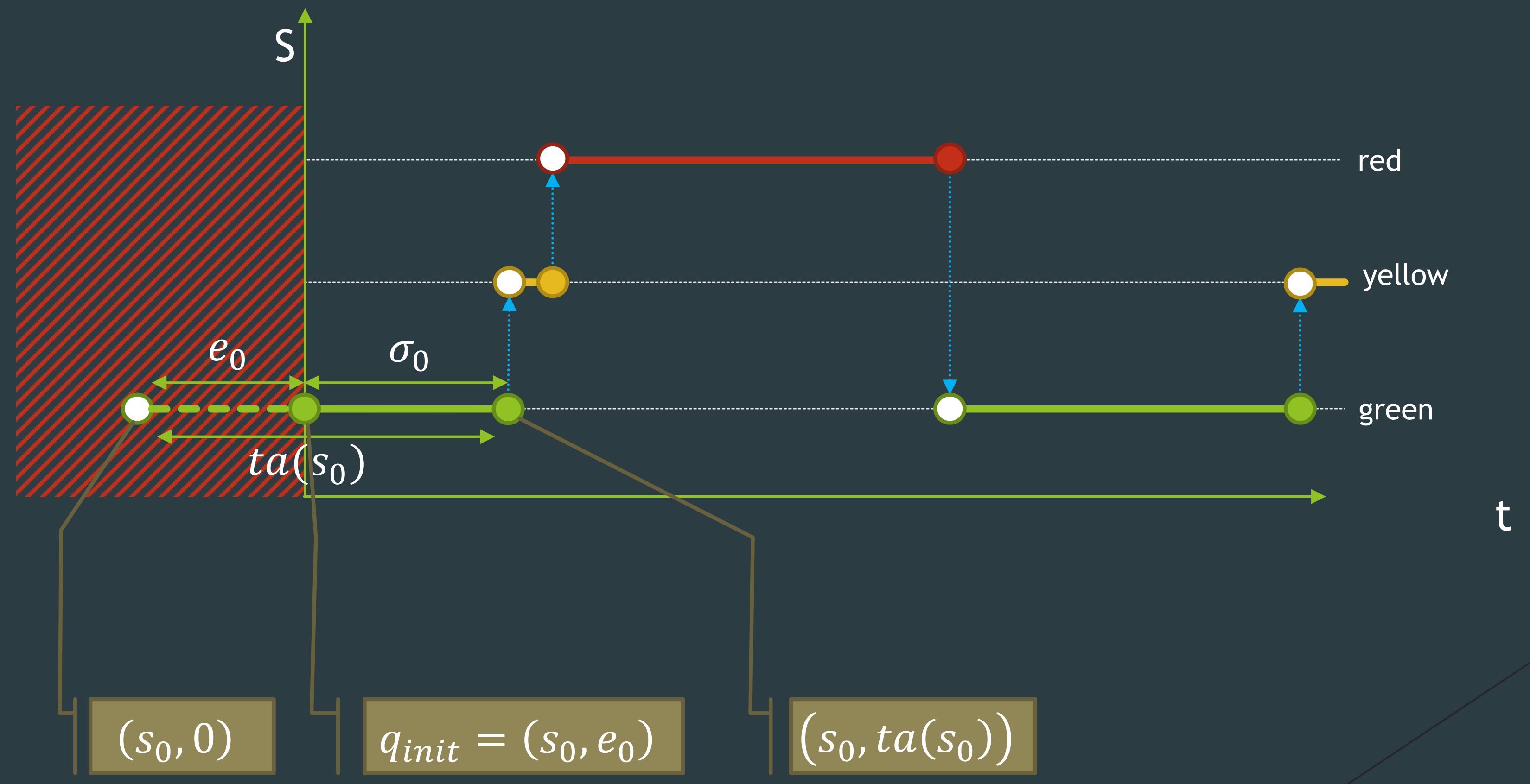
Elapsed time



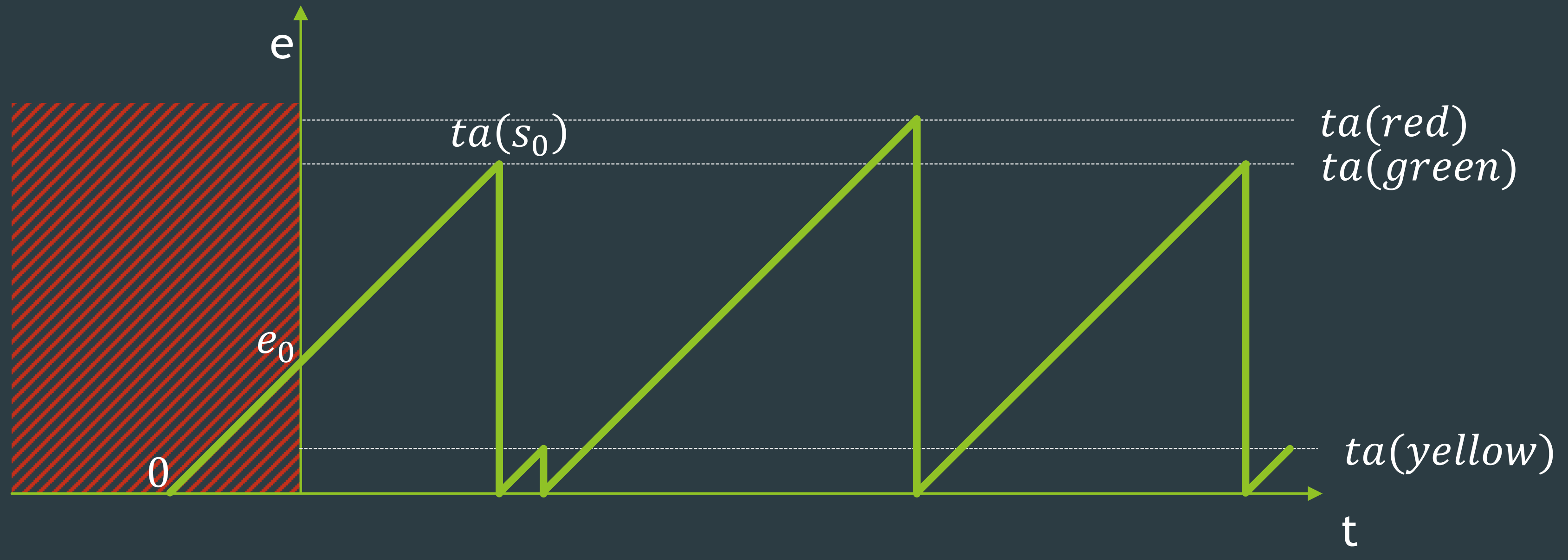
Elapsed time

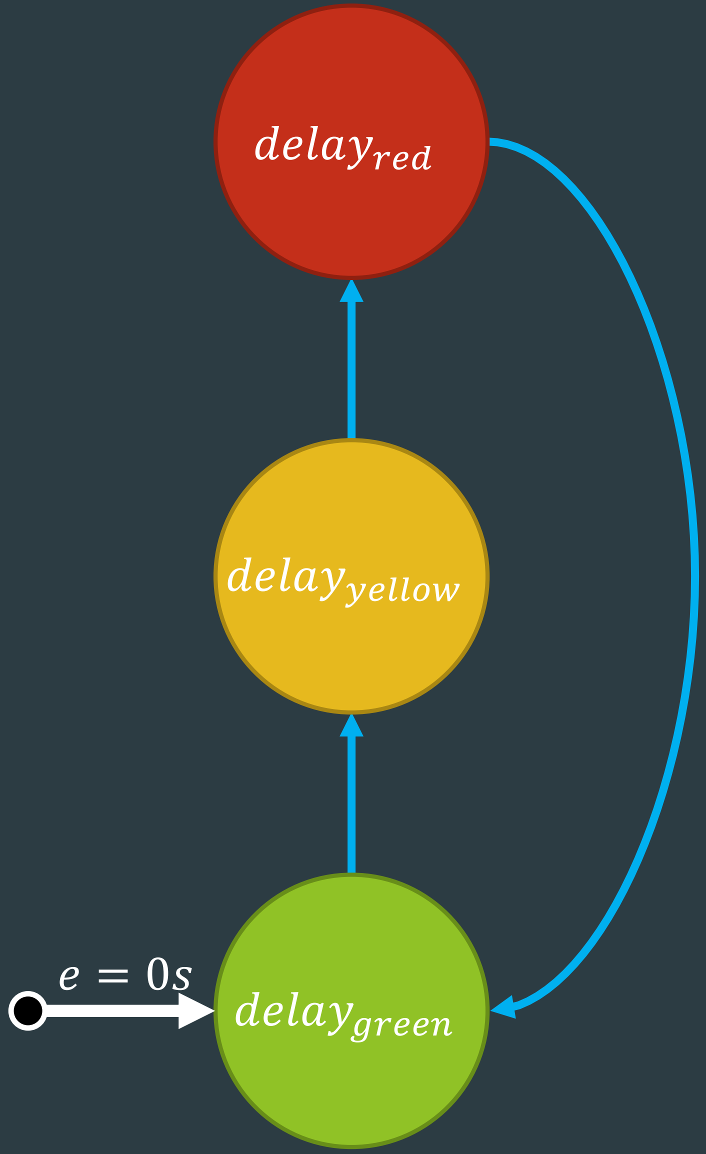


Initialization of Initial State



Elapsed time





Autonomous (no output)

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

S : set of sequential states

S = {red, yellow, green}

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

δ_{int} = {red → green,
green → yellow,
yellow → red}

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

ta = {red → $delay_{red}$,
green → $delay_{green}$,
yellow → $delay_{yellow}$}

q_{init} : Q - set of total states

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

q_{init} = (green, 0)

Abstract Syntax

S = {red, yellow, green}
δ_{int} = { red → green,
green → yellow,
yellow → red}
ta = {red → $delay_{red}$,
green → $delay_{green}$,
yellow → $delay_{yellow}$}
q_{init} = (green, 0)

Operational Semantics

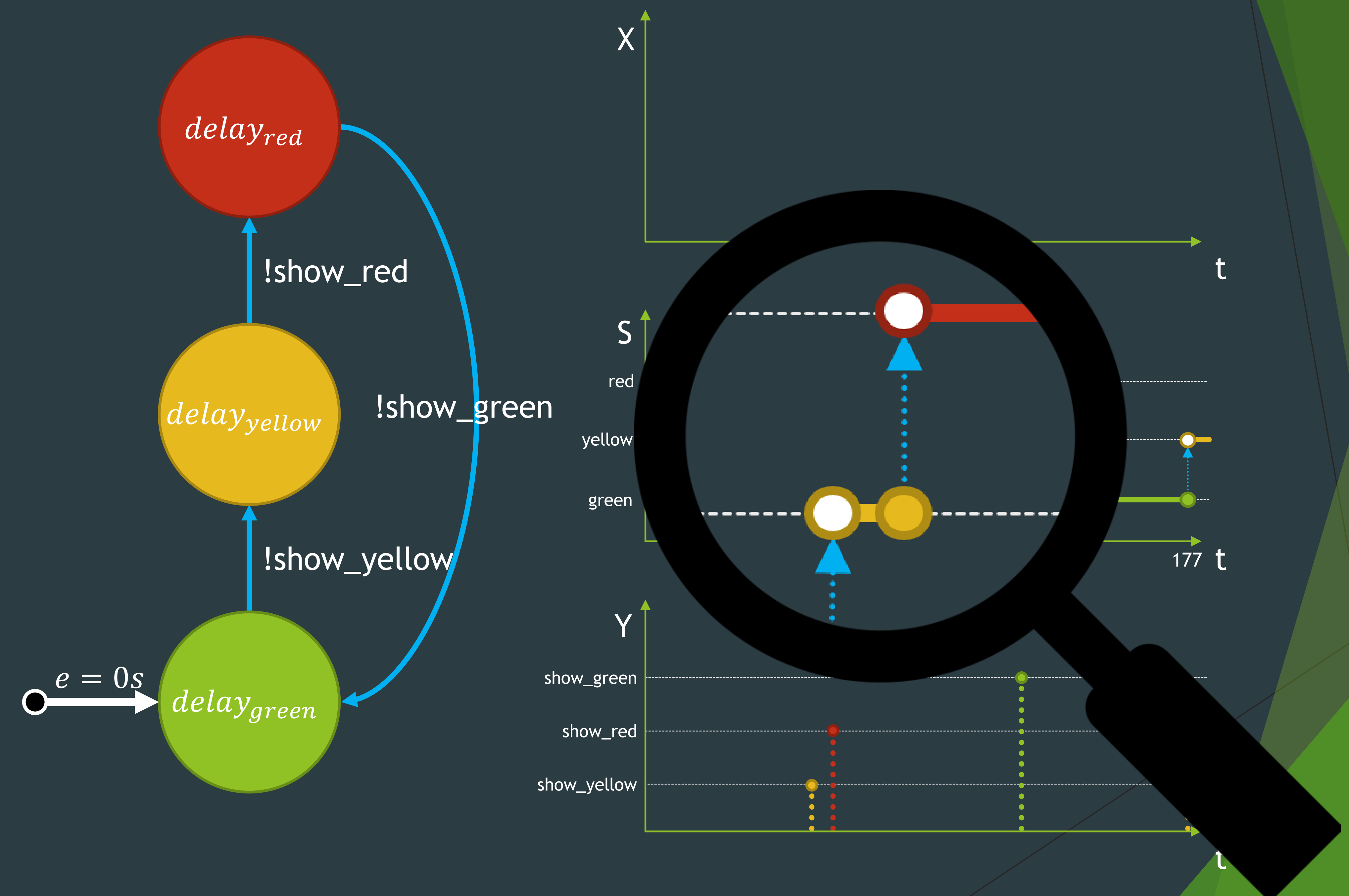
time = 0
current_state = initial_state
last_time = -initial_elapsed
while not termination_condition():
    time = last_time + ta(current_state)
    current_state = δ_{int}(current_state)
    last_time = time

Concrete Syntax

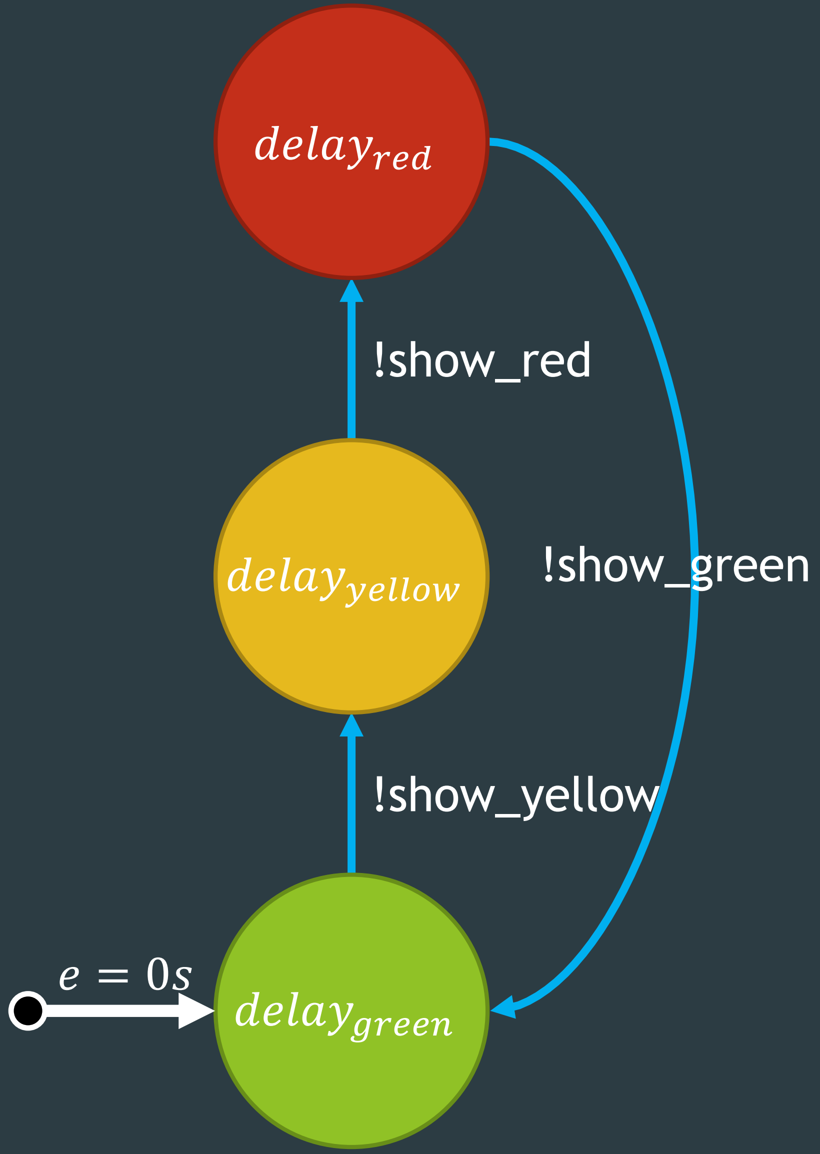
atomic_int.py

```
from pypdevs.DEVS import *

class TrafficLightAutonomous(AtomicDEVS):
    def __init__(self, delay_green,
                       delay_yellow, delay_red):
        AtomicDEVS.__init__(self, "Light")
        self.state = "green"
        self.elapsed = 0
        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
```



$delay_{red}$
!show_red
$delay_{yellow}$
!show_green
!show_yellow
$e = 0s$
$delay_{green}$
X
t
S
red
yellow
green
177
t
Y
show_green
show_red
show_yellow
t



Autonomous (with output)

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

S = {red, yellow, green}

δ_{int} = { red → green,
green → yellow,
yellow → red}

q_{init} = (green, 0)

ta = {red → $delay_{red}$,
green → $delay_{green}$,
yellow → $delay_{yellow}$}

Y : set of output events

Y = {“show_red”, “show_green”, “show_yellow”}

$\lambda : S \rightarrow Y^b$

λ = { green → [“show_yellow”],
yellow → [“show_red”],
red → [“show_green”]}

Abstract Syntax

S = {red, yellow, green}
q_{init} = (green, 0)
δ_{int} = { red → green,
green → yellow,
yellow → red}
ta = {red → $delay_{red}$,
green → $delay_{green}$,
yellow → $delay_{yellow}$}
Y = {“show_red”,
“show_green”,
“show_yellow”}
λ = {green → [“show_yellow”],
yellow → [“show_red”],
red → [“show_green”]}

Operational Semantics

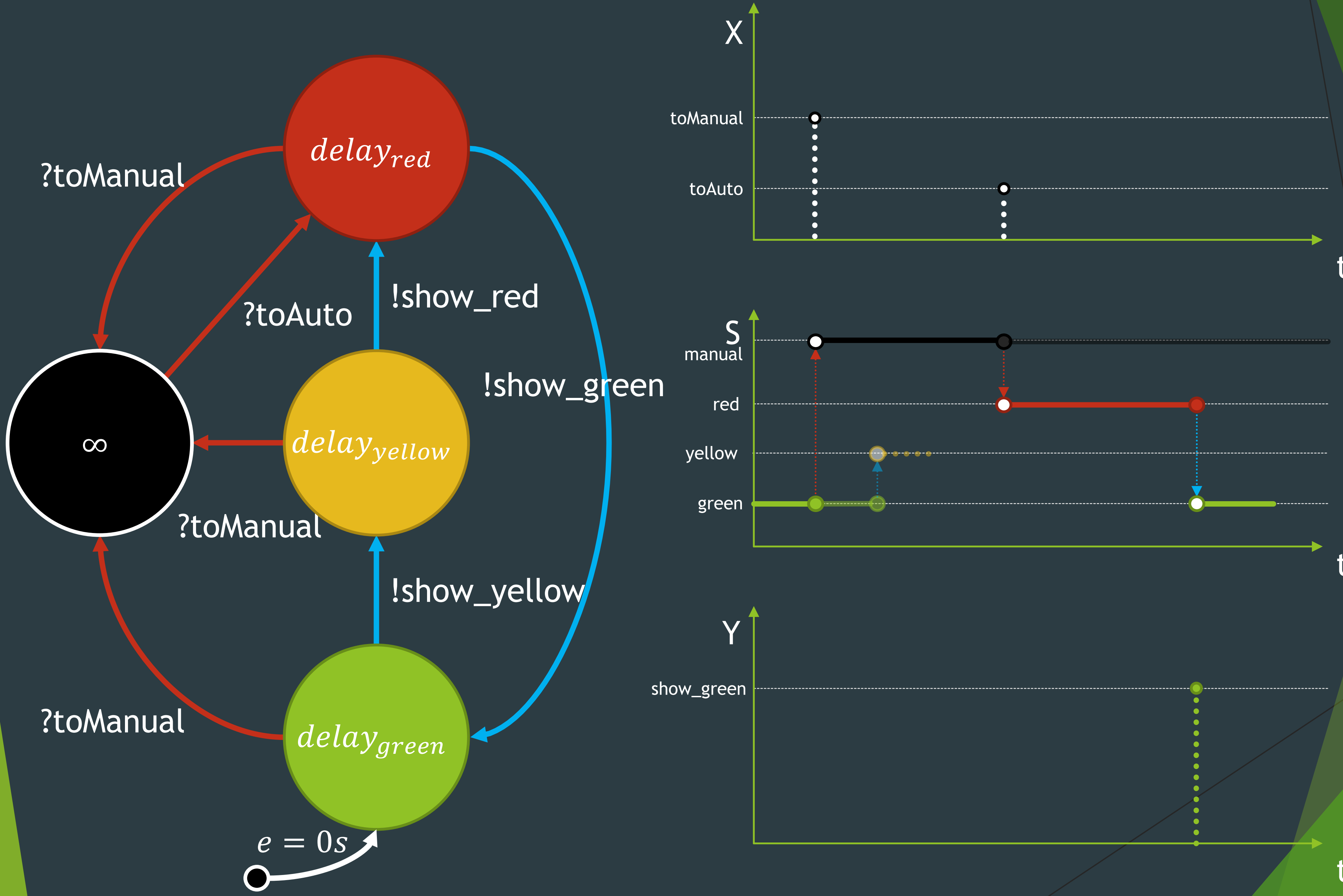
```
time = 0
current_state = initial_state
last_time = -initial_elapsed
while not termination_condition():
    time = last_time + ta(current_state)
    output(λ(current_state))
    current_state = δ_int(current_state)
    last_time = time
```

Concrete Syntax

atomic_out.py

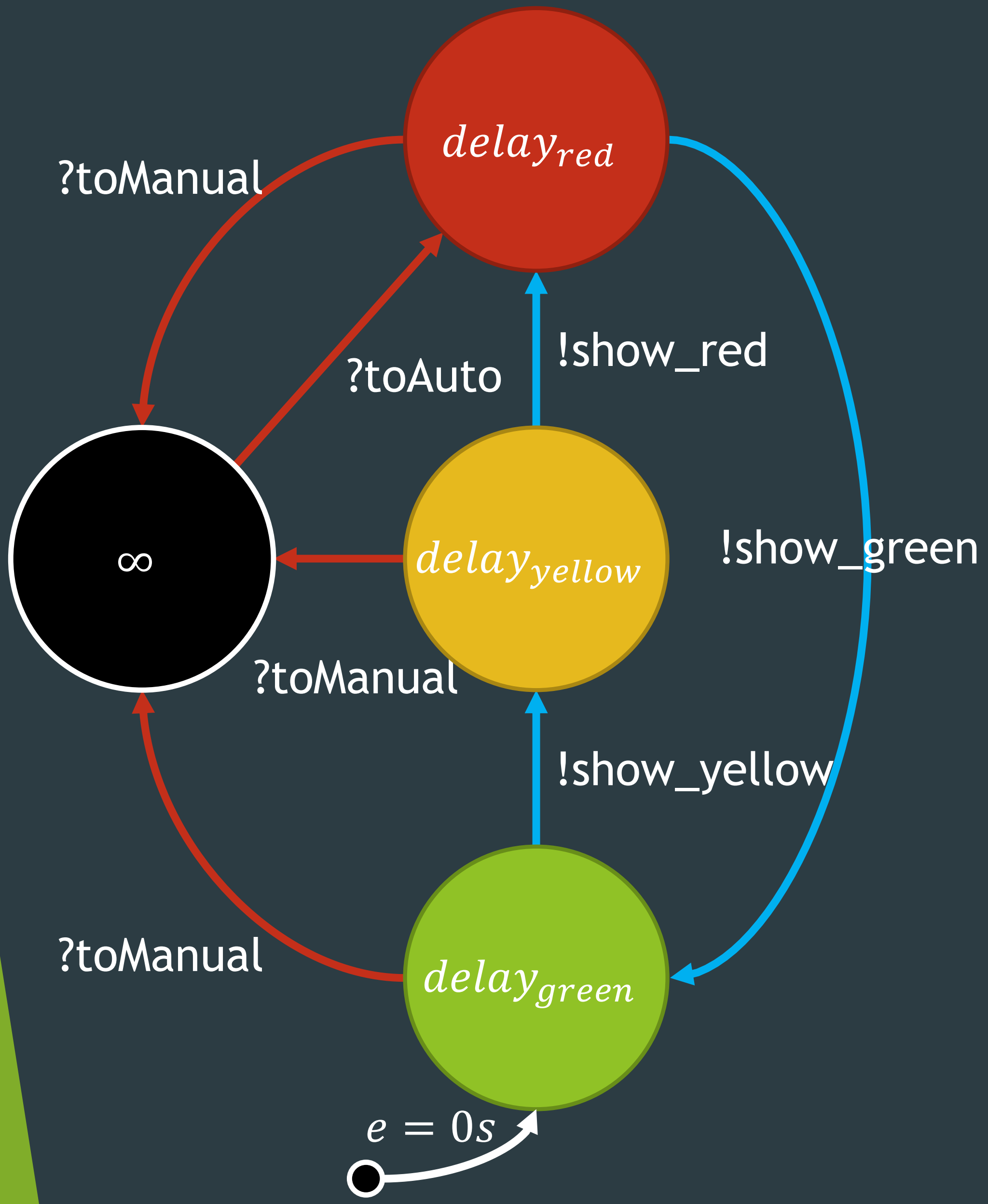
```python
from pypdevs.DEVS import *

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>:
                                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>:
                                red
                Next scheduled internal transition at time 120.00
```



?toManual
$delay_{red}$
?toAuto
!show_red
!show_green
∞
$delay_{yellow}$
?toManual
!show_yellow
?toManual
$delay_{green}$
$e = 0s$
X
toManual
toAuto
t
S
manual
red
yellow
green
t
Y
show_green
t

Reactive



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

Y = {"show_red", "show_green", "show_yellow"}
S = {red, yellow, green, manual}
q_{init} = (green, 0)
δ_{int} = {red → green,
green → yellow,
yellow → red}
λ = {green → ["show_yellow"],
yellow → ["show_red"],
red → ["show_green"]}
ta = {red → $delay_{red}$,
green → $delay_{green}$,
yellow → $delay_{yellow}$,
manual → ∞}

X : set of input events
X = {"toAuto", "toManual"}

$\delta_{ext} : Q \times X^b \rightarrow S$
$Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$
δ_{ext} = {((*, *), ["toManual"]) → "manual",
(("manual", *), ["toAuto"]) → "red"}

Abstract Syntax

Y = {“show_red”, “show_green”, “show_yellow”}
S = {red, yellow, green, manual}
q_{init} = (green, 0)
δ_{int} = {red → green,
green → yellow,
yellow → red}
λ = {green → [“show_yellow”],
yellow → [“show_red”],
red → [“show_green”]}
ta = {red → $delay_{red}$,
green → $delay_{green}$,
yellow → $delay_{yellow}$,
manual → ∞}
X = {“toAuto”, “toManual”}
δ_{ext} = {((*, *), [“toManual”]) → “manual”,
((“manual”, *), [“toAuto”]) → “red”}

Operational Semantics

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

Abstract Syntax

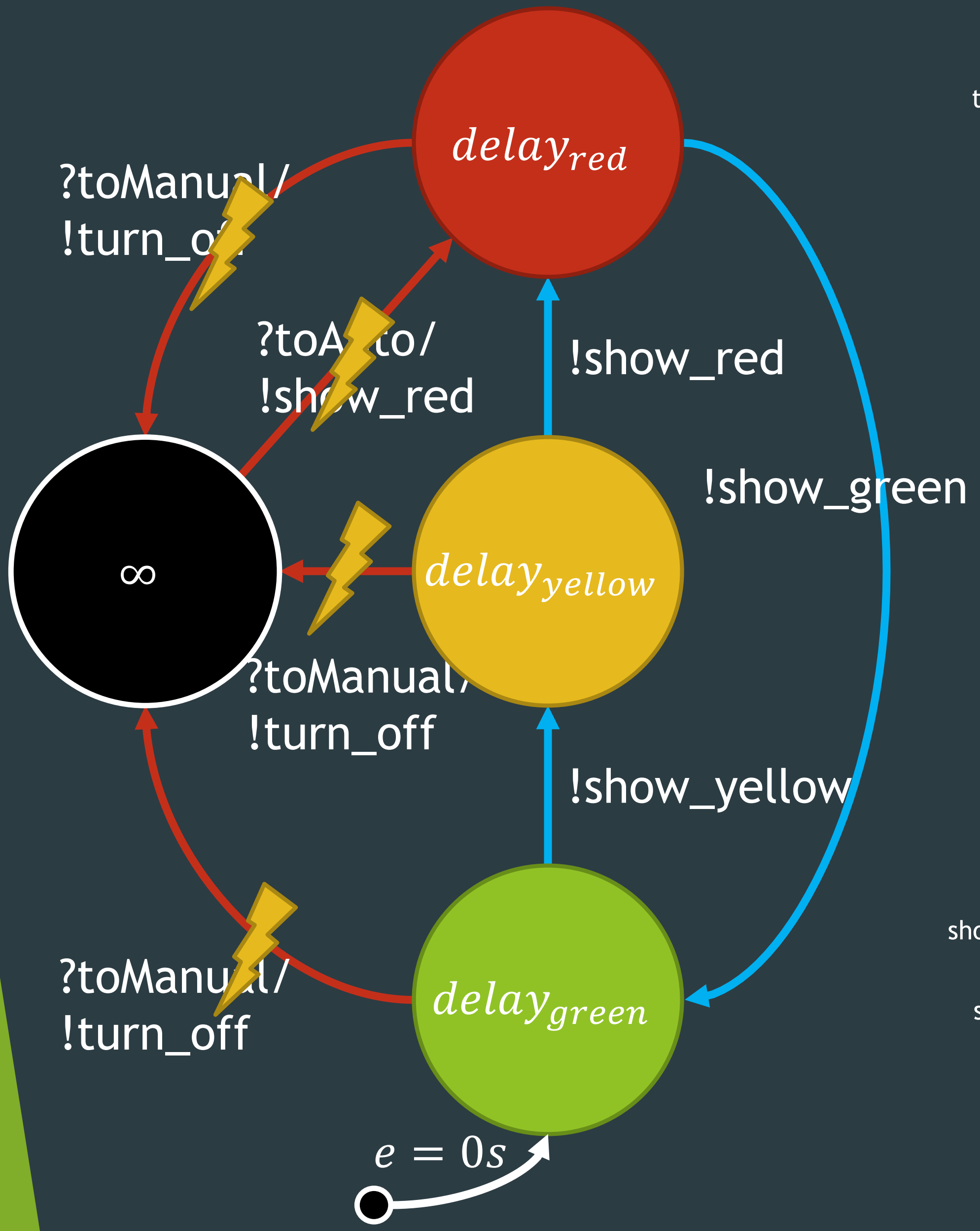
Y = {"show_red", "show_green", "show_yellow"}
S = {red, yellow, green, manual}
q_{init} = (green, 0)
δ_{int} = {red → green,
green → yellow,
yellow → red}
λ = {green → ["show_yellow"],
yellow → ["show_red"],
red → ["show_green"]}
ta = {red → $delay_{red}$,
green → $delay_{green}$,
yellow → $delay_{yellow}$,
manual → ∞}
X = {"toAuto", "toManual"}
δ_{ext} = {((*, *), ["toManual"]) → "manual",
(("manual", *), ["toAuto"]) → "red"}

Concrete Syntax

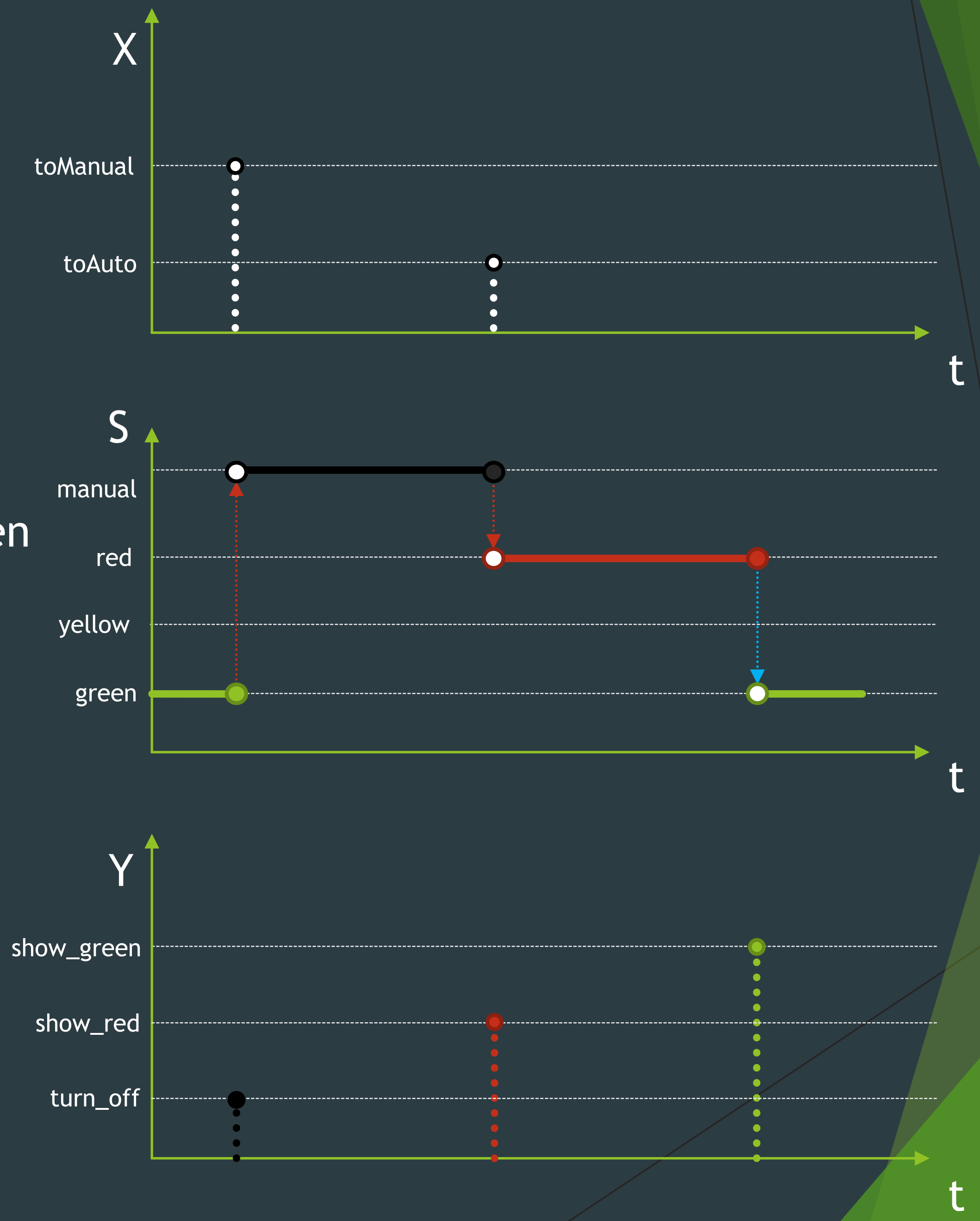
atomic_ext.py

```
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][0]
      if inp == "toManual":
         return "manual"
      elif inp == "toAuto":
         if self.state == "manual":
            return "red"
```

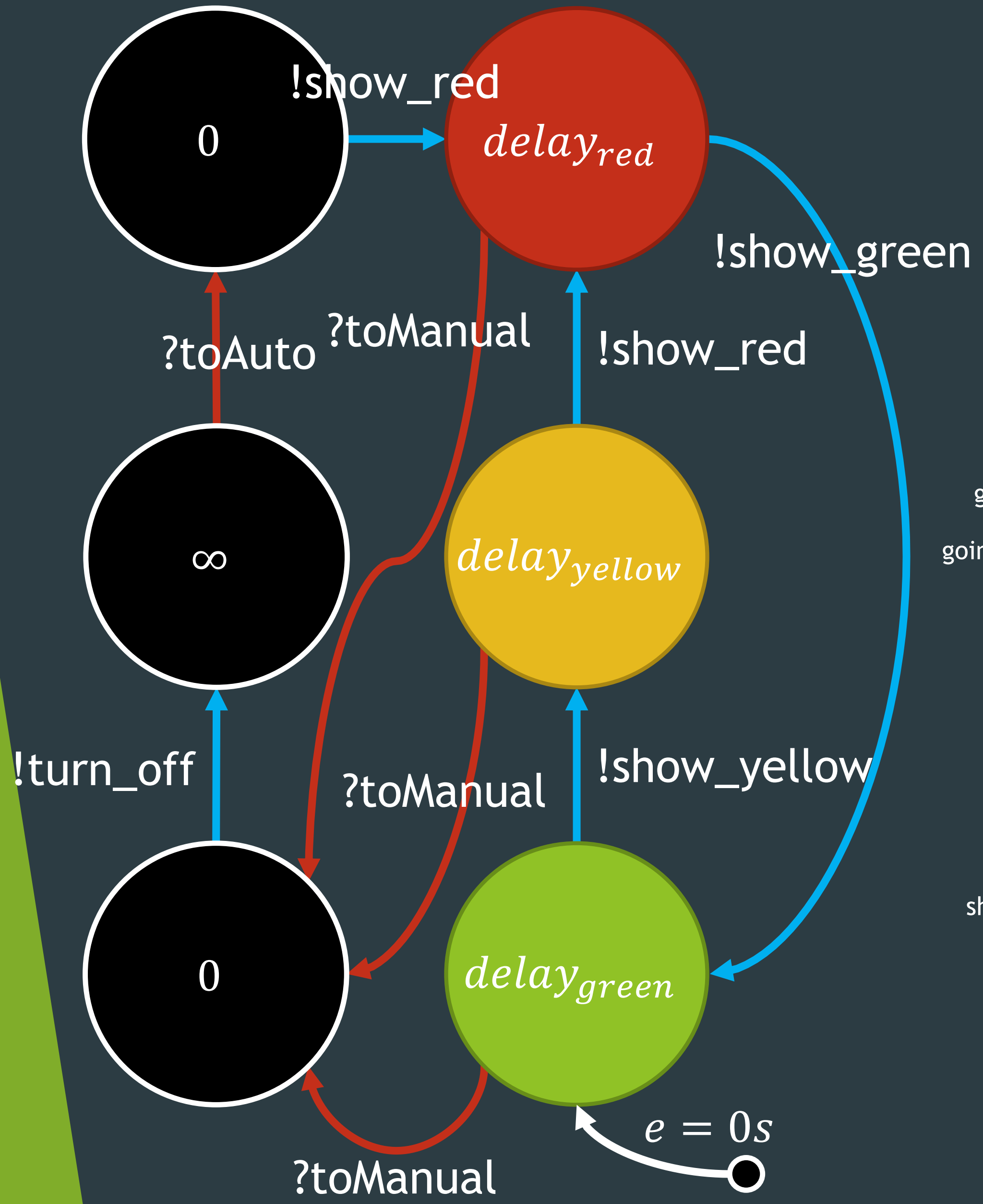
```
__ Current Time:      0.00 ____________________________________________

    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>:
                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>:
                red
        Next scheduled internal transition at time 120.00
```

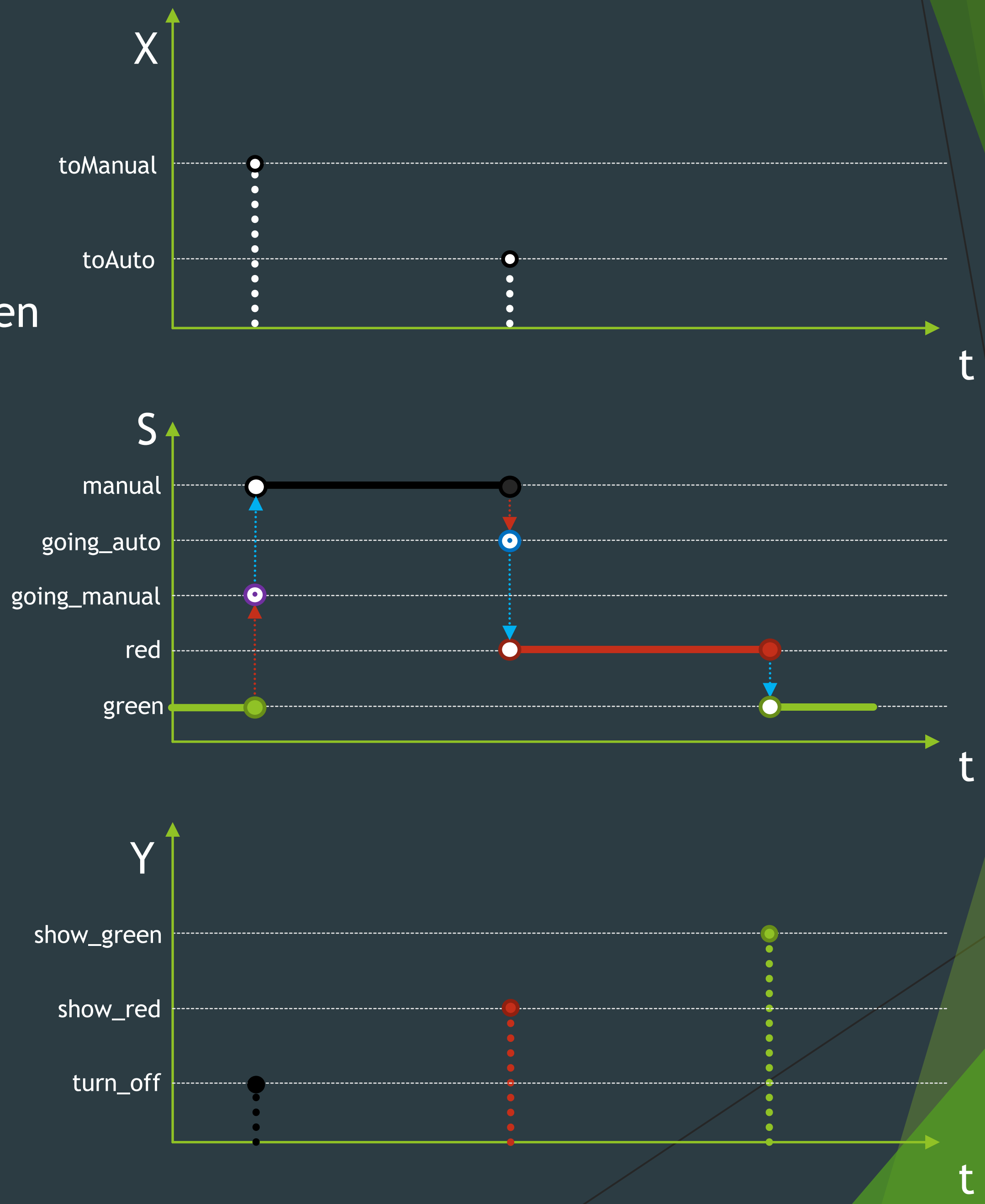


$delay_{red}$
?toManual/
!turn_off
?toAuto/
!show_red
!show_red
!show_green
∞
$delay_{yellow}$
?toManual/
!turn_off
!show_yellow
?toManual/
!turn_off
$delay_{green}$
$e = 0s$



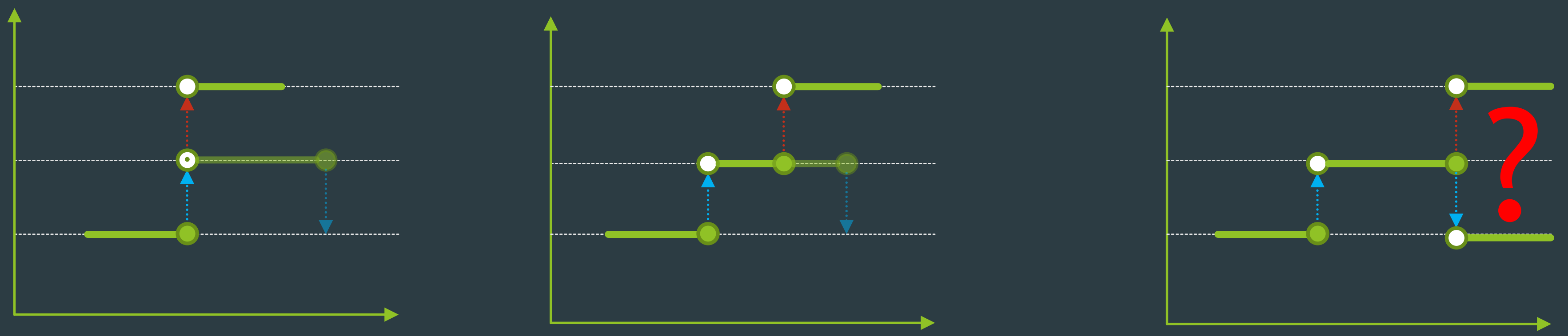
X
toManual
toAuto
t
S
manual
red
yellow
green
t
Y
show_green
show_red
turn_off
t


0
!show_red
$delay_{red}$
!show_green
?toAuto
?toManual
!show_red
∞
$delay_{yellow}$
!turn_off
?toManual
!show_yellow
0
$delay_{green}$
?toManual
$e = 0s$


X
toManual
toAuto
t
S
manual
going_auto
going_manual
red
green
t
Y
show_green
show_red
turn_off
t

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

$e = 0$ | $0 < e < ta(s)$ | $e = ta(s)$



$$\delta_{conf} : S \times X^b \to S$$

atomic_conf.py

```
from pypdevs.DEVS import *

class TrafficLight(AtomicDEVS):
    …

    def confTransition(self, inputs):
        self.elapsed = 0.0
        self.state = self.intTransition()
        self.state = self.extTransition(inputs)
        return self.state
```

Full Atomic DEVS Specification

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

X : set of input events

Y : set of output events

S : set of sequential states

$q_{init} : Q$

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

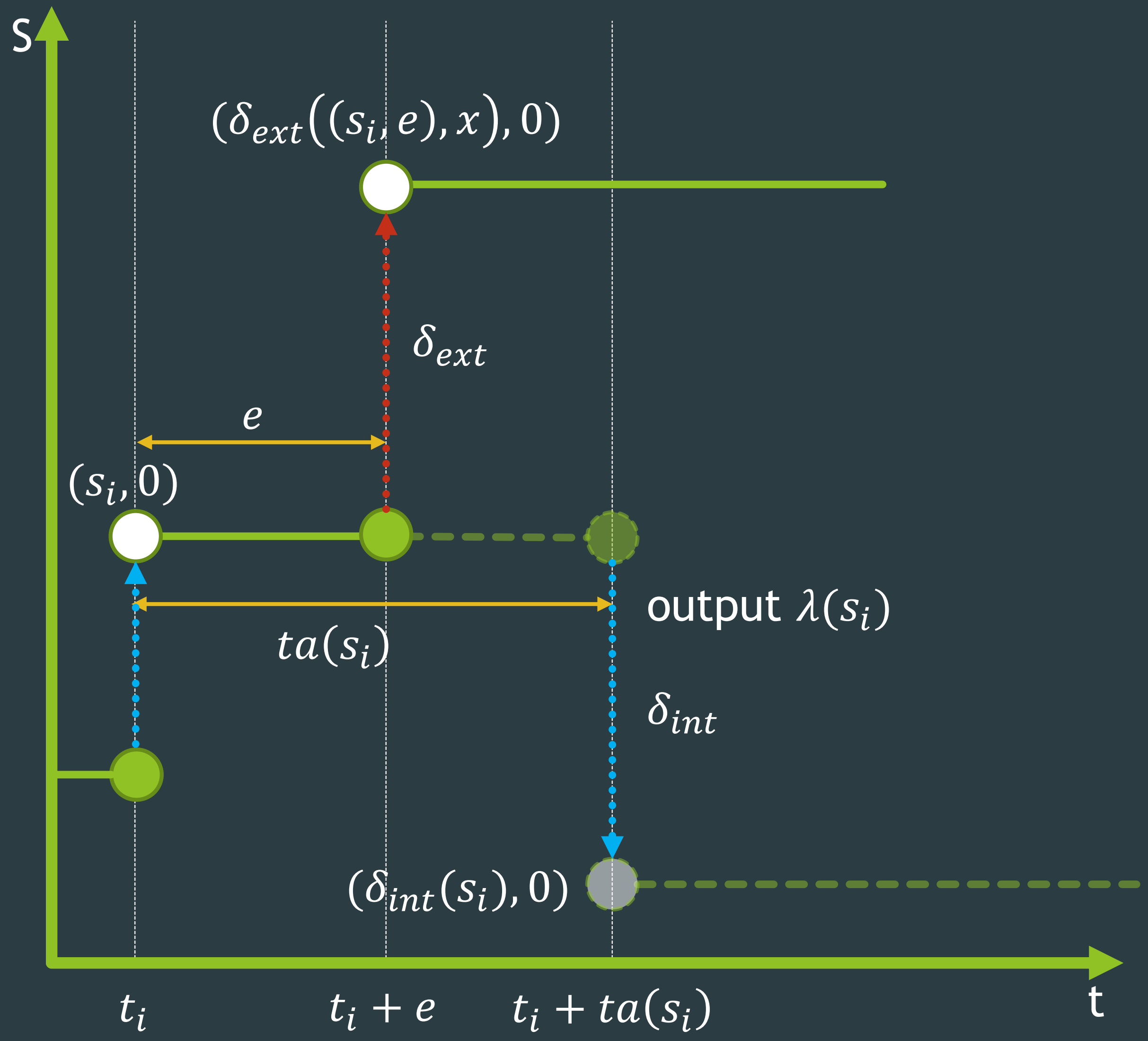
$\delta_{int} : S \rightarrow S$

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

$\delta_{conf} : S \times X^b \rightarrow S$

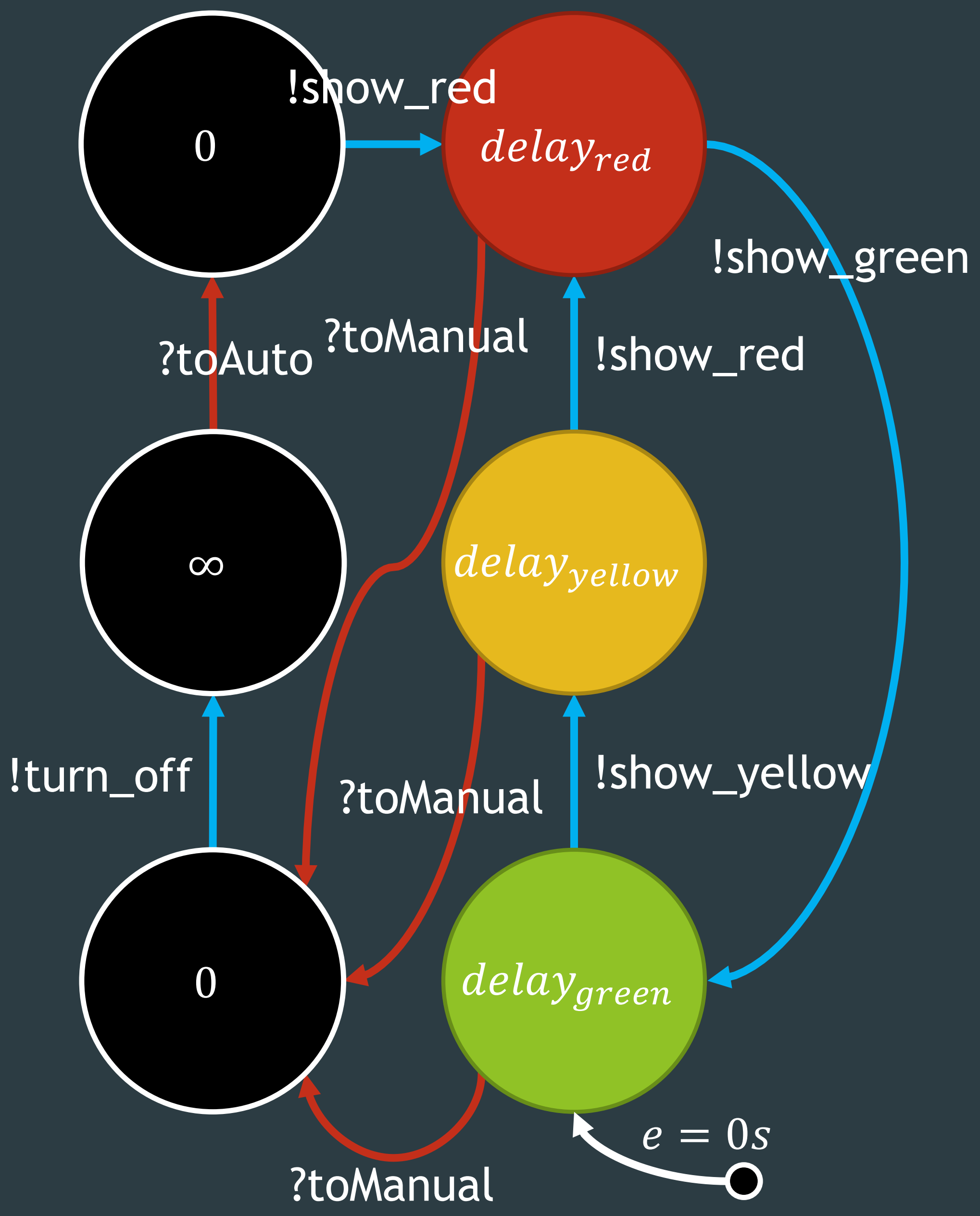
$\lambda : S \rightarrow Y^b$

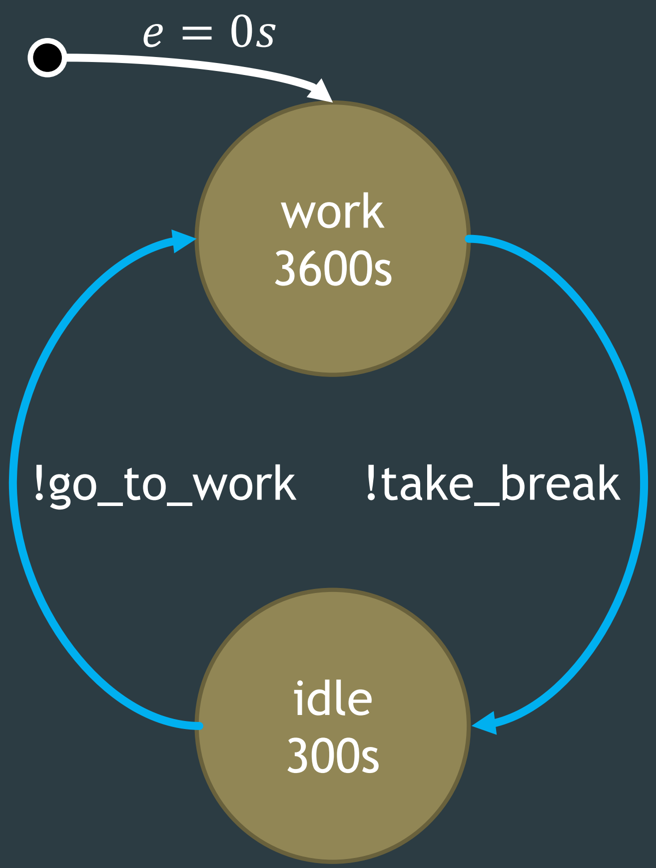
$ta : S \rightarrow \mathbb{R}^{+}_{0,+\infty}$



S
$(\delta_{ext}((s_i, e), x), 0)$
δ_{ext}
e
$(s_i, 0)$
output $\lambda(s_i)$
$ta(s_i)$
δ_{int}
$(\delta_{int}(s_i), 0)$
t_i
$t_i + e$
$t_i + ta(s_i)$
t

Coupled Models


0
!show_red
delay_red
!show_green
?toManual
?toAuto
!show_red
∞
delay_yellow
!turn_off
?toManual
!show_yellow
0
delay_green
e = 0s
?toManual



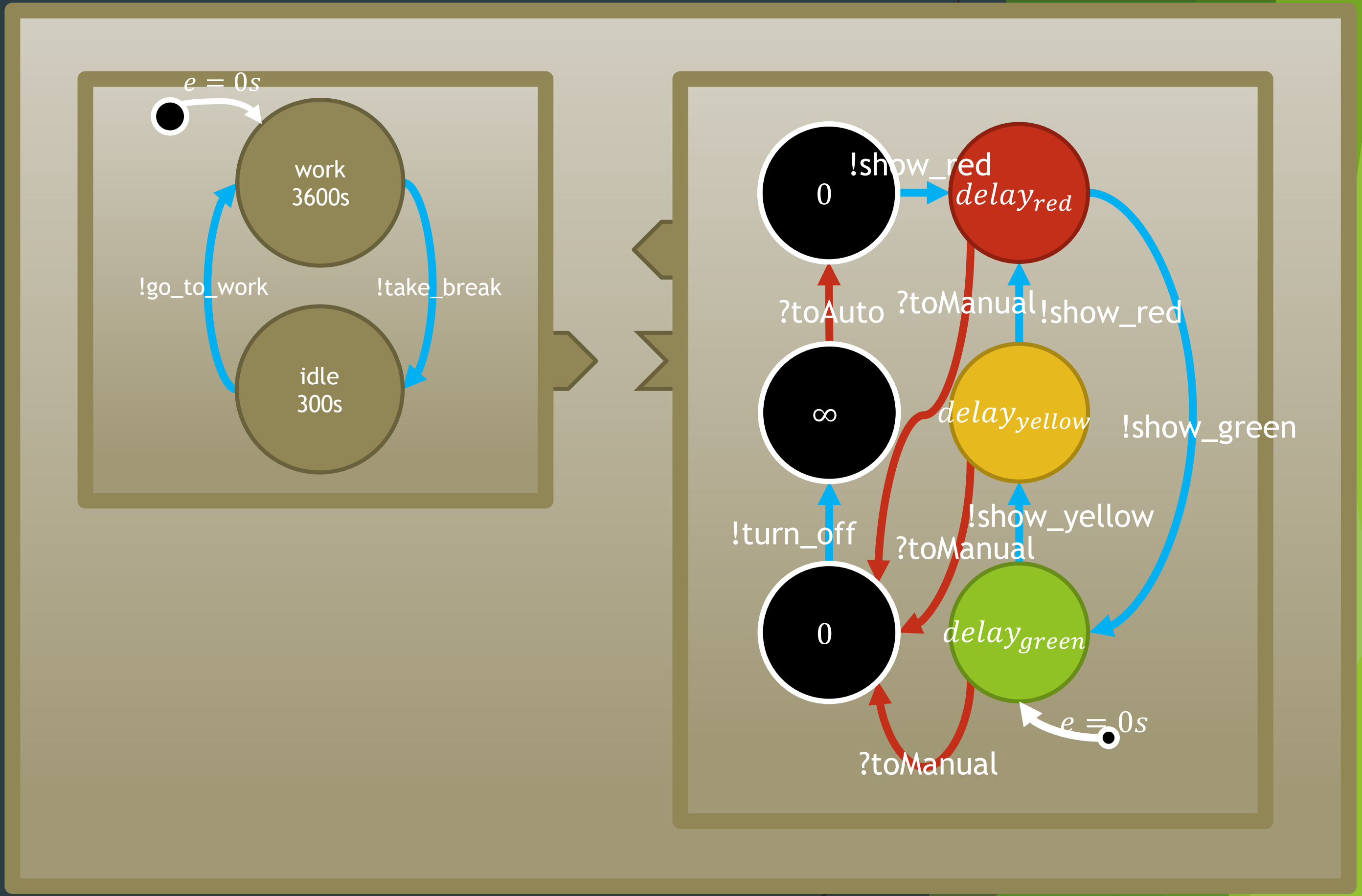
$e = 0s$
work
3600s
!go_to_work
!take_break
idle
300s

TRAFFIC

$$C = \langle D, \{M_i\} \rangle$$

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

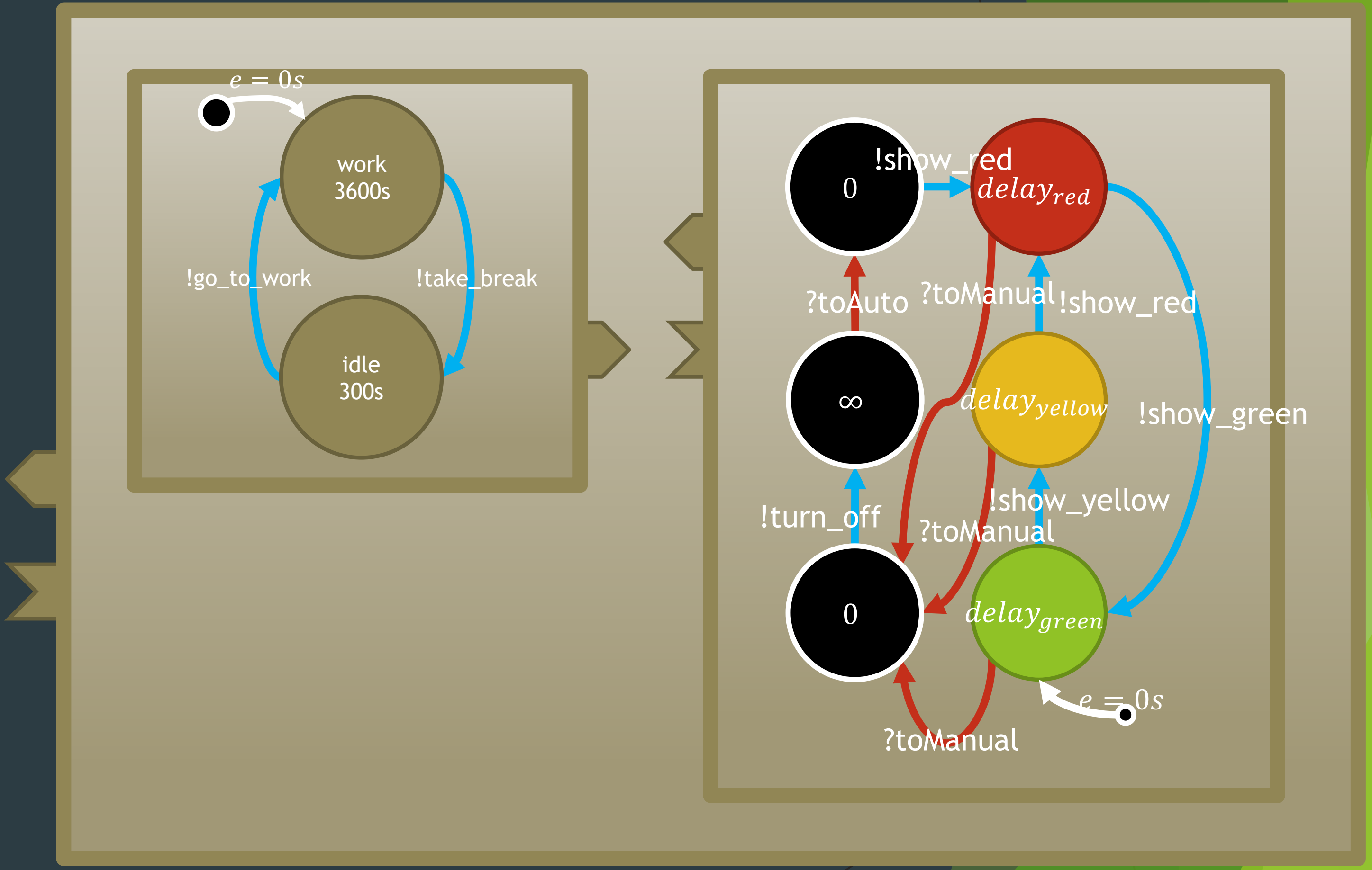
$$M_i = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle, \forall\, i \in D$$



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

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

$$M_i = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle, \forall\ i \in D$$



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

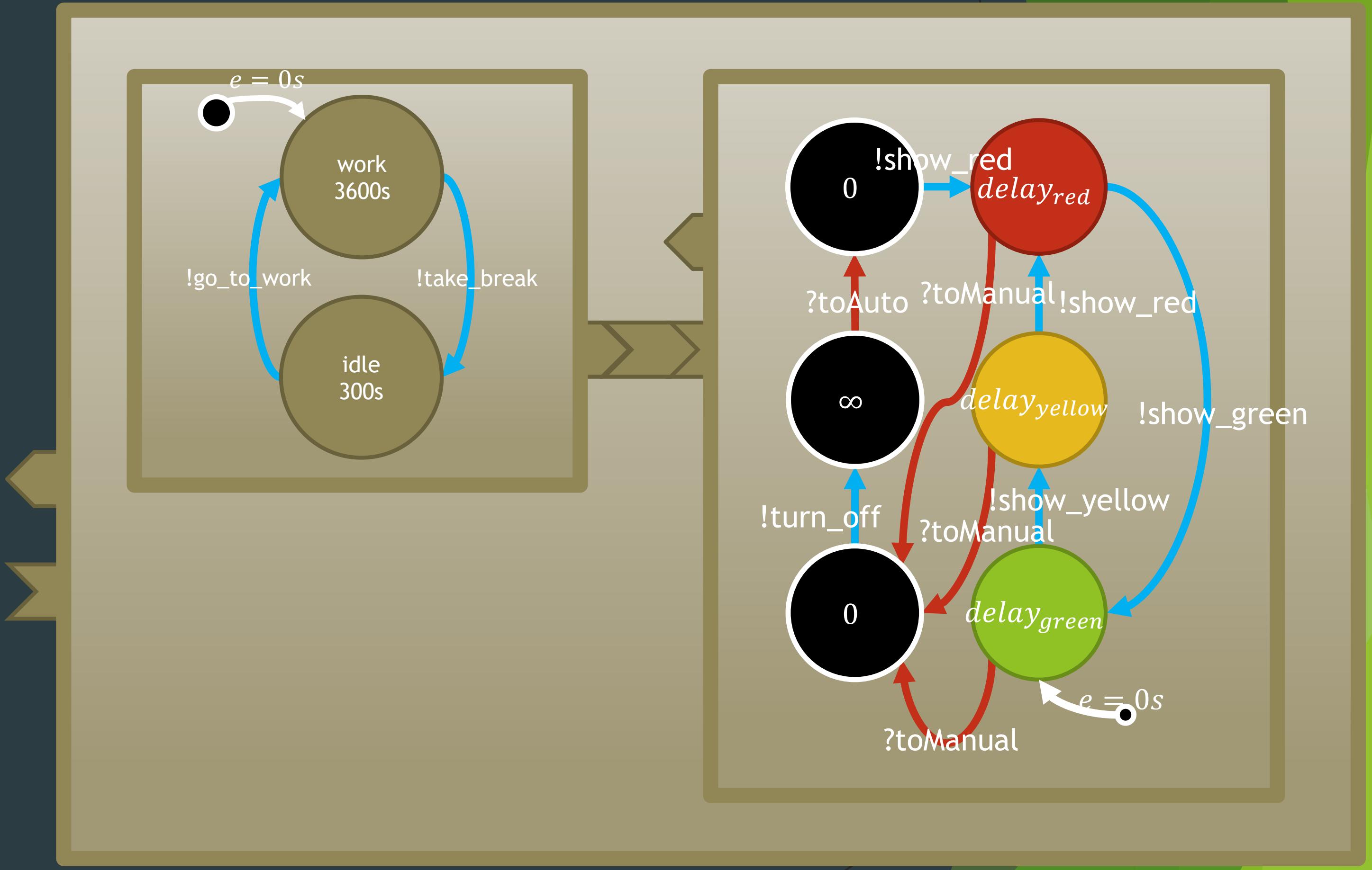
$$\{M_i | i \in D\}$$

$$M_i = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle, \forall\, i \in D$$

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

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

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

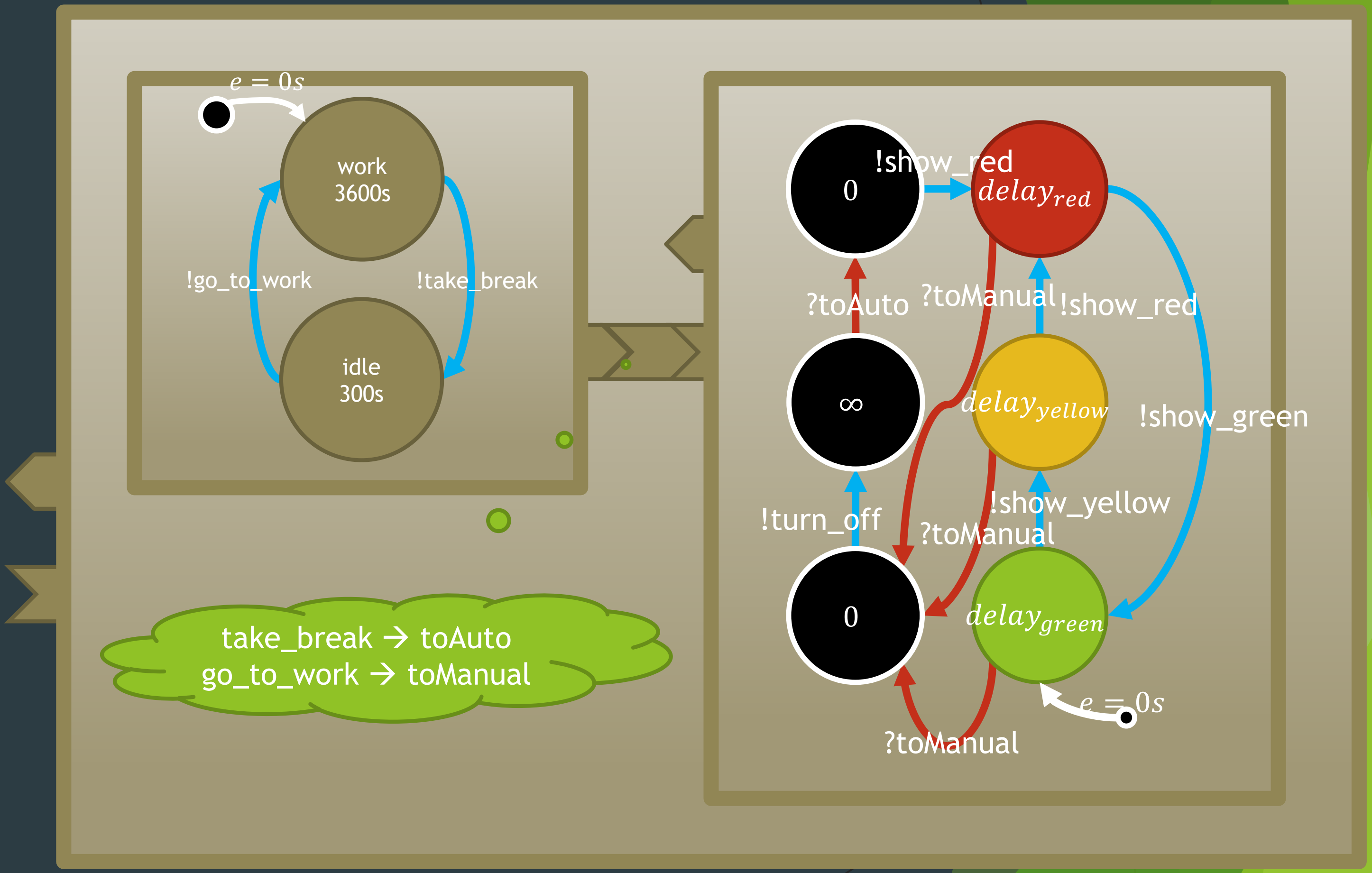


$$C = \langle X_{self}, Y_{self}, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} \rangle$$

$$\{M_i | i \in D\}$$
$$M_i = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle, \forall\, i \in D$$

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

$$\{Z_{i,j} | i \in D \cup \{self\}, j \in I_i\}$$
$$Z_{self,j} : X_{self} \rightarrow X_j, \forall\, j \in D$$
$$Z_{i,self} : Y_i \rightarrow Y_{self}, \forall\, i \in D$$
$$Z_{i,j} : Y_i \rightarrow X_j, \forall\, i,j \in D$$



Concrete Syntax

trafficlight_system.py

```
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)
```

__ Current Time: 0.00 ____

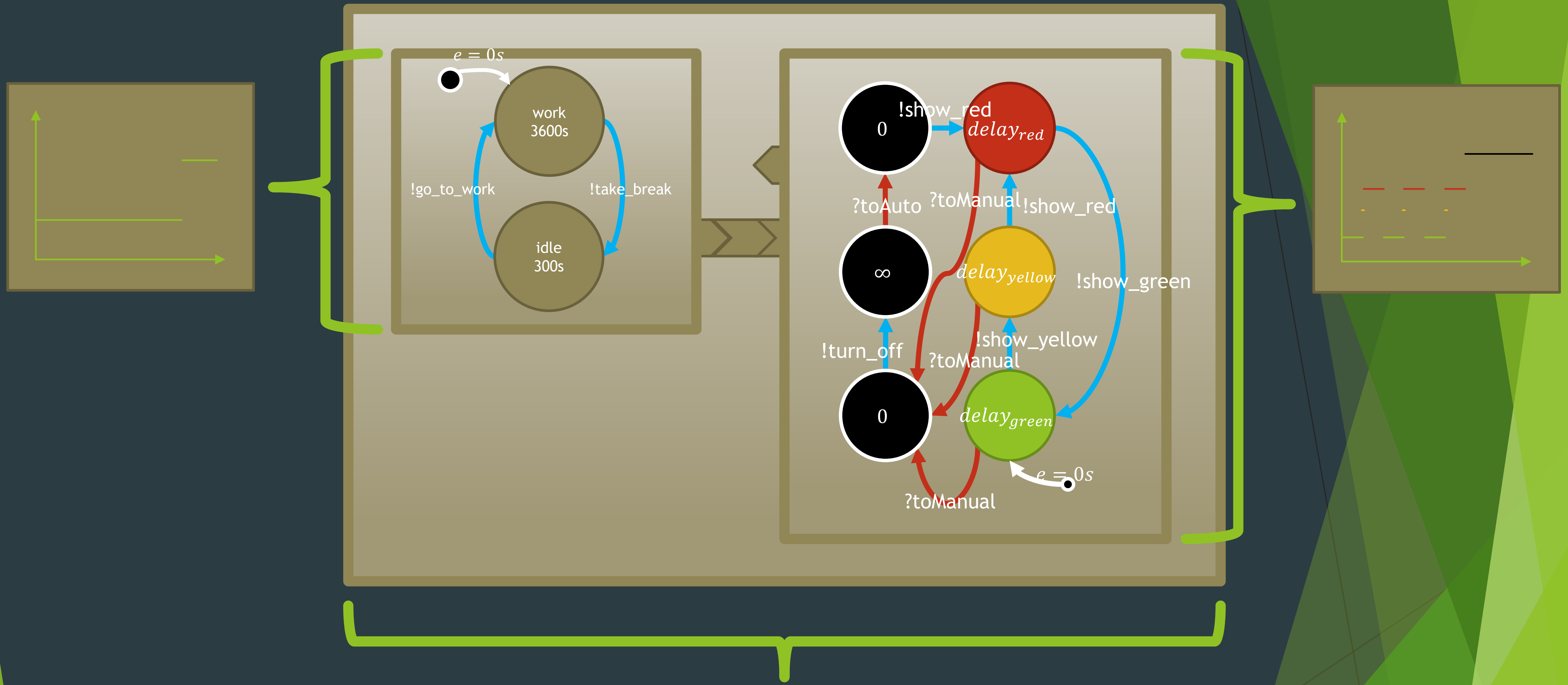
INITIAL CONDITIONS in model <system.Light>
Initial State: green
Next scheduled internal transition at time 57.00

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>:
                                manual
                New State: manual
                Next scheduled internal transition at time inf

        INTERNAL TRANSITION in model <system.policeman>
                New State: working
                Output Port Configuration:
                        port <output>:
                                manual
                Next scheduled internal transition at time 380.00
```

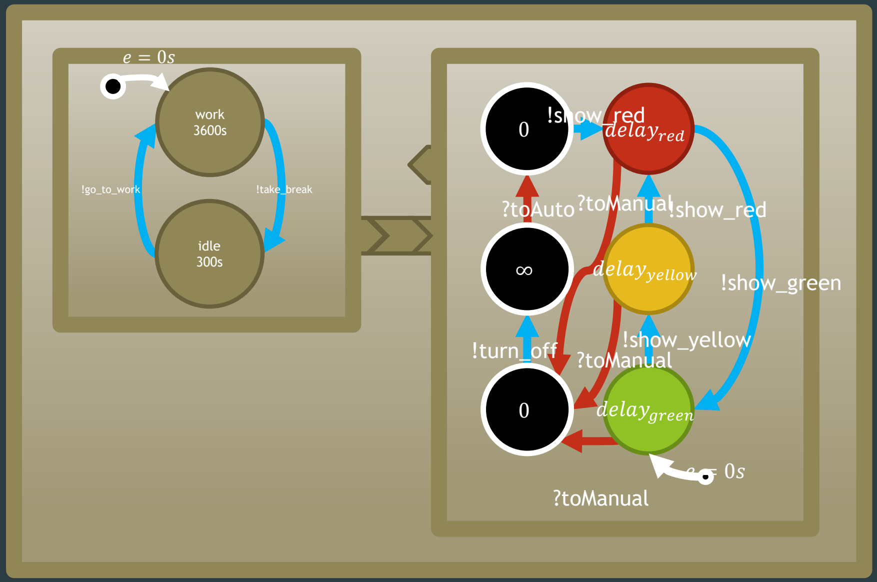
```
__  Current Time:     380.00 ____________________________________________

    EXTERNAL TRANSITION in model <system.Light>
        Input Port Configuration:
            port <interrupt>:
                auto
        New State: red
        Next scheduled internal transition at time 440.00

    INTERNAL TRANSITION in model <system.policeman>
        New State: idle
        Output Port Configuration:
            port <output>:
                auto
        Next scheduled internal transition at time 400.00
```

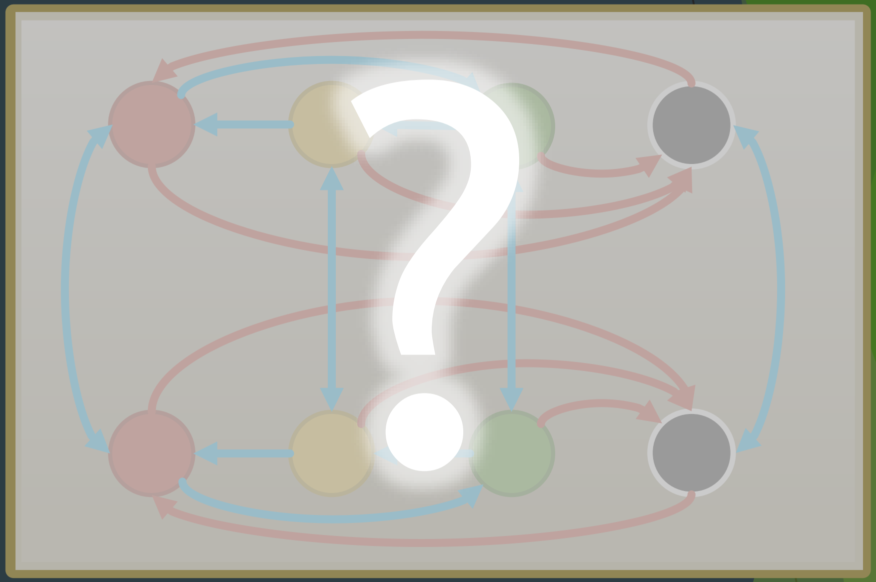
Closure under Coupling



e = 0s
work
3600s
!go_to_work
!take_break
idle
300s
0
!show_red
delay_red
?toAuto
?toManual
!show_red
∞
delay_yellow
!show_green
!turn_off
!show_yellow
?toManual
0
delay_green
e = 0s
?toManual

?

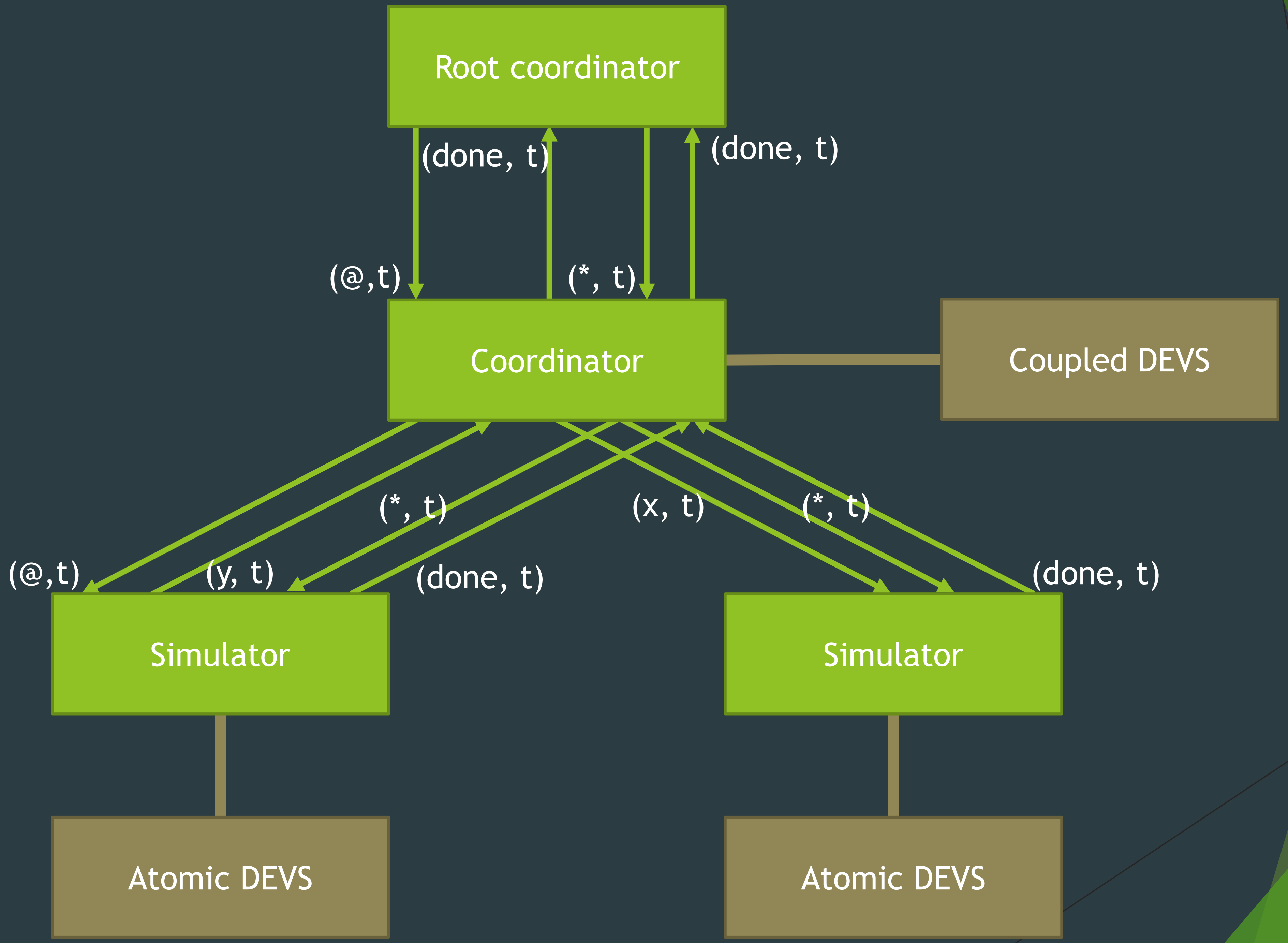




$$CM = \langle X_{self}, Y_{self}, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} \rangle$$
$$M_i = \langle X_i, Y_i, S_i, \delta_{int,i}, \delta_{ext,i}, \lambda_i, ta_i \rangle, \forall\, i \in D$$

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

Hierarchical Simulator


Root coordinator
(done, t)
(done, t)
(@,t)
(*, t)
Coordinator
Coupled DEVS
(*, t)
(x, t)
(*, t)
(@,t)
(y, t)
(done, t)
(done, t)
Simulator
Simulator
Atomic DEVS
Atomic DEVS

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

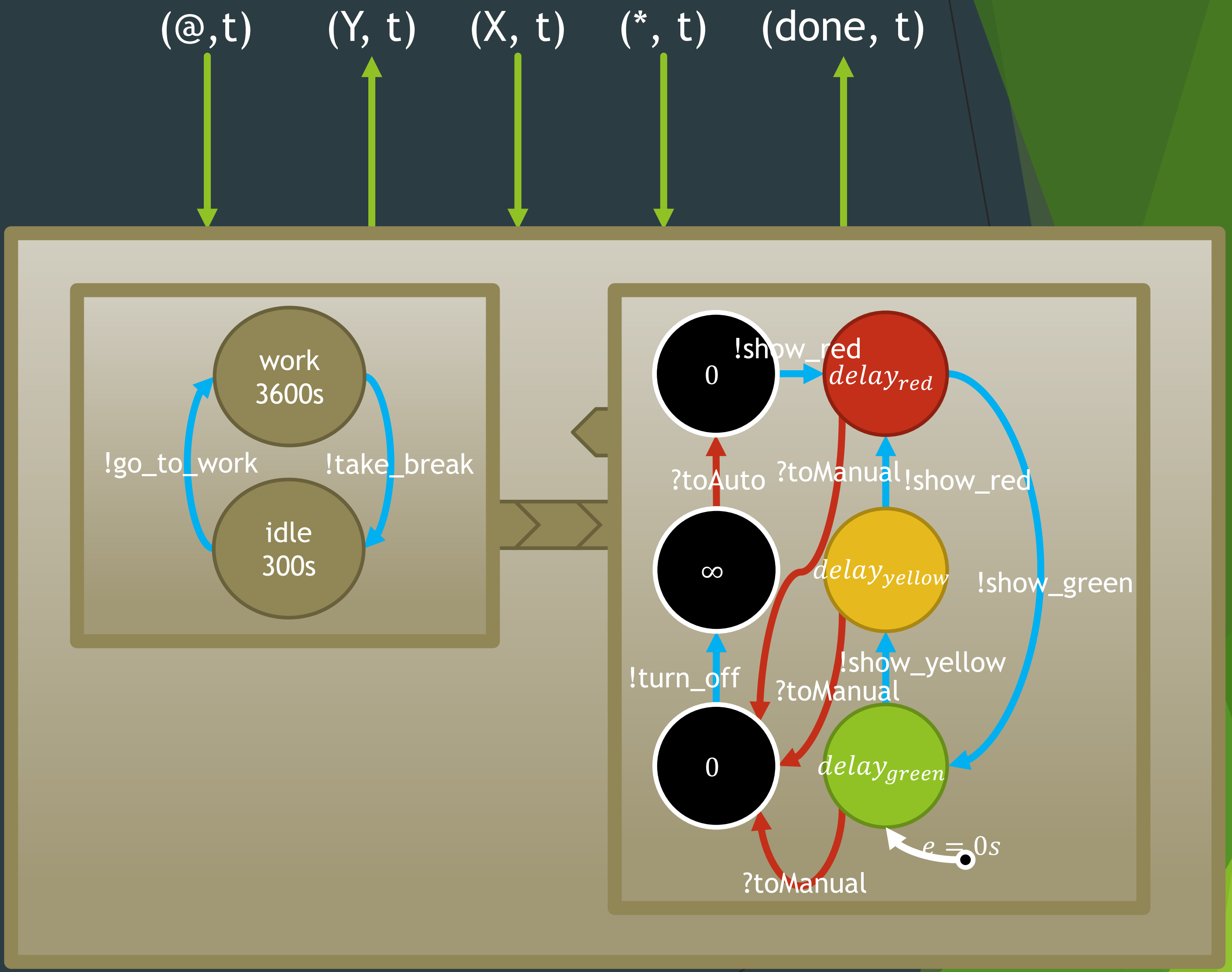


Table Of Contents

Previous topic

Differences from PyDEVS

Next topic

Examples for Parallel DEVS

This Page

Show Source

Quick search

Go

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.

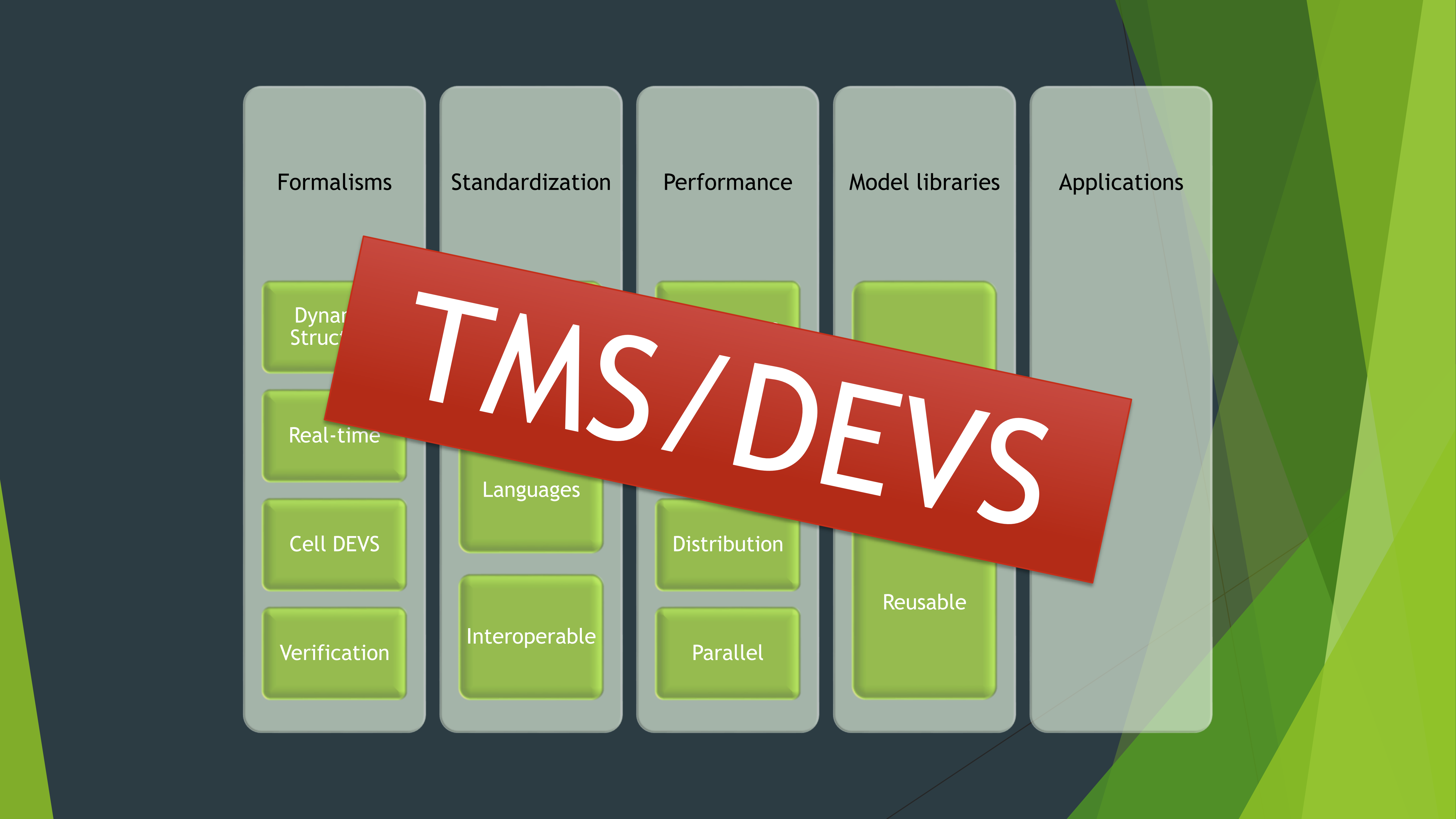
Generator

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
- External transition function will never happen (as there are no inputs)

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


Formalisms
Standardization
Performance
Model libraries
Applications
Real-time
Cell DEVS
Verification
Languages
Interoperable
Distribution
Parallel
Reusable
TMS/DEVS

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