Classic DEVS
An Introduction Using PythonPDEVSP

Yentl Van Tendeloo, Hans Vangheluwe
Introduction


An overview of PythonPDEVS

Yentl Van Tendeloo1 and Hans Vangheluwe1,2
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.

An evaluation of DEVS simulation tools

Yentl Van Tendeloo1 and Hans Vangheluwe1,2,3
1 University of Antwerp, Belgium
2 McGill University, Canada
3 Eurecom Institute, Frascati, Italy

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.
DEVS
GPSS
Meijin++
SimScript
...
Sequential Discrete Event Language

DEVS
= modular simulation assembly language
finite number of non-$\phi$ events in a finite time interval
Experimentation
Simulation

$delay_{red} = 60s$
$delay_{yellow} = 3s$
$delay_{green} = 57s$
$q_{init, light1} = (green, 0)$
$q_{init, pol1} = (idle, 280)$
$cond_{termination} = (t_{sim} \geq t_{end})$
$t_{end} = 24h$
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()
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
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
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
$delay_{\text{red}}$ 

$delay_{\text{yellow}}$ 

$delay_{\text{green}}$
$S$: set of sequential states
$S = \{ \text{red, yellow, green} \}$

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

$ta: S \rightarrow \mathbb{R}_{0,+\infty}^+$
$ta = \{ \text{red} \rightarrow \text{delay}_{\text{red}}, \text{green} \rightarrow \text{delay}_{\text{green}}, \text{yellow} \rightarrow \text{delay}_{\text{yellow}} \}$

$M = \{ S, \delta_{\text{int}}, ta \}$

Autonomous (no input)
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

\[ t_a(red) \]
\[ t_a(green) \]
\[ t_a(yellow) \]
Elapsed time

\[ ta(s_i) \]

\[ e_i \]

\[ \sigma_i \]
Initialization of Initial State

\[ s_0, \sigma_0, e_0, q_{\text{init}} = (s_0, e_0), (s_0, ta(s_0)) \]
Elapsed time

\[ t_{e_0} = t_{a(s_0)} + t_{a(green)} + t_{a(yellow)} \]
\( S \) : set of sequential states

\( S = \{ \text{red, yellow, green} \} \)

\( \delta_{\text{int}} : S \rightarrow S \)

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

\( t_{\text{a}} : S \rightarrow \mathbb{R}^+_0, +\infty \)

\( t_{\text{a}} = \{ \text{red} \rightarrow \text{delay}_{\text{red}}, \text{green} \rightarrow \text{delay}_{\text{green}}, \text{yellow} \rightarrow \text{delay}_{\text{yellow}} \} \)

\( q_{\text{init}} : Q - \text{set of total states} \)

\( Q = \{(s, e) | s \in S, 0 \leq e \leq t_{\text{a}}(s)\} \)

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

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

Concrete Syntax

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]

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

INTERNAL TRANSITION in model <light>
New State: yellow
Output Port Configuration:
Next scheduled internal transition at time 60.00

INTERNAL TRANSITION in model <light>
New State: red
Output Port Configuration:
Next scheduled internal transition at time 120.00
$e = 0 \text{s}$

- red
  - delay$_{\text{red}}$
  - !show$_{\text{red}}$
  - !show$_{\text{green}}$
  - !show$_{\text{yellow}}$

- yellow
  - delay$_{\text{yellow}}$
  - !show$_{\text{yellow}}$
  - !show$_{\text{green}}$

- green
  - delay$_{\text{green}}$
  - !show$_{\text{green}}$
  - !show$_{\text{yellow}}$

- $t = 177$
Autonomous (with output)

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

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

\[ Y : \text{set of output events} \]
\[ Y = \{ \text{"show_red"}, \text{"show_green"}, \text{"show_yellow"} \} \]
\[ \lambda : S \rightarrow Y \cup \{\phi\} \]
\[ \lambda = \{ \text{green} \rightarrow \text{"show_yellow"}, \]
\[ \text{yellow} \rightarrow \text{"show_red"}, \]
\[ \text{red} \rightarrow \text{"show_green"} \} \]
Abstract Syntax

\[ S = \{ \text{red, yellow, green} \} \]
\[ q_{\text{init}} = (\text{green, 0}) \]
\[ \delta_{\text{int}} = \{ \text{red} \rightarrow \text{green}, \]
\[ \text{green} \rightarrow \text{yellow}, \]
\[ \text{yellow} \rightarrow \text{red} \} \]
\[ ta = \{ \text{red} \rightarrow \text{delay}_{\text{red}}, \]
\[ \text{green} \rightarrow \text{delay}_{\text{green}}, \]
\[ \text{yellow} \rightarrow \text{delay}_{\text{yellow}} \} \]
\[ Y = \{ \text{“show_red”}, \]
\[ \text{“show_green”}, \]
\[ \text{“show_yellow”} \} \]
\[ \lambda = \{ \text{green} \rightarrow \text{“show_yellow”}, \]
\[ \text{yellow} \rightarrow \text{“show_red”}, \]
\[ \text{red} \rightarrow \text{“show_green”} \} \]

Concrete Syntax

```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”}
```

Operational Semantics

time = 0
current_state = initial_state
last_time = -initial_elapsed
while not termination_condition():
    time = last_time + ta(current_state)
    output(\(\lambda(\text{current\_state})\))
    current_state = \(\delta_{\text{int}}(\text{current\_state})\)
    last_time = time
INITIAL CONDITIONS in model <light>
Initial State: green
Next scheduled internal transition at 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

INTERNAL TRANSITION in model <light>
New State: red
Output Port Configuration:
  port <observer>:
    show_red
Next scheduled internal transition at time 120.00
\( M = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle \)

\( Y = \{"show\_red", "show\_green", "show\_yellow"\} \)

\( S = \{\text{red}, \text{yellow}, \text{green}, \text{manual}\} \)

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

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

\( \lambda = \{\text{green} \rightarrow "show\_yellow", \text{yellow} \rightarrow "show\_red", \text{red} \rightarrow "show\_green"\} \)

\( t_a = \{\text{red} \rightarrow \text{delay}_{\text{red}}, \text{green} \rightarrow \text{delay}_{\text{green}}, \text{yellow} \rightarrow \text{delay}_{\text{yellow}}, \text{manual} \rightarrow +\infty\} \)

\( X \): set of input events

\( X = \{"\text{toAuto}", "\text{toManual}"\} \)

\( \delta_{ext} : Q \times X \rightarrow S \)

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

\( \delta_{ext} = \{( (\ast, \ast), "\text{toManual}" ) \rightarrow "\text{manual}" , ( ("\text{manual}" , \ast), "\text{toAuto}" ) \rightarrow "\text{red}" \} \)
Abstract Syntax

\[ Y = \{\text{"show_red"}, \text{"show_green"}, \text{"show_yellow"}\} \]
\[ S = \{\text{red}, \text{yellow}, \text{green}, \text{manual}\} \]
\[ q_{\text{init}} = (\text{green}, 0) \]
\[ \delta_{\text{int}} = \{\text{red} \rightarrow \text{green}, \]
\[ \quad \text{green} \rightarrow \text{yellow}, \]
\[ \quad \text{yellow} \rightarrow \text{red}\} \]
\[ \lambda = \{\text{green} \rightarrow \"show_yellow\", \]
\[ \quad \text{yellow} \rightarrow \"show_red\", \]
\[ \quad \text{red} \rightarrow \"show_green\"\} \]
\[ ta = \{\text{red} \rightarrow delay_{\text{red}}, \]
\[ \quad \text{green} \rightarrow delay_{\text{green}}, \]
\[ \quad \text{yellow} \rightarrow delay_{\text{yellow}}, \]
\[ \quad \text{manual} \rightarrow \infty\} \]
\[ X = \{\text{"toAuto"}, \text{"toManual"}\} \]
\[ \delta_{\text{ext}} = \{(\*, \*), \text{"toManual"}) \rightarrow \text{manual,} \]
\[ \quad (\text{manual, \*}, \text{"toAuto"}) \rightarrow \text{red}\} \]

Operational Semantics

\[ \text{time} = 0 \]
\[ \text{current_state} = \text{initial_state} \]
\[ \text{last_time} = -\text{initial_elapsed} \]
\[ \text{while not termination_condition():} \]
\[ \quad \text{next_time} = \text{last_time} + ta(\text{current_state}) \]
\[ \quad \text{if time_next_ev} \leq \text{next_time:} \]
\[ \quad \quad e = \text{time_next_ev} - \text{last_time} \]
\[ \quad \quad \text{time} = \text{time_next_ev} \]
\[ \quad \quad \text{current_state} = \delta_{\text{ext}}((\text{current_state}, e), \text{next_ev}) \]
\[ \quad \text{else:} \]
\[ \quad \quad \text{time} = \text{next_time} \]
\[ \quad \quad \text{output(}\lambda(\text{current_state})) \]
\[ \quad \quad \text{current_state} = \delta_{\text{int}}(\text{current_state}) \]
\[ \quad \text{last_time} = \text{time} \]
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"
INITIAL CONDITIONS in model <light>
  Initial State: green
  Next scheduled internal transition at 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

INTERNAL TRANSITION in model <light>
  New State: red
  Output Port Configuration:
    port <observer>:
      show_red
  Next scheduled internal transition at time 120.00
\[ Q = \{ (s, e) | s \in S, 0 \leq e \leq ta(s) \} \]

- \( e = 0 \)
- \( 0 < e < ta(s) \)
- \( e = ta(s) \)
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
\[
e = 0\text{s}
\]
Full Atomic DEVS Specification

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

- **\( X \)**: set of input events
- **\( Y \)**: set of output events
- **\( S \)**: set of sequential states
- **\( q_{\text{init}} \)**: \( Q \)
  \[ Q = \{ (s, e) | s \in S, 0 \leq e \leq ta(s) \} \]
- **\( \delta_{\text{int}} \)**: \( S \rightarrow S \)
- **\( \delta_{\text{ext}} \)**: \( Q \times X \rightarrow S \)
- **\( \lambda \)**: \( S \rightarrow Y \cup \{ \phi \} \)
- **\( ta \)**: \( S \rightarrow \mathbb{R}_{0,+}^+ \)
\[ (\delta_{\text{ext}}(s_i, e), x), 0) \]

\[ \delta_{\text{ext}} \]

\[ (s_i, 0) \]

\[ e \]

\[ t_i \]

\[ t_i + e \]

\[ t_i + ta(s_i) \]

\[ \delta_{\text{int}} \]

\[ \delta_{\text{int}}(s_i), 0 \]

\[ \text{output } \lambda(s_i) \]
Coupled Models
\[ e = 0 \text{s} \]
work 3600s

!go_to_work !take_break

idle 300s

e = 280s
\[ C = \langle D, MS \rangle \]
\[ MS = \{ M_i | i \in D \} \]
\[ M_i = \{ X_i, Y_i, S_i, q_{init,i}, \delta_{int,i}, \delta_{ext,i}, \lambda_i, t_{a_i} \}, \forall i \in D \]
\( \mathcal{D} = \{ \text{light}_1, \text{pol}_1 \} \)

\( C = \langle \mathcal{D}, MS \rangle \)

\( MS = \{ M_i | i \in \mathcal{D} \} \)

\( M_i = \langle X_i, Y_i, S_i, q_{\text{init}, i}, \delta_{\text{int}, i}, \delta_{\text{ext}, i}, \lambda_i, t_{a_i} \rangle, \forall i \in \mathcal{D} \)
\[ C = \langle X_{\text{self}}, Y_{\text{self}}, D, MS \rangle \]

\[ MS = \{ M_i | i \in D \} \]

\[ M_i = \langle X_i, Y_i, S_i, q_{\text{init},i}, \delta_{\text{int},i}, \delta_{\text{ext},i}, \lambda_i, t_{a_i} \rangle, \forall i \in D \]
\[ C = \{X_{self}, Y_{self}, D, MS, IS\} \]
\[ MS = \{M_i | i \in D\} \]
\[ M_i = \{X_i, Y_i, S_i, q_{init,i}, \delta_{int,i}, \delta_{ext,i}, \lambda_i, t\alpha_i\}, \forall i \in D \]
\[ IS = \{I_i | i \in D \cup \{\text{self}\}\} \]
\[ \forall i \in D \cup \{\text{self}\} : I_i \subseteq D \cup \{\text{self}\} \]
\[ \forall i \in D \cup \{\text{self}\} : i \notin I_i \]

\[ I_i : \text{Influencees of } i \]
\[ C = \{X_{\text{self}}, Y_{\text{self}}, D, MS, IS\} \]
\[ MS = \{M_i | i \in D\} \]
\[ M_i = \{X_i, Y_i, S_i, q_{\text{init},i}, \delta_{\text{int},i}, \delta_{\text{ext},i}, \lambda_i, t\alpha_i\}, \forall i \in D \]
\[ IS = \{l_i | i \in D \cup \{\text{self}\}\} \]
\[ \forall i \in D \cup \{\text{self}\} : l_i \subseteq D \cup \{\text{self}\} \]
\[ \forall i \in D \cup \{\text{self}\} : i \notin I_i \]

\( I_i : \text{Influencees of } i \)

\( I_{\text{self}} = \{\text{light1}\} \)

\( I_{\text{pol1}} = \{\text{light1}\} \)

\( I_{\text{light1}} = \{\text{self}\} \)
\[ C = \{ X_{self}, Y_{self}, D, MS, IS, ZS \} \]

\[ MS = \{ M_i | i \in D \} \]

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

\[ IS = \{ I_i | i \in D \cup \{ \text{self} \} \} \]

\[ \forall i \in D \cup \{ \text{self} \} : I_i \subseteq D \cup \{ \text{self} \} \]

\[ \forall i \in D \cup \{ \text{self} \} : i \notin I_i \]

\[ ZS = \{ Z_{i,j} | i \in D \cup \{ \text{self} \}, j \in I_i \} \]

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

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

\[ Z_{i,j} : Y_i \rightarrow X_j, \forall i, j \in D \]
\[ C = \{ X_{self}, Y_{self}, D, MS, IS, ZS \} \]
\[ MS = \{ M_i | i \in D \} \]
\[ M_i = \{ X_i, Y_i, S_i, q_{init}, i, \delta_{int}, i, \delta_{ext}, i, \lambda_i, \tau a_i \}, \forall i \in D \]
\[ IS = \{ I_i | i \in D \cup \{ self \} \} \]
\[ \forall i \in D \cup \{ self \} : I_i \subseteq D \cup \{ self \} \]
\[ \forall i \in D \cup \{ self \} : i \notin I_i \]
\[ ZS = \{ Z_{i,j} | i \in D \cup \{ self \}, j \in I_i \} \]
\[ Z_{self,j} : X_{self} \to X_j, \forall j \in D \]
\[ Z_{i,self} : Y_i \to Y_{self}, \forall i \in D \]
\[ Z_{i,j} : Y_i \to X_j, \forall i, j \in D \]
\[ C = \langle X_{self}, Y_{self}, D, MS, IS, ZS, \text{select} \rangle \]

\[ MS = \{ M_i | i \in D \} \]

\[ M_i = \langle X_i, Y_i, S_i, q_{\text{init},i}, \delta_{\text{int},i}, \delta_{\text{ext},i}, \lambda_i, t_{a_i} \rangle, \forall i \in D \]

\[ IS = \{ I_i | i \in D \cup \{ \text{self} \} \} \]

\[ \forall i \in D \cup \{ \text{self} \}: I_i \subseteq D \cup \{ \text{self} \} \]

\[ \forall i \in D \cup \{ \text{self} \}: i \notin I_i \]

\[ ZS = \{ Z_{i,j} | i \in D \cup \{ \text{self} \}, j \in I_i \} \]

\[ Z_{\text{self},j} : X_{self} \rightarrow X_j, \forall j \in D \]

\[ Z_{i,\text{self}} : Y_i \rightarrow Y_{self}, \forall i \in D \]

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

\[ \text{select} : 2^D \rightarrow D \]

\[ \forall E \subseteq D, E \neq \emptyset: \text{select}(E) \in E \]
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
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
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
INTERNAL TRANSITION in model <system.light>
Output Port Configuration:
  port <observer>:
    turn_off
New State: manual
Next scheduled internal transition at time inf
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
INTERNAL TRANSITION in model <system.light>
Output Port Configuration:
  port <observer>:
    show_red
New State: red
Next scheduled internal transition at time 3680.00
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
$CM = \langle X_{\text{self}}, Y_{\text{self}}, D, MS, IS, ZS \rangle$

$\text{flatten}(CM) = \langle X, Y, S, q_{\text{init}}, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta \rangle$
Hierarchical Simulator
## DEVS Semantics

<table>
<thead>
<tr>
<th></th>
<th>Operational Semantics</th>
<th>Denotational Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic DEVS</td>
<td>Abstract Simulator</td>
<td></td>
</tr>
<tr>
<td>Coupled DEVS</td>
<td>Hierarchical Simulator</td>
<td>Closure under Coupling</td>
</tr>
</tbody>
</table>

Conclusions

- Atomic DEVS
- Coupled DEVS
- Closure under coupling
- Abstract Simulator
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
- Input function never returns a message it did not generate
- External transition function will never happen (as there are no inputs)

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]

Formalisms
- Dynamic Structure
- Real-time
- Cell DEVS
- Verification

Standardization
- Tools
- Languages
- Interoperable

Performance
- Algorithms
- Activity
- Distribution
- Parallel

Model libraries
- Example
- Reusable

Applications