Classic DEVS

An Introduction Using PythonPDEVS

Yentl Van Tendeloo, Hans Vangheluwe
Introduction
Bernard P. Zeigler. 
*Theory Of Modelling And Simulation.*  

Bernard P. Zeigler. 
*Multifacetted Modelling and Discrete Event Simulation.*  

Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. 
*Theory Of Modelling And Simulation.*  
Yentl Van Tendeloo and Hans Vangheluwe.
An Overview of PythonPDEVS.

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 total state. This was left implicit in the original DEVS specification.

---

Extending the DEVS Formalism with Initialization Information

Yentl Van Tendeloo*   Hans Vangheluwe‡
{Yentl.VanTendeloo,Hans.Vangheluwe}@uantwerpen.be

DEVS is a popular formalism to model system behaviour using a discrete-event abstraction. The main advantages of DEVS are its rigorous and precise specification, as well as its support for modular, hierarchical construction of models. DEVS frequently serves as a simulation “assembly language” to which models in other formalisms are translated, either giving meaning to new (domain-specific) languages, or reproducing semantics of existing languages. Despite this rigorous definition of its syntax and semantics, initialization of DEVS models is left unspecified in both the Classic and Parallel DEVS formalism definition. In this paper, we extend the DEVS formalism by including an initial total state. Extensions to syntax as well as denotational (closure under coupling) and operational semantics (abstract simulator) are presented. The extension is applicable to both main variants of the DEVS formalism. Our extension is such that it adds to, but does not alter the original specification. All changes are illustrated by means of a traffic light example.

Keywords: Classic DEVS, Parallel DEVS, Experimentation, Initialization

---

Sequential Discrete Event Language

Meijn++

GPSS

SimScript

DEVS

= modular simulation assembly language
Vangheluwe, Hans. DEVS as a common denominator for multi-formalism hybrid systems modelling.
finite number of non-$\phi$ events in a finite time interval
Experimentation
Simulation

\[
\begin{align*}
\text{delay}_{\text{red}} &= 60s \\
\text{delay}_{\text{yellow}} &= 3s \\
\text{delay}_{\text{green}} &= 57s \\
q_{\text{init,}\text{light1}} &= (\text{green}, 0) \\
q_{\text{init,}\text{pol1}} &= (\text{idle}, 280) \\
\text{cond}_{\text{termination}} &= (t_{\text{sim}} \geq t_{\text{end}}) \\
t_{\text{end}} &= 24h
\end{align*}
\]
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
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
\( M = \langle S, \delta_{\text{int}}, ta \rangle \)

\( S \) : set of sequential states
\( S = \{\text{red, yellow, green}\} \)

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

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

- ta(red)
- ta(green)
- ta(yellow)

- enter red
- enter green
- enter yellow
Elapsed time

\[ \text{ta}(s_i) \]

\[ e_i \]

\[ \sigma_i \]
Initialization of Initial State

\[ (s_0, 0) \quad q_{init} = (s_0, e_0) \quad (s_0, ta(s_0)) \]
Elapsed time

- $e_0$ to $e$
- $t_a(s_0)$
- $t_a(red)$
- $t_a(green)$
- $t_a(yellow)$

- enter yellow
- enter red
- enter yellow
- enter green
- enter yellow
\( S_0: \text{set of sequential states} \)
\[ S_0 = \{\text{red, yellow, green}\} \]
\( \delta_{\text{int}}: S_0 \rightarrow S_0 \)
\[ \delta_{\text{int}} = \{\text{red} \rightarrow \text{green}, \text{green} \rightarrow \text{yellow}, \text{yellow} \rightarrow \text{red}\} \]
\( t_a: S_0 \rightarrow \mathbb{R}^+_{0, +\infty} \)
\[ t_a = \{\text{red} \rightarrow \text{delay}_{\text{red}}, \text{green} \rightarrow \text{delay}_{\text{green}}, \text{yellow} \rightarrow \text{delay}_{\text{yellow}}\} \]
\( q_{\text{init}}: Q - \text{set of total states} \)
\[ Q = \{(s, e) | s \in S_0, 0 \leq e \leq t_a(s)\} \]
\[ q_{\text{init}} = (\text{green}, 0) \]
Abstract Syntax

\[ S = \{ \text{red, yellow, green} \} \]

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

\[ t_{\text{a}} = \{ \text{red \rightarrow delay_red, green \rightarrow delay_green, yellow \rightarrow delay_yellow} \} \]

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

Concrete Syntax

```python
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

```python
time = 0
current_state = initial_state
last_time = -initial_elapsed
while not termination_condition():
    time = last_time + ta(current_state)
    current_state = \delta_{\text{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
自主 (带有输出) 

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

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

- \( Y : \text{set of output events} \)
- \( Y = \{\text{"show_red"}, \text{"show_green"}, \text{"show_yellow"}\} \)
- \( \lambda : S \rightarrow Y \cup \{\phi\} \)
- \( \lambda = \{ \text{green} \rightarrow \text{"show_yellow"}, \)  
  \( \text{yellow} \rightarrow \text{"show_red"}, \)  
  \( \text{red} \rightarrow \text{"show_green"}\} \)

\( e = 0s \)
Abstract Syntax

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

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

\[
\text{time} = 0 \\
\text{current_state} = \text{initial_state} \\
\text{last_time} = \text{-initial_elapsed} \\
\text{while not} \quad \text{termination_condition}(): \\
\quad \text{time} = \text{last_time} + \text{ta(} \text{current_state} \text{)} \\
\quad \text{output}(\lambda(\text{current_state})) \\
\quad \text{current_state} = \delta_{int}(\text{current_state}) \\
\quad \text{last_time} = \text{time}
\]
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
To Manual

Delay $\text{delay}_{\text{red}}$

To Manual

Delay $\text{delay}_{\text{yellow}}$

To Manual

Delay $\text{delay}_{\text{green}}$

$e = 0s$

X

toManual

toAuto

S

manual

red

yellow

green

Y

show_green

red

yellow

green

X

t

S

t

Y

t
Reactive

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

\[ Y = \{ "show\_red", "show\_green", "show\_yellow" \} \]

\[ S = \{ red, yellow, green, manual \} \]

\[ q_{init} = (green, 0) \]

\[ \delta_{int} = \{ red \rightarrow green, \]
\[ green \rightarrow yellow, \]
\[ yellow \rightarrow red \} \]

\[ \lambda = \{ green \rightarrow "show\_yellow", \]
\[ yellow \rightarrow "show\_red", \]
\[ red \rightarrow "show\_green" \} \]

\[ ta = \{ red \rightarrow \text{delay}_red, \]
\[ green \rightarrow \text{delay}_green, \]
\[ yellow \rightarrow \text{delay}_yellow, \]
\[ manual \rightarrow +\infty \} \]

\[ X : \text{set of input events} \]

\[ X = \{ "toAuto", "toManual" \} \]

\[ \delta_{ext} : Q \times X \rightarrow S \]

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

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

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

Operational Semantics

\[ \text{time} = 0 \]
\[ \text{current\_state} = \text{initial\_state} \]
\[ \text{last\_time} = -\text{initial\_elapsed} \]
\[ \text{while not termination\_condition()}: \]
\[ \quad \text{next\_time} = \text{last\_time} + \text{ta(\text{current\_state})} \]
\[ \quad \text{if } \text{time\_next\_ev} \leq \text{next\_time}: \]
\[ \quad \quad \text{e} = \text{time\_next\_ev} - \text{last\_time} \]
\[ \quad \quad \text{time} = \text{time\_next\_ev} \]
\[ \quad \quad \text{current\_state} = \delta_{\text{ext}}((\text{current\_state}, \text{e}), \text{next\_ev}) \]
\[ \quad \text{else}: \]
\[ \quad \quad \text{time} = \text{next\_time} \]
\[ \quad \quad \text{output}(\lambda(\text{current\_state})) \]
\[ \quad \quad \text{current\_state} = \delta_{\text{int}}(\text{current\_state}) \]
\[ \text{last\_time} = \text{time} \]
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":
            self.state = 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
\[ Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\} \]
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 = δ_{ext}((current_state, e), next_ev)
    else:
        time = next_time
        output(\lambda(current_state))
        current_state = δ_{int}(current_state)
    last_time = time
Full Atomic DEVS Specification

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

- \( X \) : set of input events
- \( Y \) : set of output events
- \( S \) : set of sequential states
- \( q_{\text{init}} : Q \)
  \[ Q = \{(s, e)|s \in S, 0 \leq e \leq t\alpha(s)\}\]
- \( \delta_{\text{int}} : S \rightarrow S \)
- \( \delta_{\text{ext}} : Q \times X \rightarrow S \)
- \( \lambda : S \rightarrow Y \cup \{\phi\} \)
- \( t\alpha : S \rightarrow \mathbb{R}^+_0, +\infty \)
\[ (\delta_{ext}((s_i, e), x), 0) \]
\[ e \]
\[ (s_i, 0) \]
\[ t_{i + \tau_a(s_i)} \]
\[ \text{output } \lambda(s_i) \]
\[ \delta_{int} \]
\[ (\delta_{int}(s_i), 0) \]
\[ t_i \]
\[ t_i + e \]
\[ t_i + t\alpha(s_i) \]
\[ s \]
\[ t \]
Coupled Models
!go_to_work \to \!take\_break

work 3600s

idle 300s

e = 280s
\( C = \langle D, MS \rangle \)

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

\( M_i = \langle X_i, Y_i, S_i, q_{init,i}, \delta_{int,i}, \delta_{ext,i}, \lambda_i, t\alpha_i \}, \forall i \in D \)
$D = \{\text{light}_1, \text{pol}_1\}$

$C = \langle D, MS \rangle$

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

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

$l_i$ : Influencees of $i$
$I_i$ : Influencees of $i$

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

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

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

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

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

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

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

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

$I_{light1} = \{\text{self}\}$
\[ C = \{X_{\text{self}}, Y_{\text{self}}, D, MS, IS, ZS\} \]
\[ MS = \{M_i | i \in D\} \]
\[ M_i = \{X_i, Y_i, S_i, q_{\text{init}}, i, \delta_{\text{int}}, i, \delta_{\text{ext}}, i, \lambda_i, ta_i\}, \forall i \in D \]
\[ IS = \{I_i | i \in D \cup \{\text{self}\}\} \]
\[ \forall i \in D \cup \{\text{self}\} : I_i \subseteq D \cup \{\text{self}\} \]
\[ \forall i \in D \cup \{\text{self}\} : i \notin I_i \]
\[ ZS = \{Z_{i,j} | i \in D \cup \{\text{self}\}, j \in I_i\} \]
\[ Z_{\text{self},j} : X_{\text{self}} \rightarrow X_j, \forall j \in D \]
\[ Z_{i,\text{self}} : Y_i \rightarrow Y_{\text{self}}, \forall i \in D \]
\[ Z_{i,j} : Y_i \rightarrow X_j, \forall i, j \in D \]
$C = \{X_{self}, Y_{self}, D, MS, IS, ZS\}$

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

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

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

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

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

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

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

$Z_{i,j}: Y_i \rightarrow X_j, \forall i, j \in D$
$C = \langle X_{\text{self}}, Y_{\text{self}}, D, MS, IS, ZS, \text{select} \rangle$

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

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

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

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

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

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

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

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

$\text{select} : 2^D \rightarrow D$

$\forall E \subseteq D, E \neq \emptyset: \text{select}(E) \in E$
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(
            q_init_pol1 = ("idle", 280), q_init_light1 = ("green", 0),
            delay_red = 60, delay_yellow = 3, delay_green = 57)
        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
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
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
$$M_{pol1}$$

- work 3600s
- idle 300s

$$M_{light1}$$

- going_auto 0s
- manual $\infty$
- going_manual 0s
- green delay_green
- yellow delay_yellow
- red delay_red

- show_red
- show_yellow
- show_green

- turn_off

- self

- take_break $\rightarrow$ toAuto
- go_to_work $\rightarrow$ toManual

- ?
\[ CM = \langle X_{\text{self}}, Y_{\text{self}}, D, MS, IS, ZS \rangle \]

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

<table>
<thead>
<tr>
<th>Atomic DEVS</th>
<th>Coupled DEVS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operational Semantics</strong></td>
<td><strong>Denotational Semantics</strong></td>
</tr>
<tr>
<td>Abstract Simulator</td>
<td>[1]</td>
</tr>
<tr>
<td>Hierarchical Simulator</td>
<td>Closure under Coupling</td>
</tr>
</tbody>
</table>

Limitations of Classic DEVS

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

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 is called exactly once
- External transition function will never happen (as there are no inputs)

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

Full Atomic DEV5 Specification

\[ M = (X, Y, S, \phi_{\text{init}}, \delta_{\text{init}}, \delta_{\text{ext}}, \lambda, \tau_a) \]

- \( X \) : set of input events
- \( Y \) : set of output events
- \( S \) : set of sequential states

\[ \phi_{\text{init}} : Q \rightarrow Q \]

\[ \delta_{\text{init}} : S \rightarrow \mathbb{S} \]

\[ \delta_{\text{ext}} : Q \times X \rightarrow S \]

\[ \lambda : S \rightarrow Y \cup \{\phi\} \]

\[ \tau_a : S \rightarrow \mathbb{R}_{\geq 0} \]

\[ C = \{ \text{array} \text{, array} \text{, B, M, S, E, \ldots, select} \} \]

\[ M = (\text{array} \text{, array} \text{, array} \text{, array} \text{, array}) \]

\[ X = \{0, 1\} \]

\[ Y = \{0, 1\} \]

\[ S = \{0, 1\} \]

\[ \phi = \{\phi\} \]

\[ \lambda = \{\lambda\} \]

\[ \tau_a = \{\tau_a\} \]


E-mail: Yentl.VanTendeloo@uantwerpen.be