WINTER SIMULATION CONFERENCE

SIMULATION FOR A SMART WORLD:
FROM SMART DEVICES TO SMART CITIES

IN PERSON: DECEMBER 13-15 | VIRTUAL: DECEMBER 15-17

Specifying and simulating hybrid modelling languages: the combination of (embedding in) ODE/CT-CBD and TFSA by mapping them onto DEVS

PI-DEV + TFSA + ***TFSA>(LCC+ODE)*** *onto DEVS*

Randy Paredis

Joachim Denil

Hans Vangheluwe

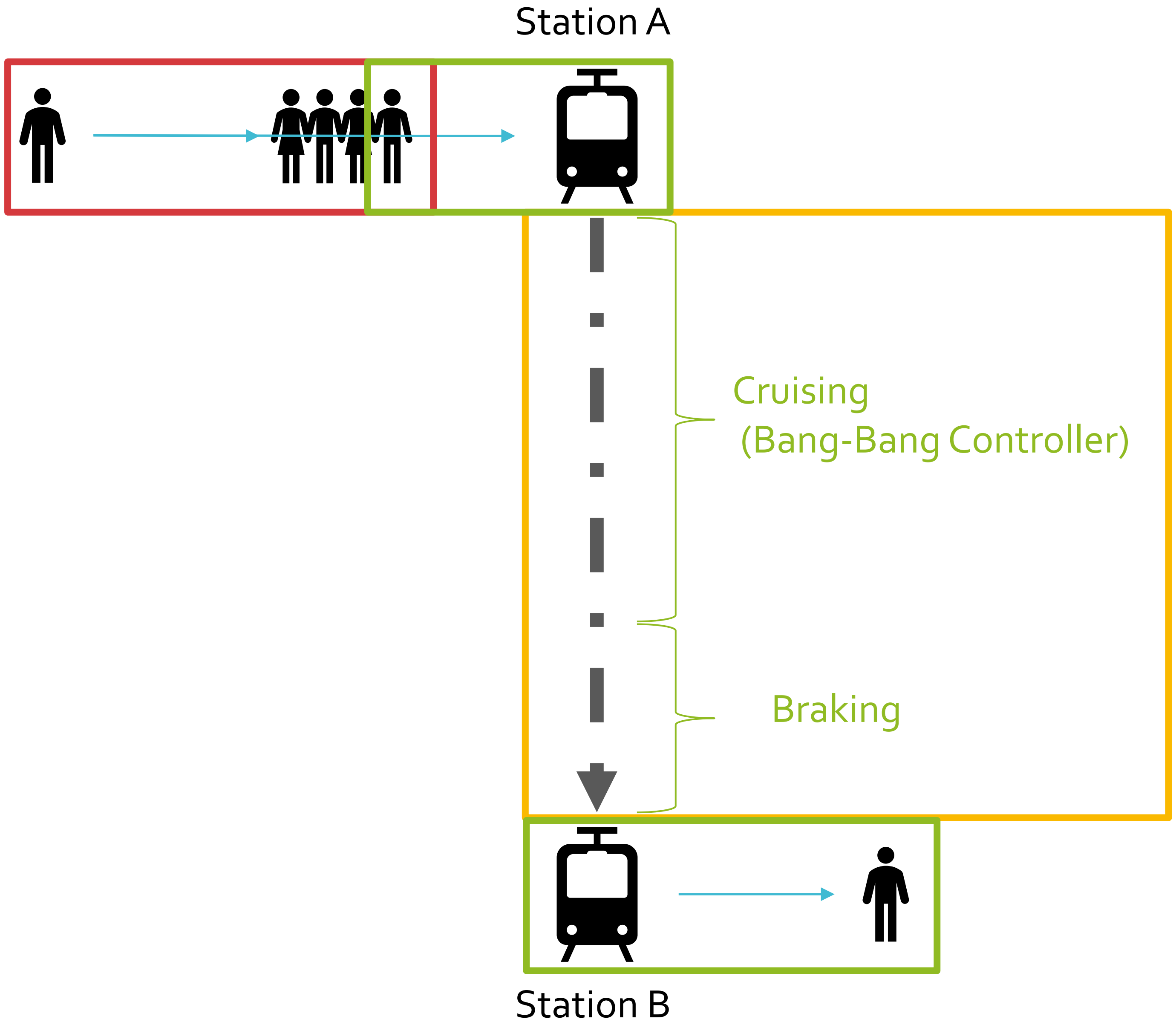
FLANDERS MAKE

MSDL
Modelling, Simulation and Design Lab

Universiteit Antwerpen

Use Case:

PRT



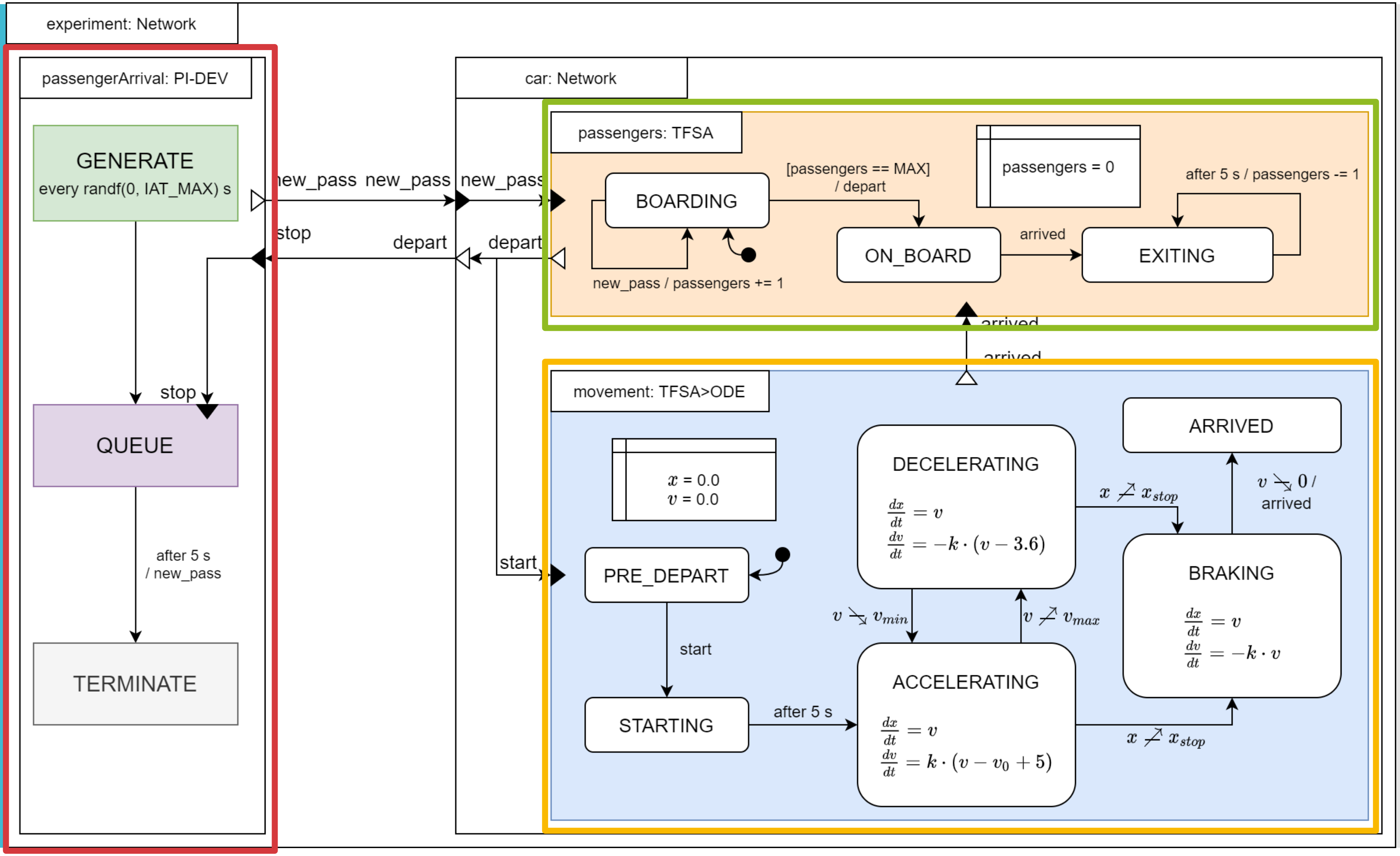
Station A

Cruising
(Bang-Bang Controller)

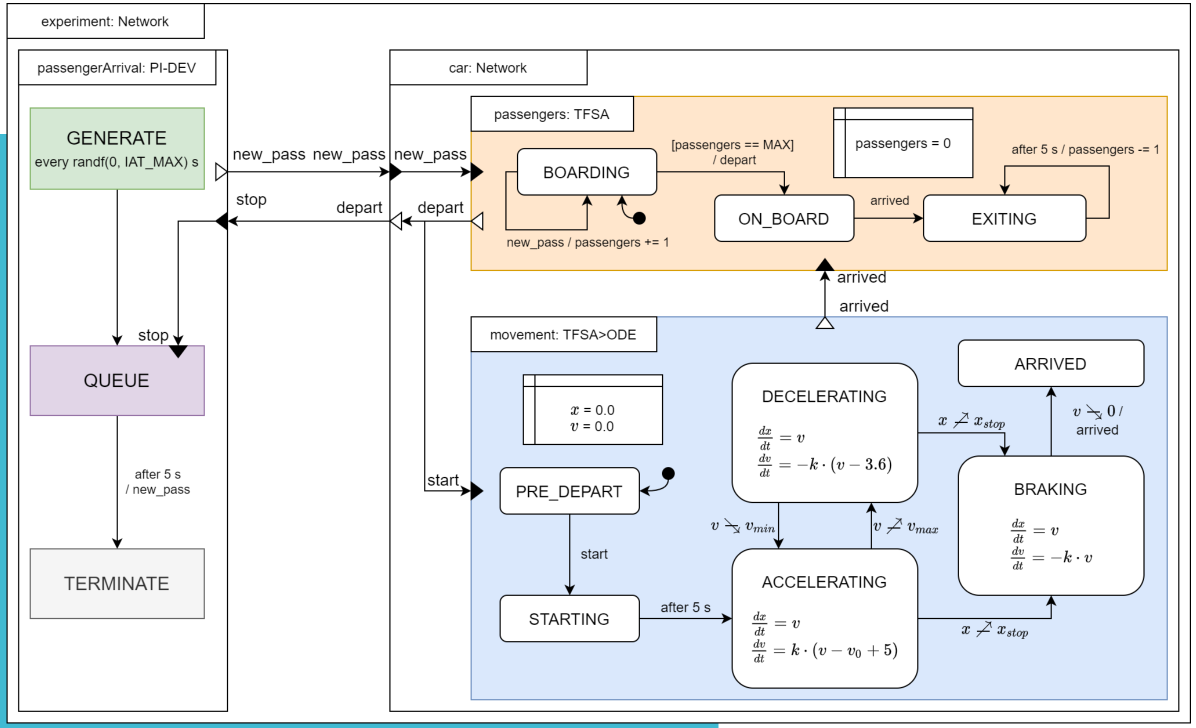
Braking

Station B

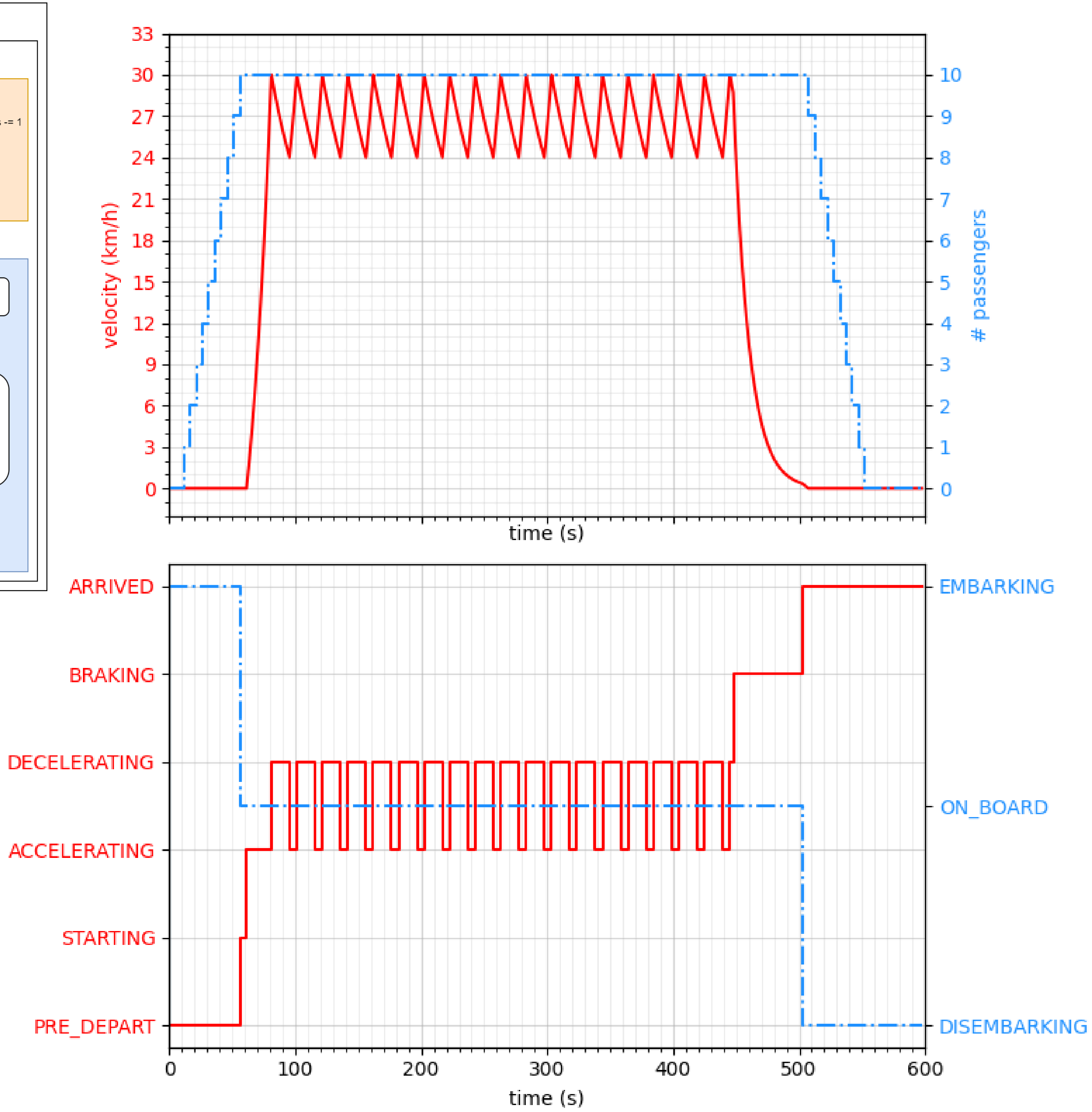
Model



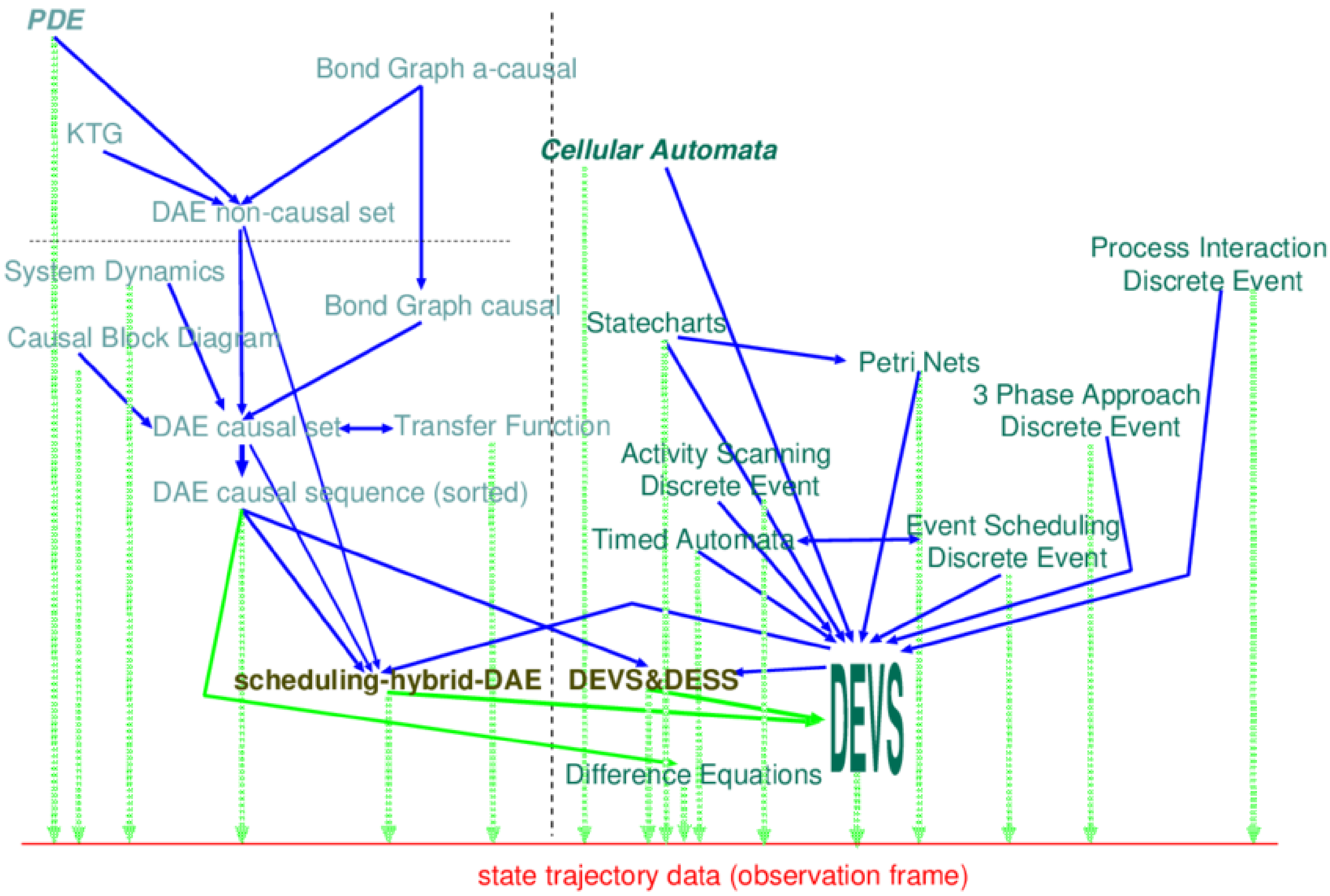
experiment: Network
passengerArrival: PI-DEV
GENERATE
every randf(0, IAT_MAX) s
QUEUE
stop
after 5 s
/ new_pass
TERMINATE
new_pass
new_pass
new_pass
stop
depart
depart
start
car: Network
passengers: TFSA
BOARDING
[passengers == MAX]
/ depart
passengers = 0
after 5 s / passengers -= 1
ON_BOARD
arrived
EXITING
new_pass / passengers += 1
arrived
arrived
movement: TFSA>ODE
x = 0.0
v = 0.0
PRE_DEPART
start
STARTING
after 5 s
DECELERATING
$\frac{dx}{dt} = v$
$\frac{dv}{dt} = -k \cdot (v - 3.6)$
$v \searrow v_{min}$
$v \nearrow v_{max}$
ACCELERATING
$\frac{dx}{dt} = v$
$\frac{dv}{dt} = k \cdot (v - v_0 + 5)$
$x \nearrow x_{stop}$
$x \nearrow x_{stop}$
ARRIVED
$v \searrow 0$ /
arrived
BRAKING
$\frac{dx}{dt} = v$
$\frac{dv}{dt} = -k \cdot v$



Simulation Trace

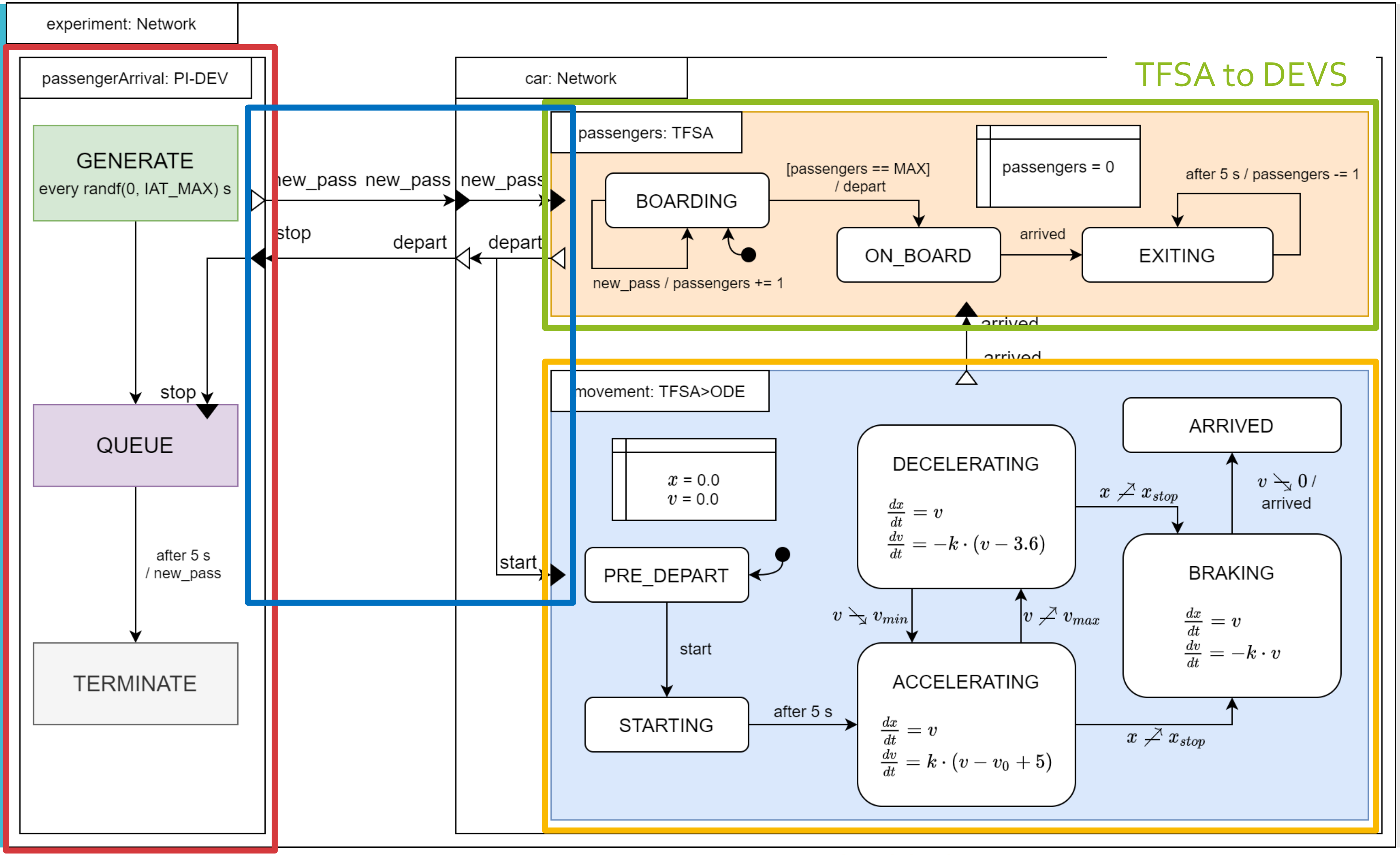


DEVS as a common denominator



Vangheluwe, H., & Vansteenkiste, G. C. (1996) "*A multi-paradigm modeling and simulation methodology: Formalisms and Languages*". In *European Simulation Symposium (ESS)* , pages 168 – 172. Society for Computer Simulation International (SCS). Genoa, Italy

Model

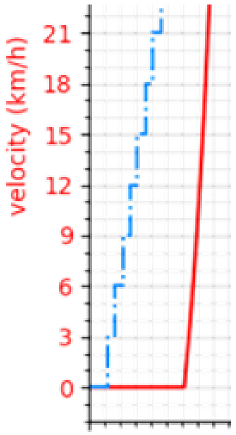


GPSS2DEVS

ODE embedded in TFSA to DEVS

Paredis, R. & Vangheluwe, H. (2020) "*Translating Process Interaction World View Models To DEVS: GPSS To (Python(P))DEVS*". In *Proceedings of the 2020 Winter Simulation Conference (WSC)*, pages 2221-2232.

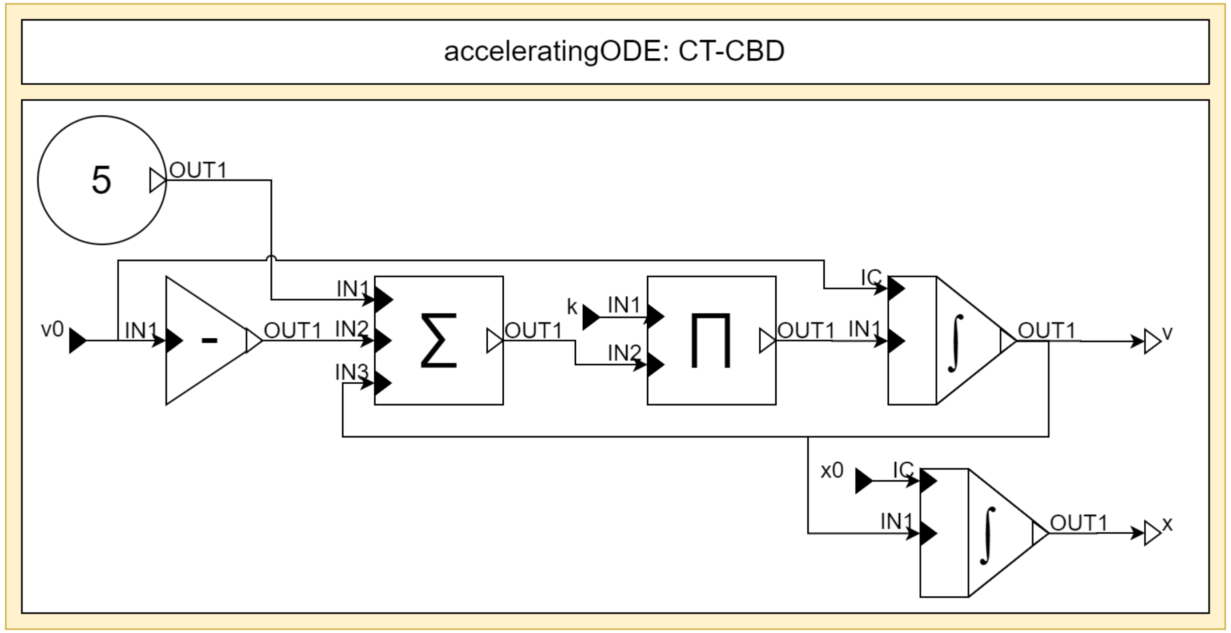
ODE and CT-CBD



accelerating: ODE

$$\frac{dx}{dt} = v$$
$$\frac{dv}{dt} = k \cdot (v - v_0 + 5)$$

$$x(0) = x_0$$
$$v(0) = v_0$$



Discretization

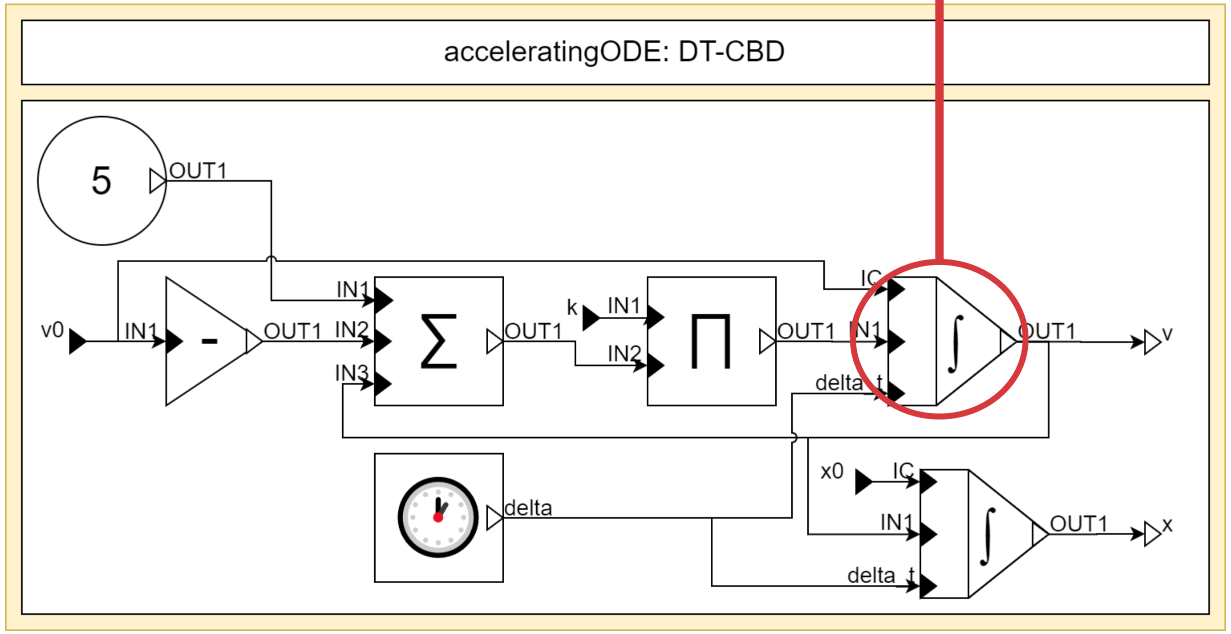
$$y(t) = \mathrm{y}(\mathrm{t} - \Delta t) + \frac{\Delta t}{2} \cdot (x(t) + x(t - \Delta t))$$

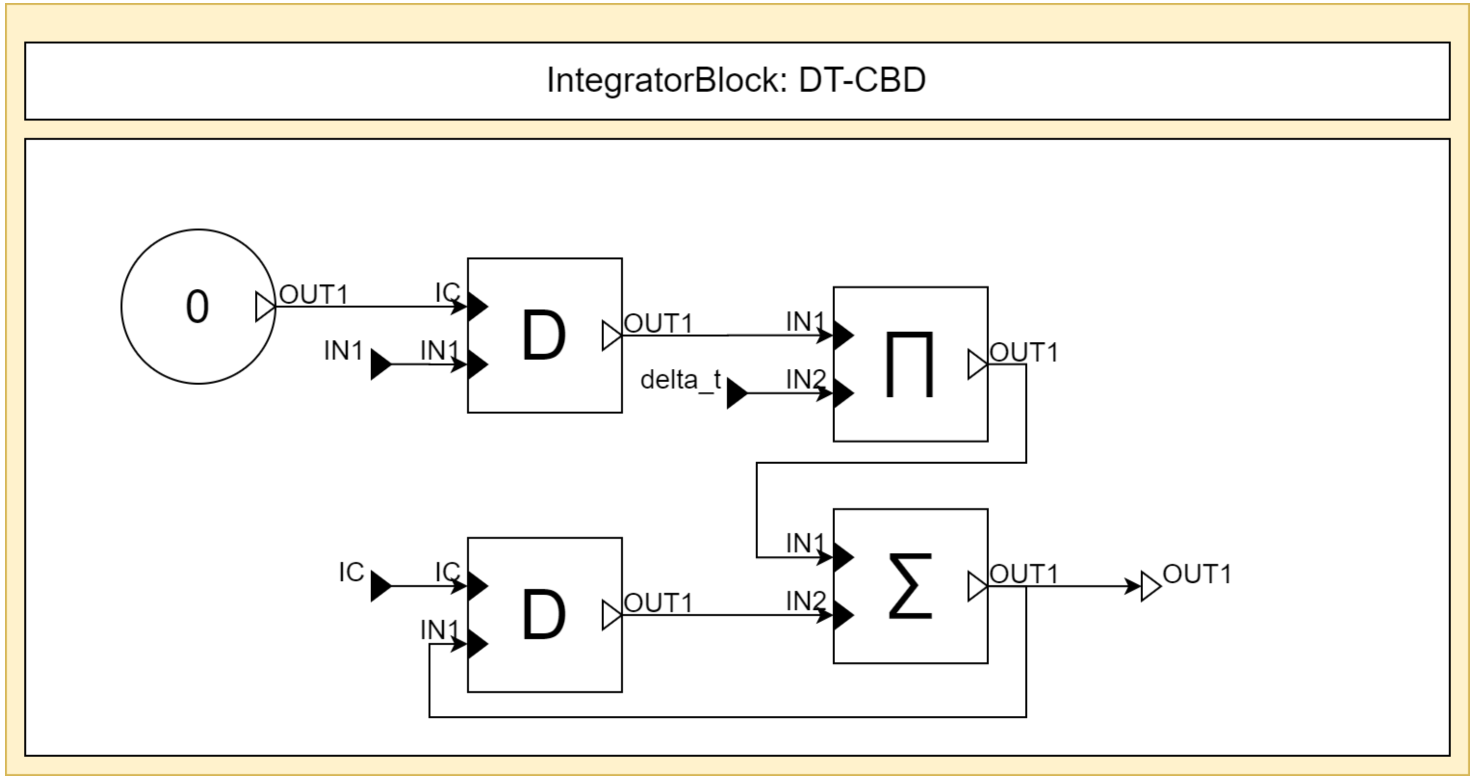
$$y(t) = \mathrm{y}(\mathrm{t} - \Delta t) + \Delta t \cdot x(t - \Delta t)$$

Simpson's Formula

$$y(t) = \mathrm{y}(\mathrm{t} - \Delta t) + \Delta t \cdot x(t)$$

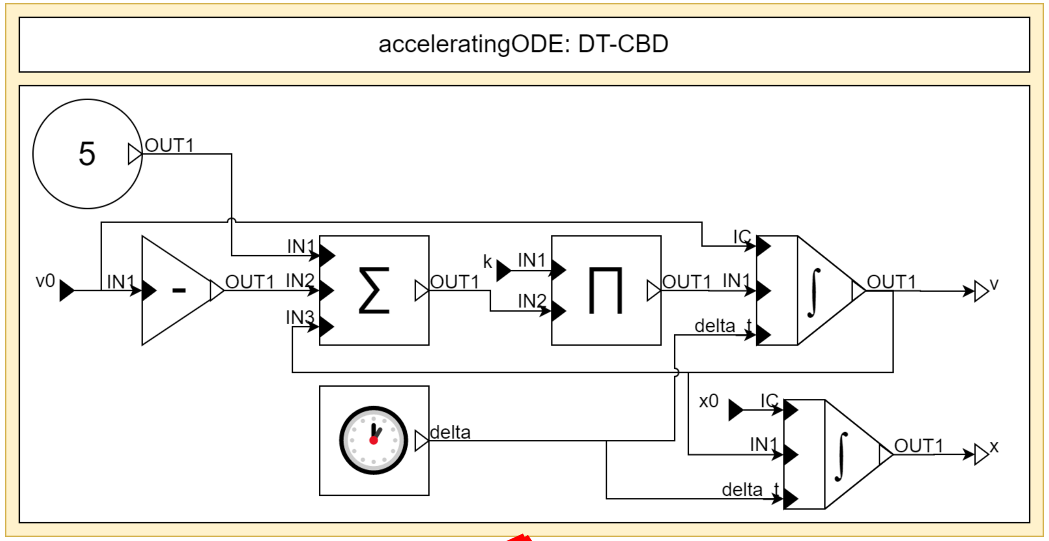
Runge-Kutta Methods

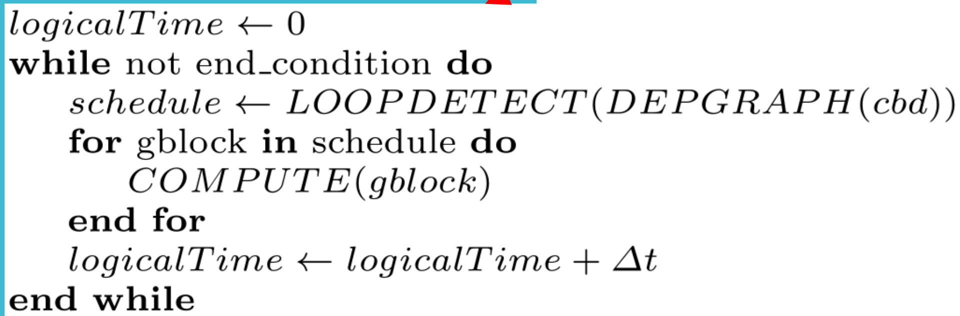




$$y(t) = y(\mathrm{t} - \Delta t) + \Delta t \cdot x(t - \Delta t)$$
$$y(0) = IC$$

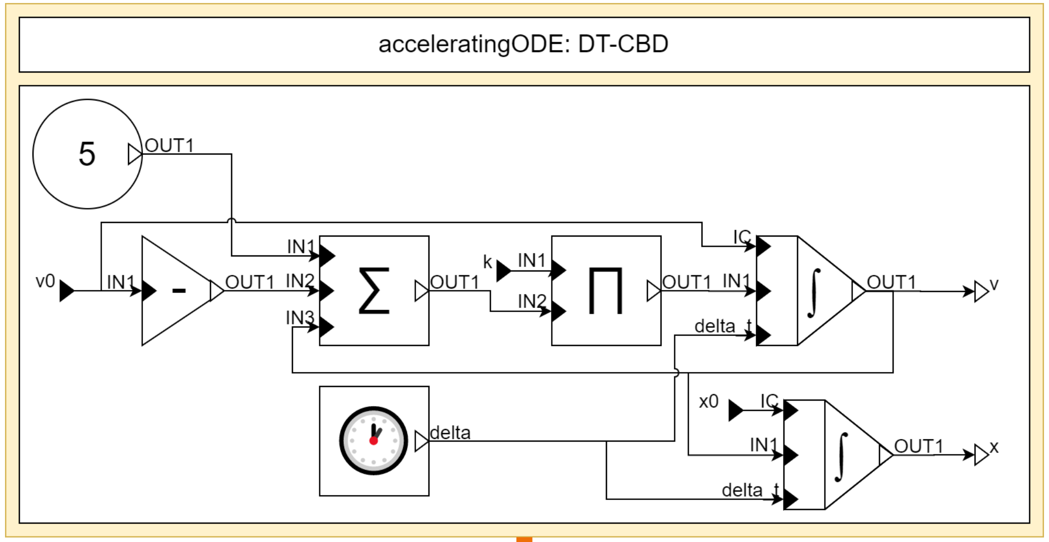
CBD Simulation



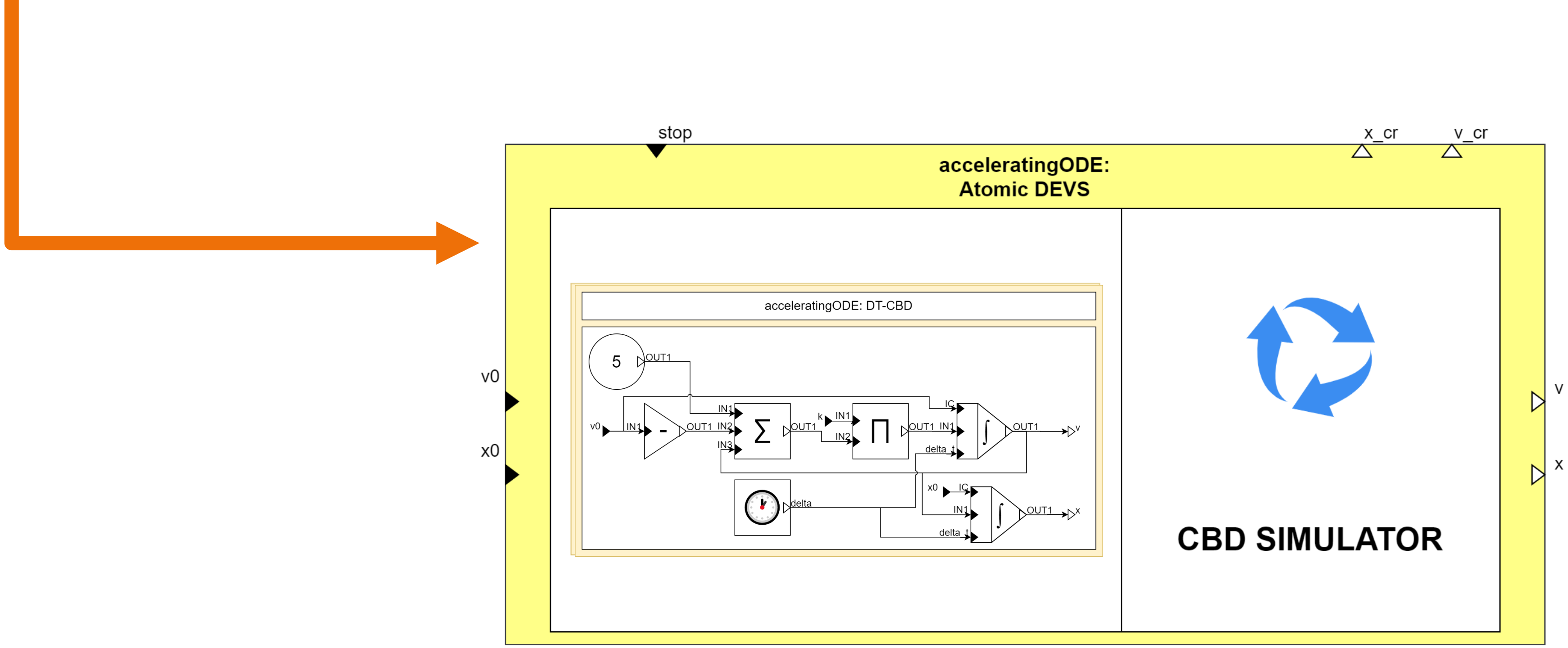


```
$logicalTime \leftarrow 0$
while not end_condition do
    $schedule \leftarrow LOOPDETECT(DEPGRAPH(cbd))$
    for gblock in schedule do
        $COMPUTE(gblock)$
    end for
    $logicalTime \leftarrow logicalTime + \Delta t$
end while
```

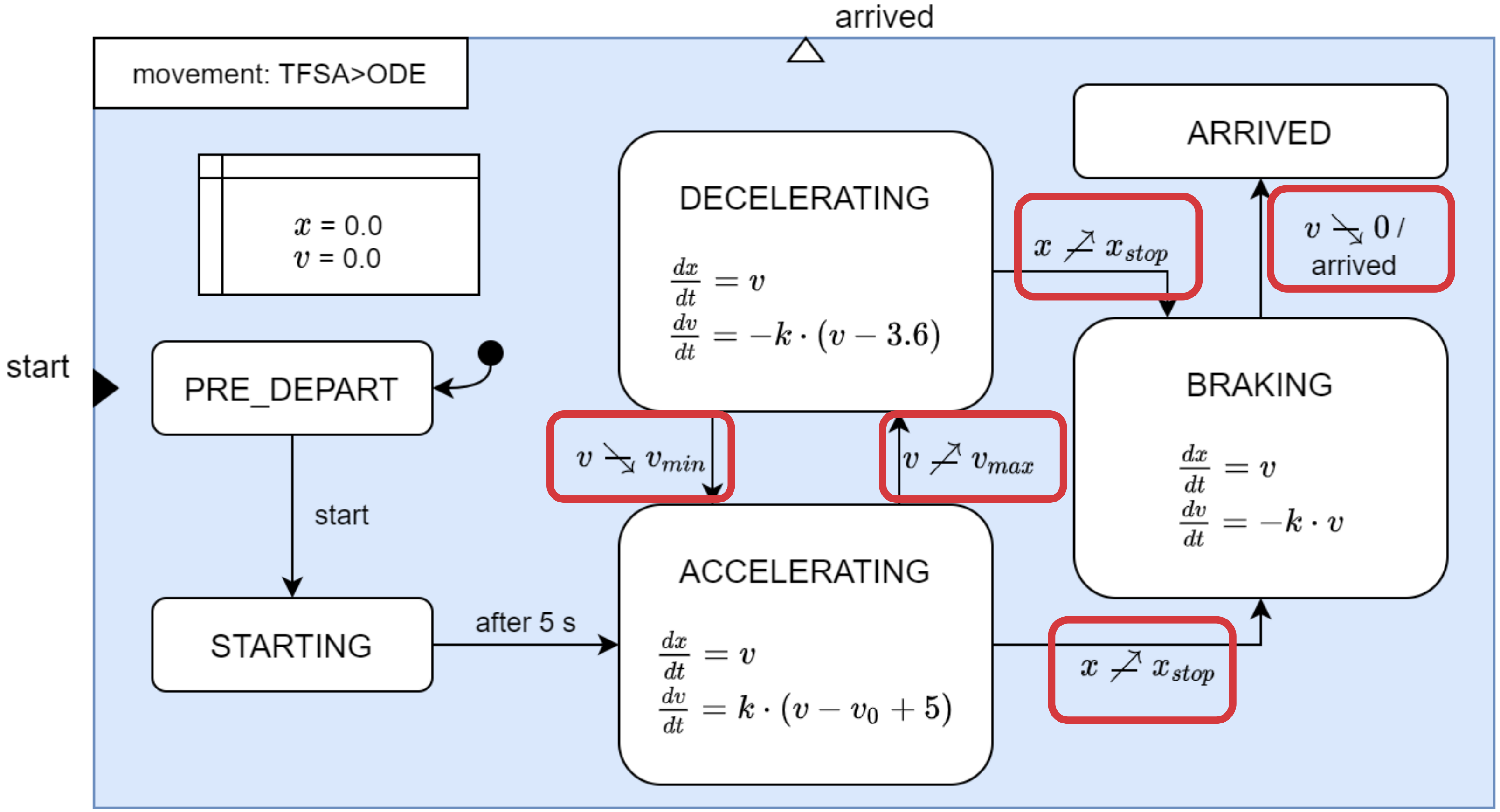
```
five.OUT1 = 5
sum.IN1 = five.OUT1
neg.IN1 = v0
neg.OUT1 = -neg.IN1
sum.IN2 = neg.OUT1
sum.IN3 = v_int.OUT1
sum.OUT1 = sum.IN1 + sum.IN2 + sum.IN3
prod.IN1 = k
prod.IN2 = sum.OUT1
prod.OUT1 = prod.IN1 * prod.IN2
int_v.IC = v0
int_v.IN1 = prod.OUT1
int_v.delta_t = delta
v = int_v.OUT1
int_x.IC = x0
int_x.IN1 = int_v.OUT1
int_x.delta_t = delta
x = int_x.OUT1
```



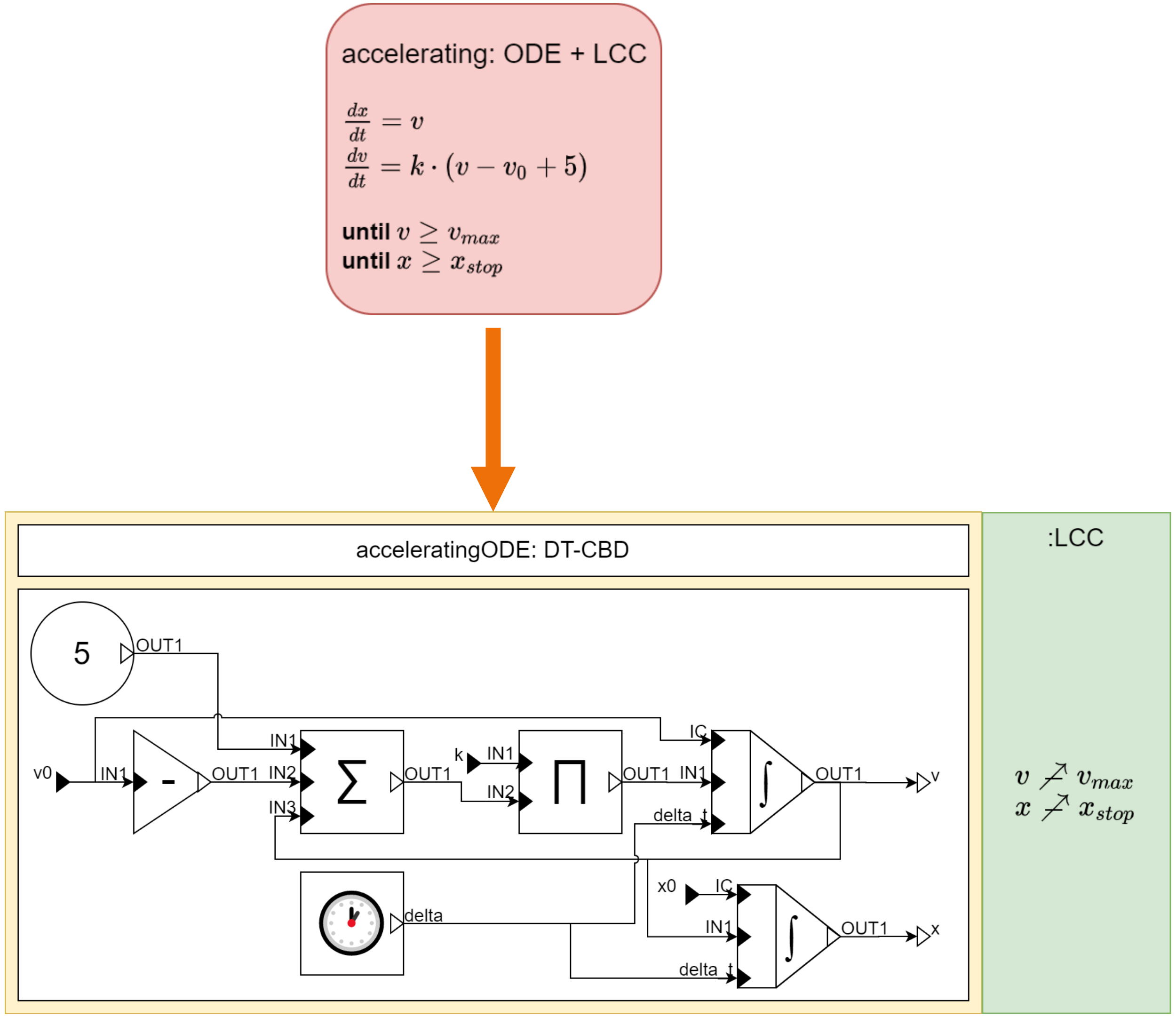
```
$logicalTime \leftarrow 0$
while not end_condition do
    $schedule \leftarrow LOOPDETECT(DEPGRAPH(cbd))$
    for gblock in schedule do
        $COMPUTE(gblock)$
    end for
    $logicalTime \leftarrow logicalTime + \Delta t$
end while
```



State Event Location



State Event Location



accelerating: ODE + LCC

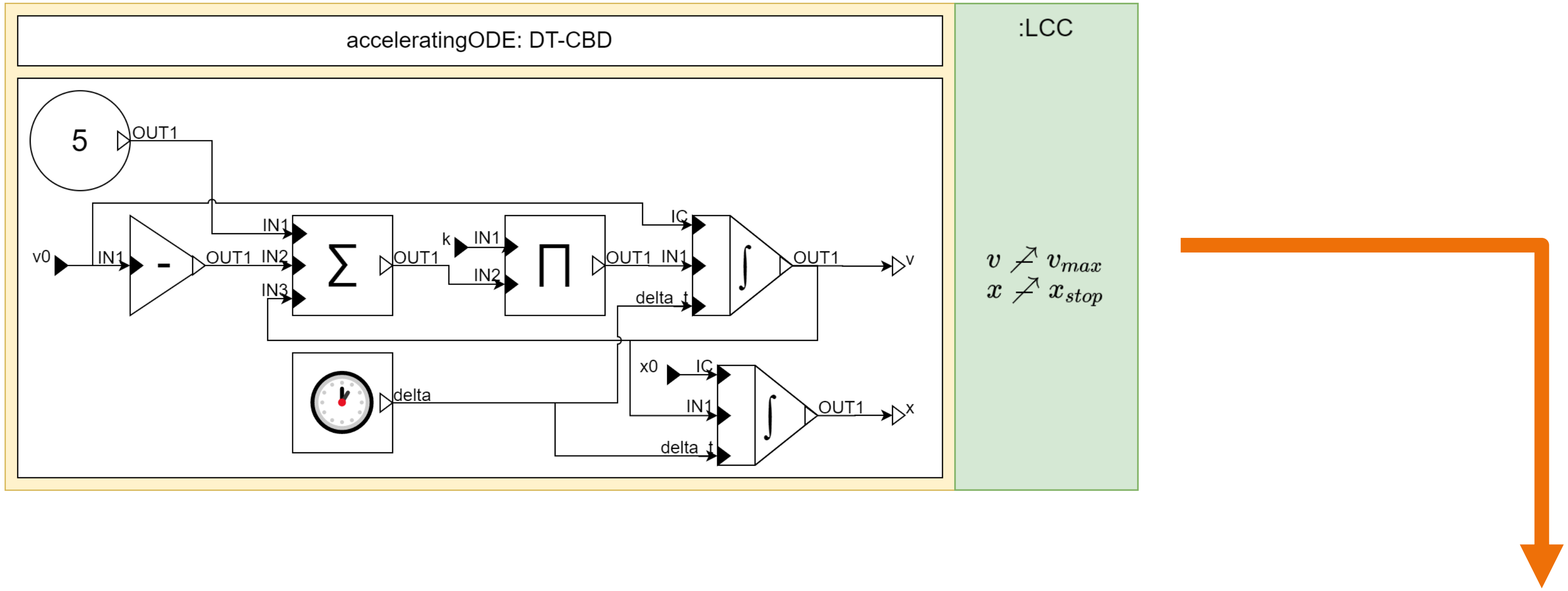
$$\frac{dx}{dt} = v$$
$$\frac{dv}{dt} = k \cdot (v - v_0 + 5)$$

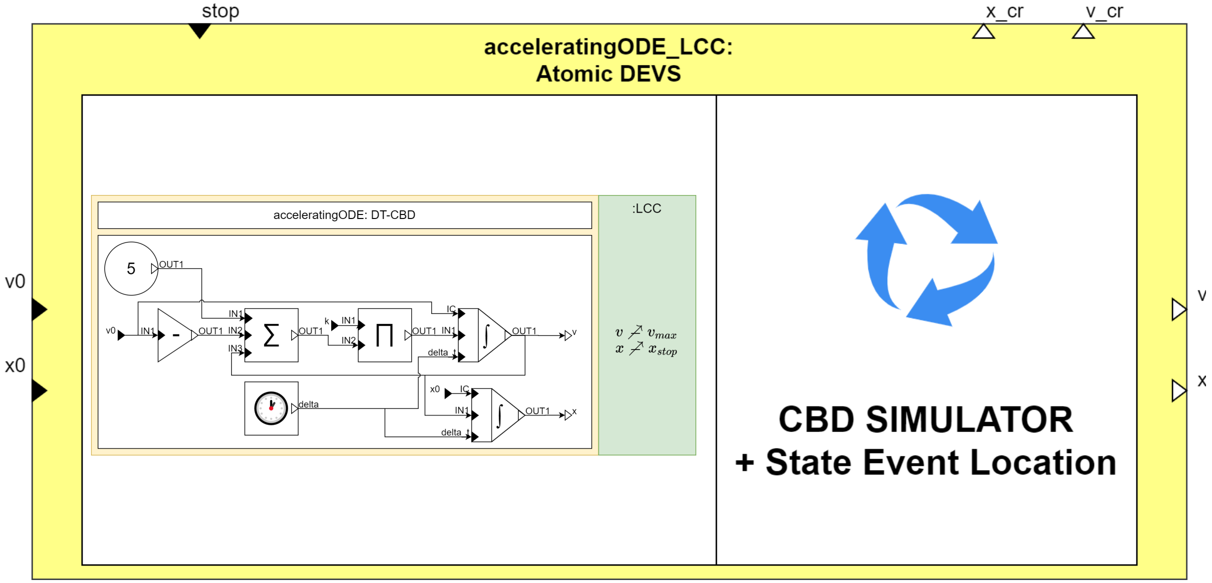
until $v \geq v_{max}$
until $x \geq x_{stop}$

acceleratingODE: DT-CBD

:LCC

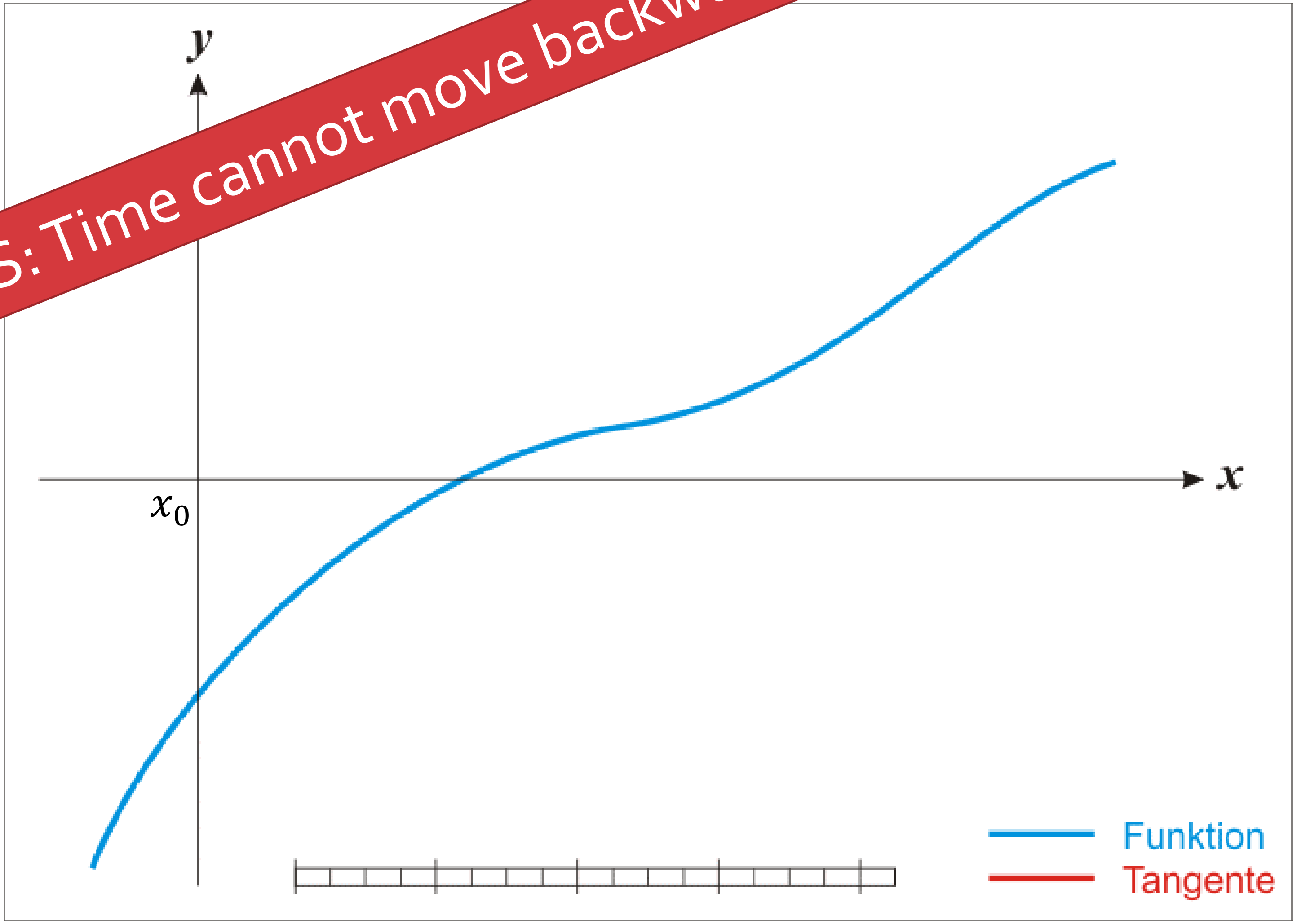
$v \not\nearrow v_{max}$
$x \not\nearrow x_{stop}$



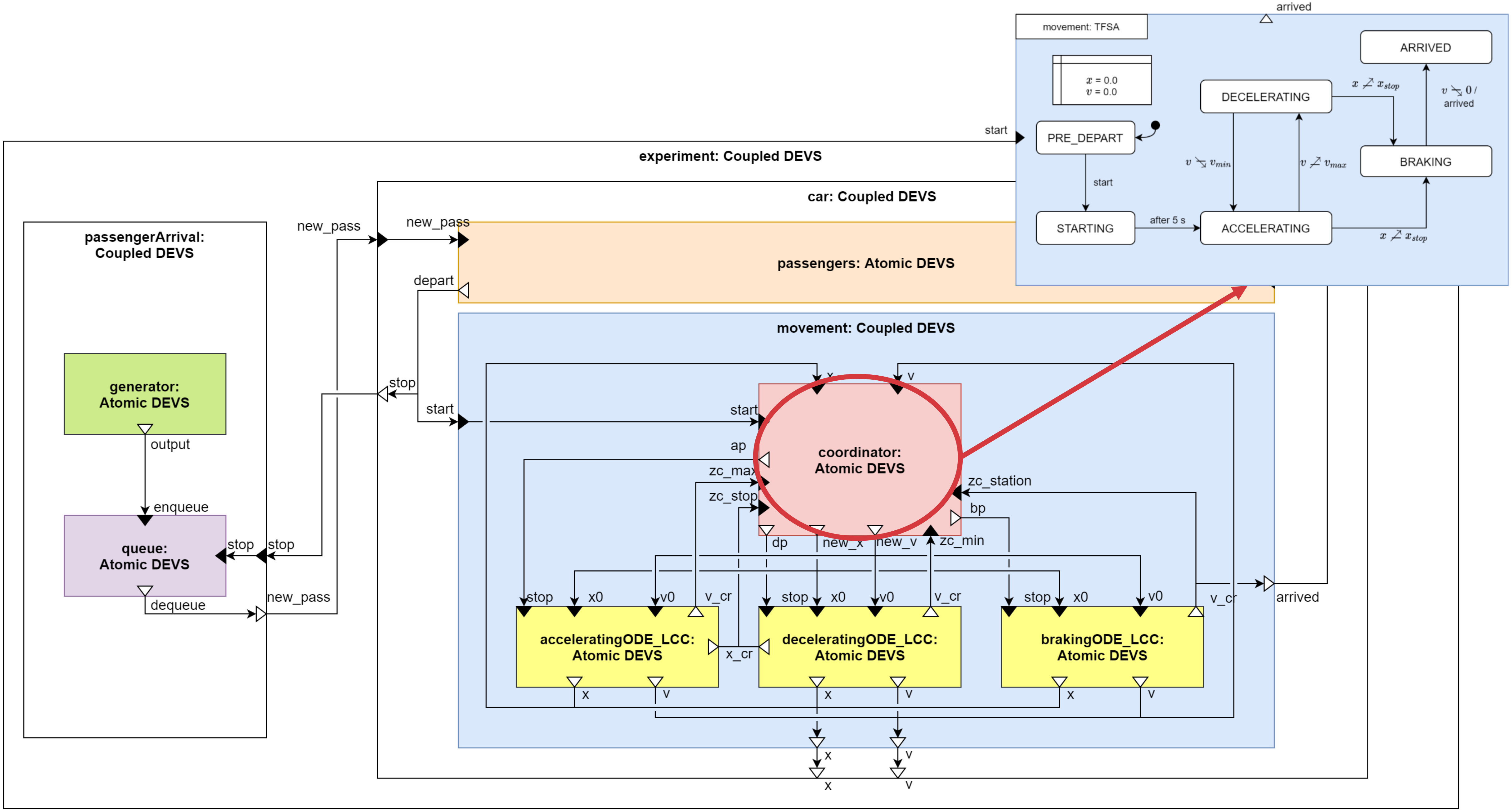


X = zero-order hold of inputs + "stop"
Y = zero-order hold of outputs + state event locations
S = set of all CBD states
δ_{int} runs next CBD simulation step **if** "computation"
δ_{ext} stops simulation, reinits and possibly restarts
λ, **ta**: see paper

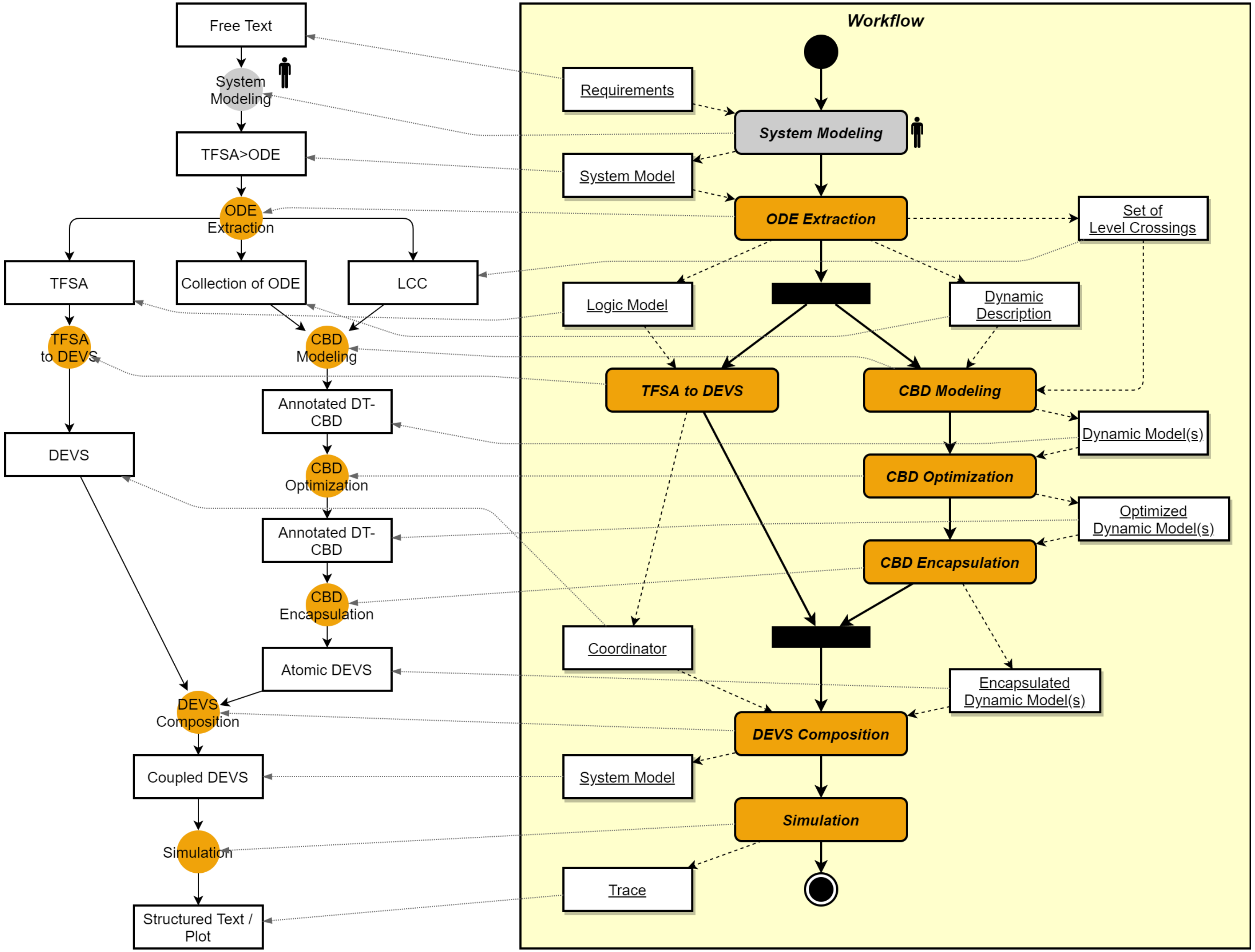
State Event Location



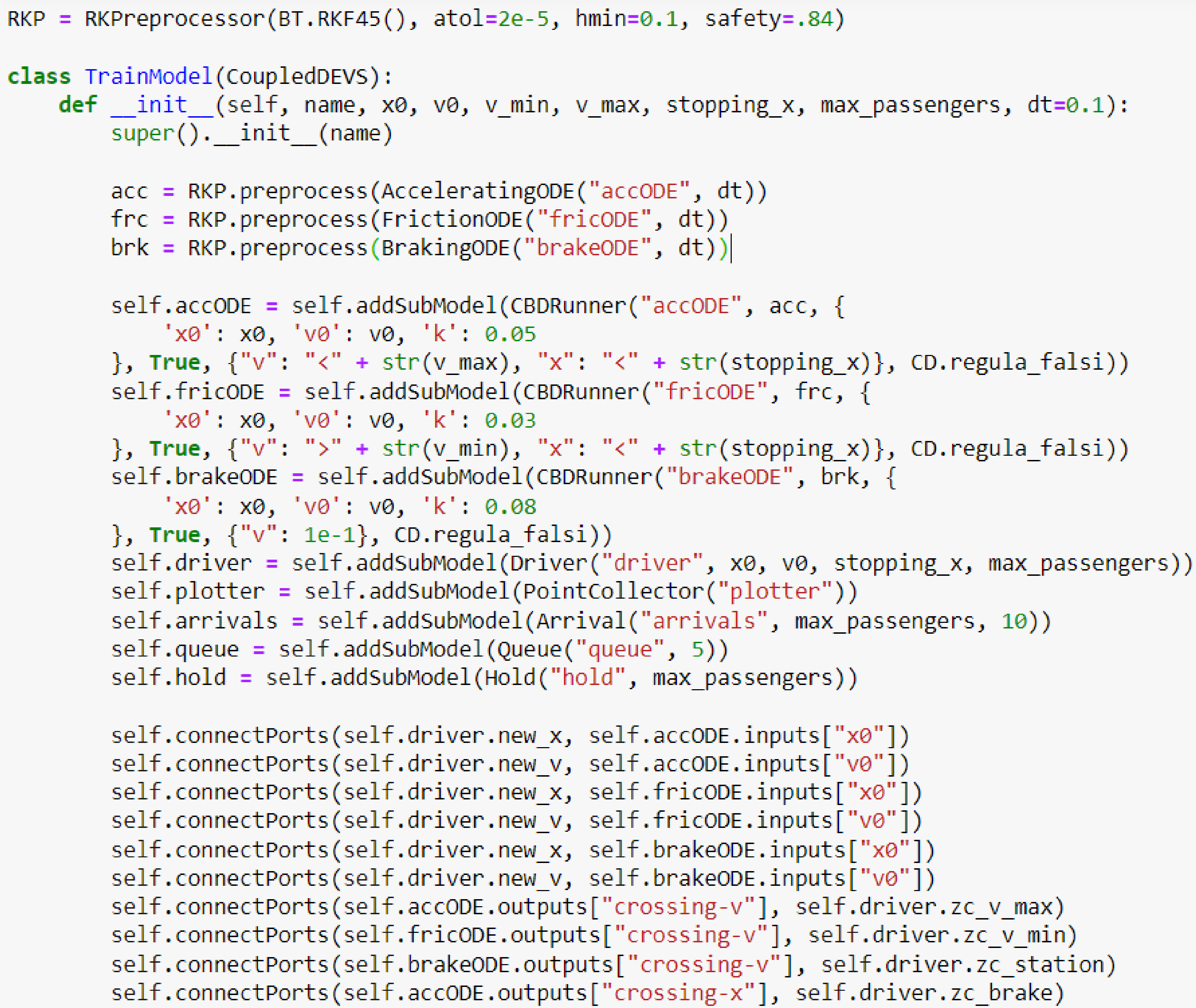
DEVS: Time cannot move backwards!


experiment: Coupled DEVS
passengerArrival: Coupled DEVS
generator: Atomic DEVS
output
enqueue
queue: Atomic DEVS
stop
stop
dequeue
new_pass
new_pass
new_pass
stop
depart
start
car: Coupled DEVS
passengers: Atomic DEVS
movement: Coupled DEVS
coordinator: Atomic DEVS
x
v
start
ap
zc_max
zc_stop
zc_station
bp
dp
new_x
new_v
zc_min
stop
x0
v0
v_cr
acceleratingODE_LCC: Atomic DEVS
x_cr
deceleratingODE_LCC: Atomic DEVS
brakingODE_LCC: Atomic DEVS
arrived
x
v
movement: TFSA
arrived
x = 0.0
v = 0.0
start
PRE_DEPART
start
STARTING
after 5 s
ACCELERATING
DECELERATING
$v \searrow v_{min}$
$v \nearrow v_{max}$
$x \nearrow x_{stop}$
$x \nearrow x_{stop}$
BRAKING
$v \searrow 0$ / arrived
ARRIVED

FTG+PM

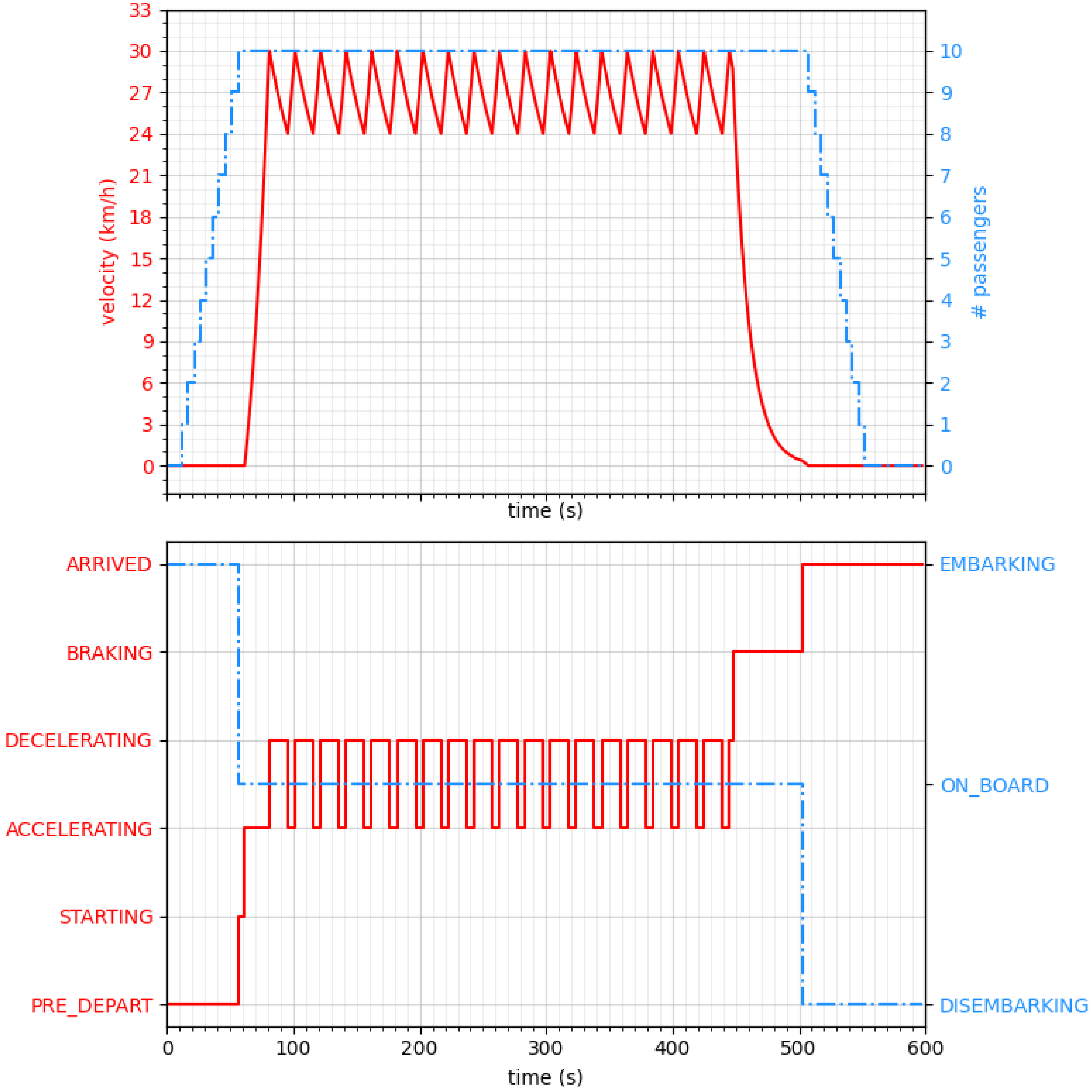


Mustafiz, S., J. Denil, L. Lúcio, & Vangheluwe, H. (2012) "*The FTG+PM Framework for Multi-Paradigm Modelling: an Automotive Case Study*". In *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling*, pp. 13–18. ACM.



```python
RKP = RKPreprocessor(BT.RKF45(), atol=2e-5, hmin=0.1, safety=.84)

class TrainModel(CoupledDEVS):
    def __init__(self, name, x0, v0, v_min, v_max, stopping_x, max_passengers, dt=0.1):
        super().__init__(name)

        acc = RKP.preprocess(AcceleratingODE("accODE", dt))
        frc = RKP.preprocess(FrictionODE("fricODE", dt))
        brk = RKP.preprocess(BrakingODE("brakeODE", dt))

        self.accODE = self.addSubModel(CBDRunner("accODE", acc, {
            'x0': x0, 'v0': v0, 'k': 0.05
        }, True, {"v": "<" + str(v_max), "x": "<" + str(stopping_x)}, CD.regula_falsi))
        self.fricODE = self.addSubModel(CBDRunner("fricODE", frc, {
            'x0': x0, 'v0': v0, 'k': 0.03
        }, True, {"v": ">" + str(v_min), "x": "<" + str(stopping_x)}, CD.regula_falsi))
        self.brakeODE = self.addSubModel(CBDRunner("brakeODE", brk, {
            'x0': x0, 'v0': v0, 'k': 0.08
        }, True, {"v": 1e-1}, CD.regula_falsi))
        self.driver = self.addSubModel(Driver("driver", x0, v0, stopping_x, max_passengers))
        self.plotter = self.addSubModel(PointCollector("plotter"))
        self.arrivals = self.addSubModel(Arrival("arrivals", max_passengers, 10))
        self.queue = self.addSubModel(Queue("queue", 5))
        self.hold = self.addSubModel(Hold("hold", max_passengers))

        self.connectPorts(self.driver.new_x, self.accODE.inputs["x0"])
        self.connectPorts(self.driver.new_v, self.accODE.inputs["v0"])
        self.connectPorts(self.driver.new_x, self.fricODE.inputs["x0"])
        self.connectPorts(self.driver.new_v, self.fricODE.inputs["v0"])
        self.connectPorts(self.driver.new_x, self.brakeODE.inputs["x0"])
        self.connectPorts(self.driver.new_v, self.brakeODE.inputs["v0"])
        self.connectPorts(self.accODE.outputs["crossing-v"], self.driver.zc_v_max)
        self.connectPorts(self.fricODE.outputs["crossing-v"], self.driver.zc_v_min)
        self.connectPorts(self.brakeODE.outputs["crossing-v"], self.driver.zc_station)
        self.connectPorts(self.accODE.outputs["crossing-x"], self.driver.zc_brake)
```

Simulation Trace



Future Work

- Optimizations
  - Adaptive Stepsize (and StEL coordination)
  - Symbolic Optimizations
  - Memoization (of simulation sub-results)
  - Parallelization (schedule)
- Traceability for Debugging
- Numerical Accuracy Study
- Co-Simulation (Architecture of Coupled ODE Models) ~ FMI

Questions / Discussion