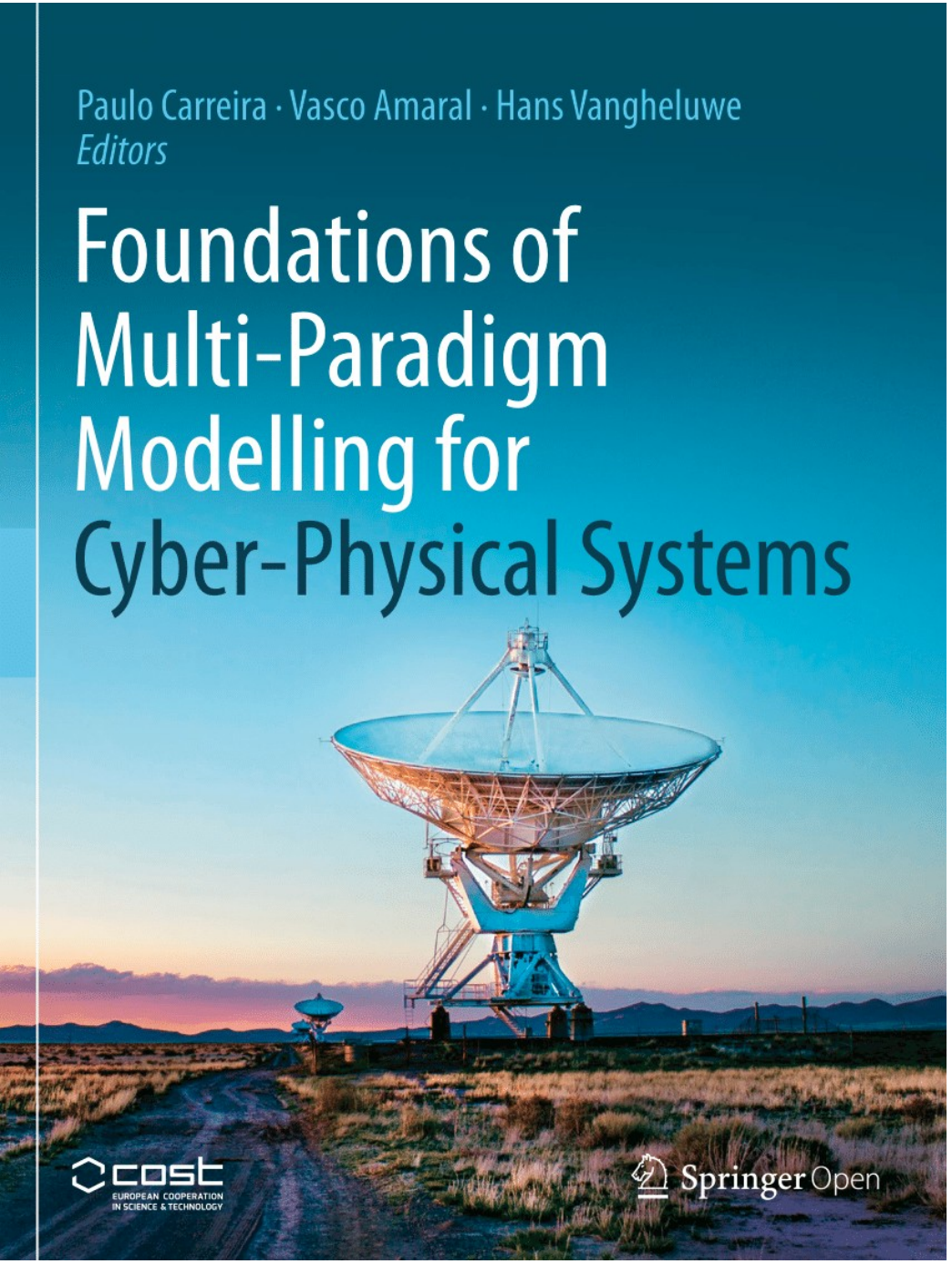
Causal Block Diagrams (CBDs) a family of formalisms

Hans Vangheluwe

Universiteit Antwerpen
Ansymo
Antwerp Systems & Software Modelling
University of Antwerp
Nexor
Cyber-Physical Systems
University of Antwerp
FLANDERS MAKE
MANUFACTURING INNOVATION NETWORK
MPM4CPS
cost

Physical Systems Modelling

- Problem-Specific (technological)
- Domain-Specific (e.g., translational mechanical)
- (general) Laws of Physics
- Power Flow/Bond Graphs (physical: energy/power)
- Computationally a-causal
  (Mathematical and Object-Oriented) ← **Modelica**
- Causal Block Diagrams (data flow)
- Numerical (Discrete) Approximations
- Computer Algorithmic + Numerical
  (Floating Point vs. Fixed Point)
- As-Fast-As-Possible vs. Real-time (XiL)
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)
- Dynamic Structure



Gomes C., Denil J., Vangheluwe H. (2020) Causal-Block Diagrams: A Family of Languages for Causal Modelling of Cyber-Physical Systems. In: Carreira P., Amaral V., Vangheluwe H. (eds) Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems. Springer, Cham.
https://doi.org/10.1007/978-3-030-43946-0_4

CBD CAUSAL BLOCK DIAGRAM

ABD A-CAUSAL BLOCK DIAGRAM

"COMPUTATIONAL CAUSALITY"

CAUSE ⟶ CONSEQUENCE

$+ : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

MATH

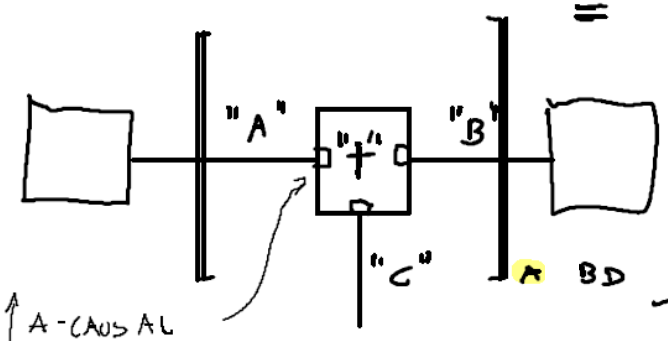
"WHAT"

$mvA + mvB + mvC = \phi$

$(mvA, mvB, mvC) \in \mathbb{R}^3$

A-CAUSAL

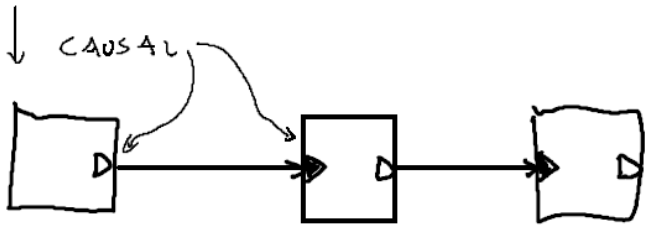
=

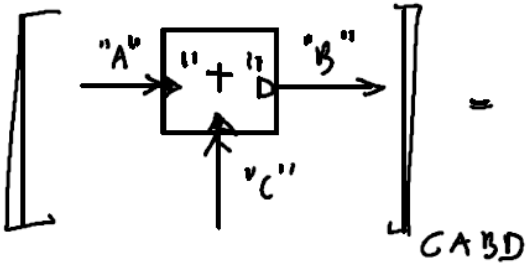


ABD

A-CAUSAL

CAUSALITY ASSIGNMENT





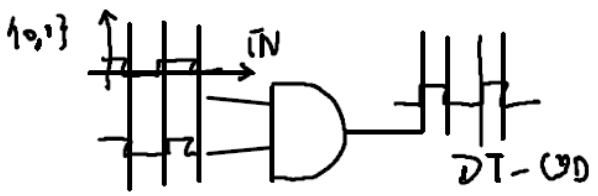
CABD

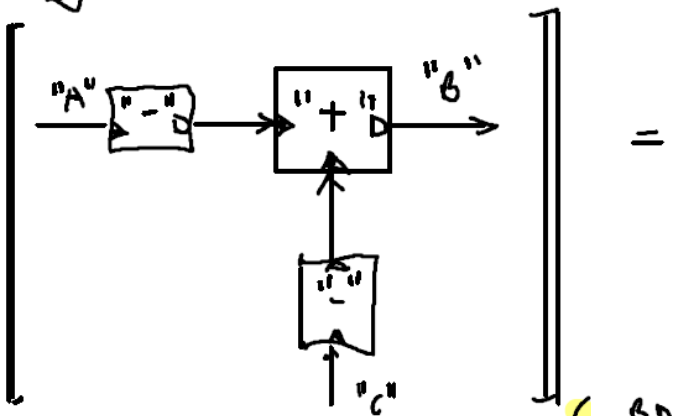
vA, vB, vC : float

vB := vA + vC

CODE

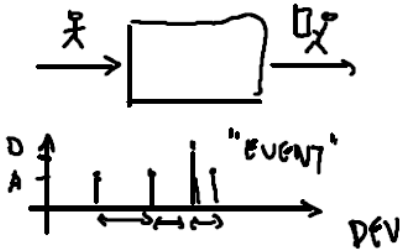
"HOW"



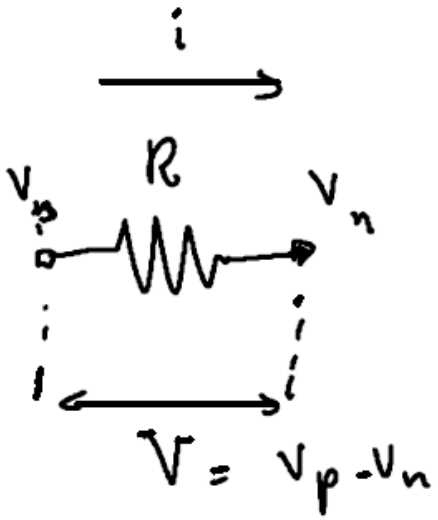


CBD

vB := - vA - vC



(computationally) a-causal vs. causal



$V := R \times i$

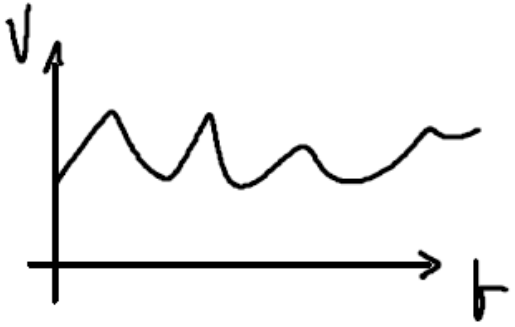
$R := V/i$

$i := V/R$

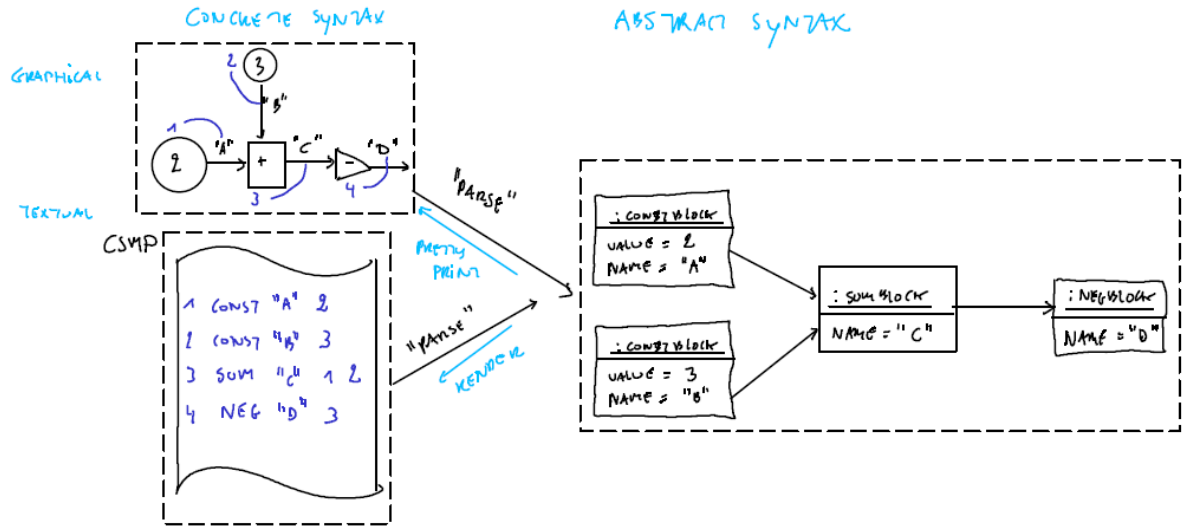
$V - Ri = \phi$

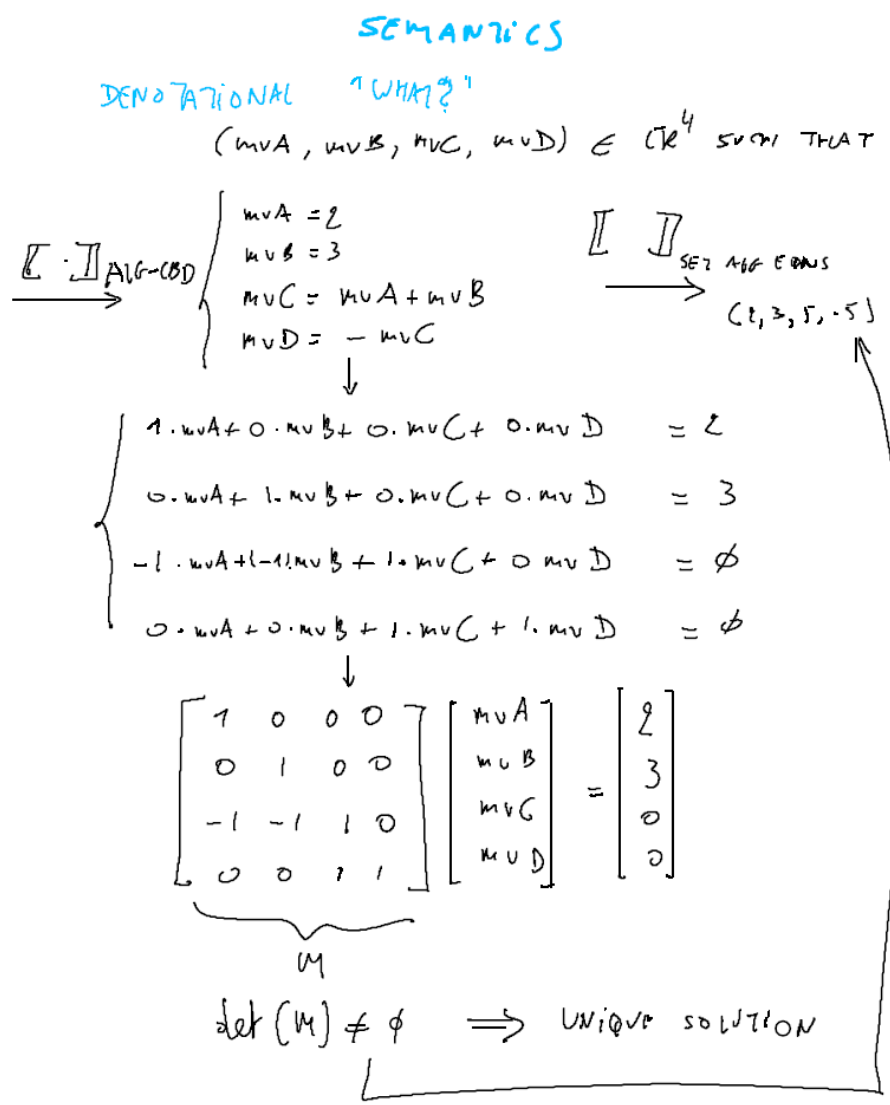
MODELICA ~~~> SIMULINK

CAUSALITY ASGN.

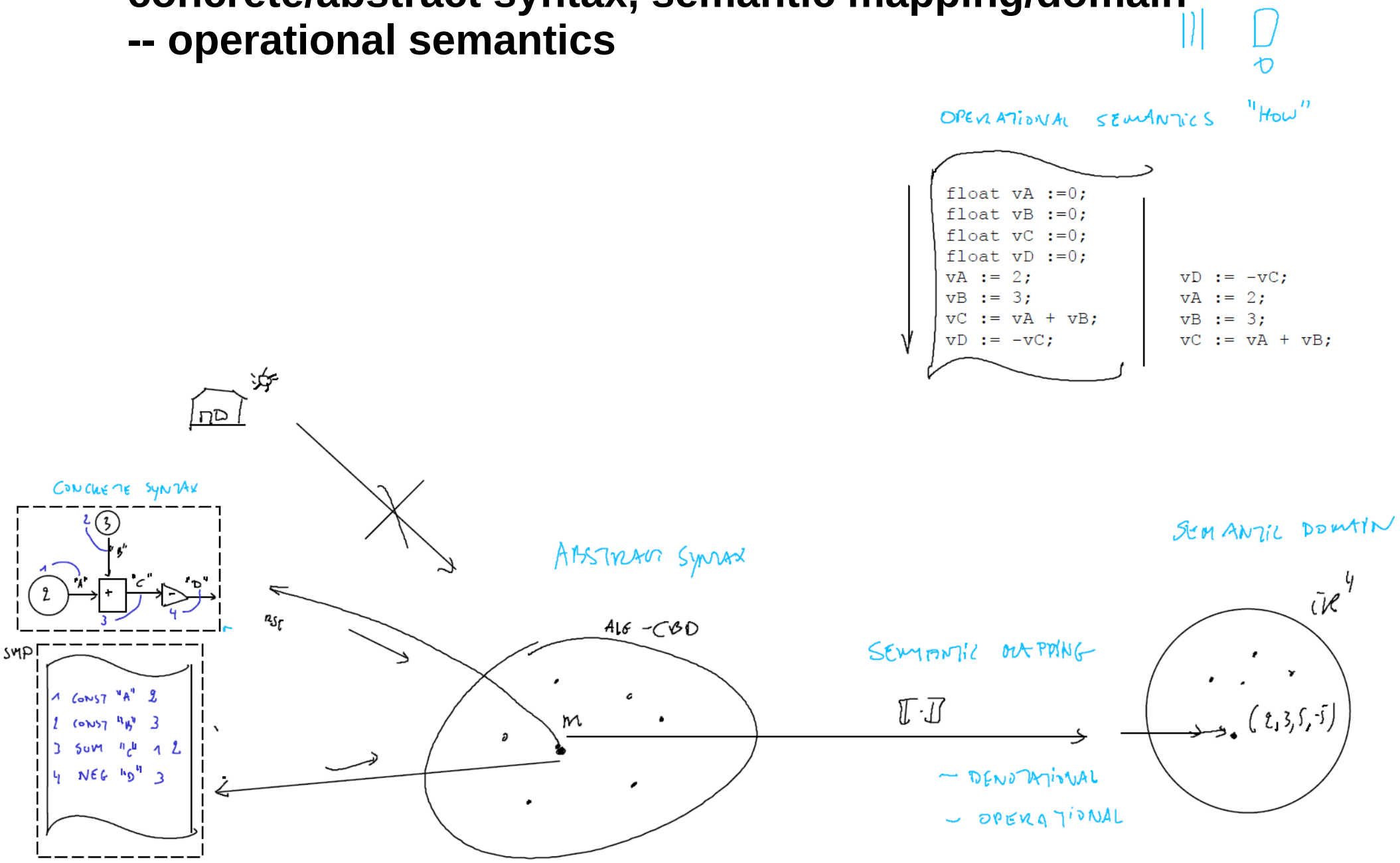


concrete/abstract syntax, semantic mapping/domain -- denotational semantics

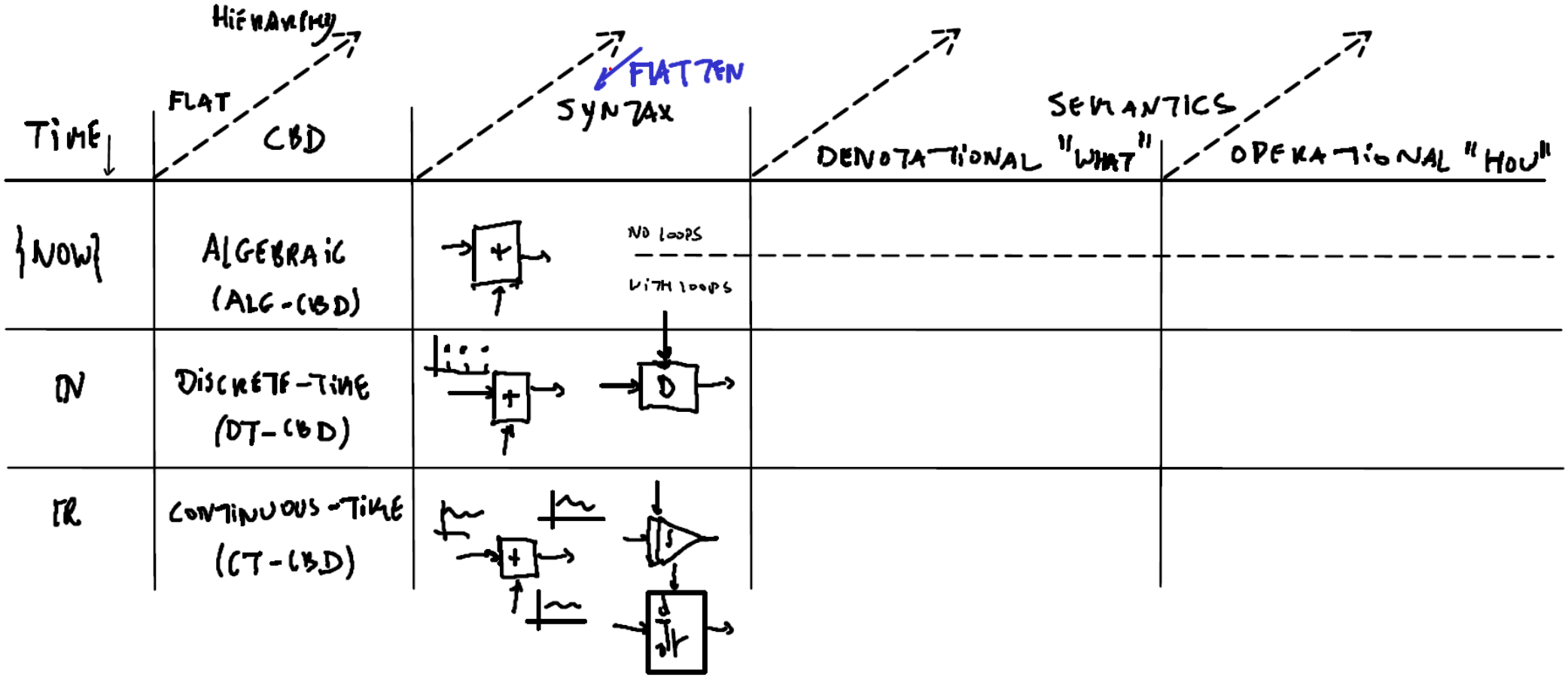




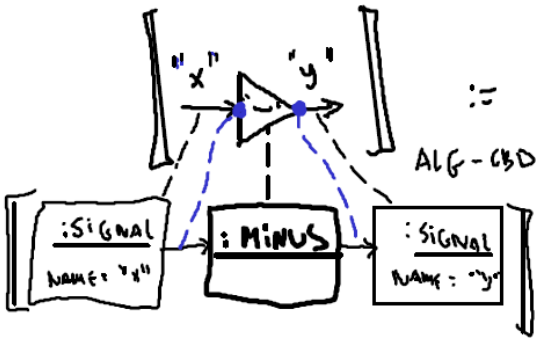
concrete/abstract syntax, semantic mapping/domain -- operational semantics



a family of Causal Block Diagram (CBD) formalisms



denotational semantics of individual blocks



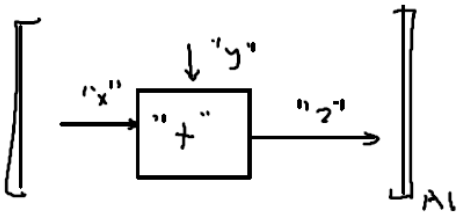
:=

$[\![\text{"y"}]\!] = -[\![\text{"x"}]\!]$

$mvy = -mvx$

$mvx, mvy \in \mathbb{R}$

$- : \mathbb{R} \to \mathbb{R}$



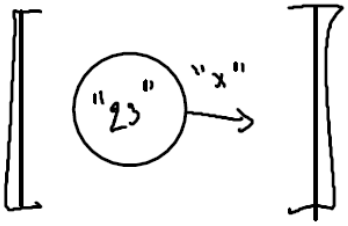
:=

$[\![\text{"z"}]\!] = [\![\text{"x"}]\!] + [\![\text{"y"}]\!]$

$mvz = mvx + mvy$

$mvx, mvy, mvz \in \mathbb{R}$

$+ : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$

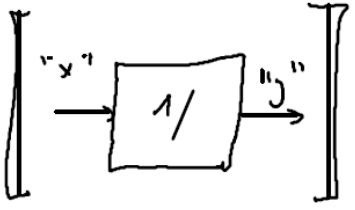


:=

$[\![\text{"x"}]\!] = [\![\text{"23"}]\!]$

$mvx = 23$

$mvx \in \mathbb{R}$



:=

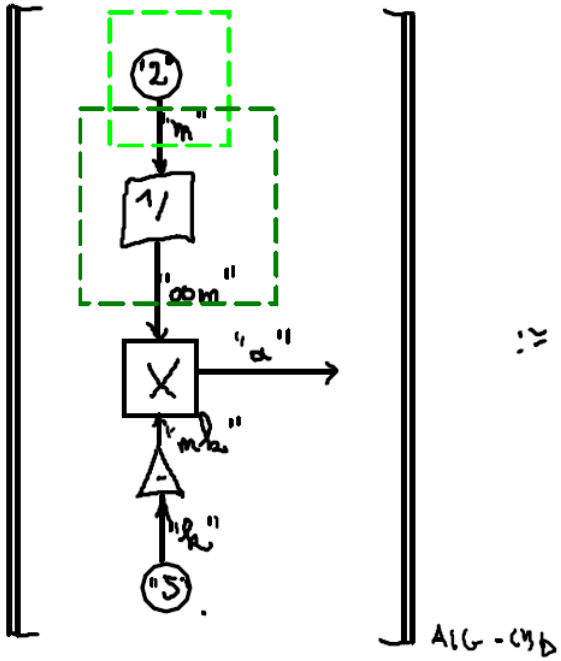
$[\![\text{"y"}]\!] = 1/[\![\text{"x"}]\!]$

$mvy = 1/mvx$

$mvx \in \mathbb{R} \setminus \{0\}, mvy \in \mathbb{R}$

$mvx, mvy \in \mathbb{R} \cup \{\text{EXCEPT}\}$

denotational semantics of “composition”



$:\simeq$ ALG-CMP

$m_{VM}, m_{VOOM}, m_{VM}, m_{VA}, m_{VMK}, m_{VK} \in \mathbb{R}$

SUCH THAT $m_{VM} \neq 0$

$m_{VM} = 2$

$m_{VOOM} = 1/m_{VM}$

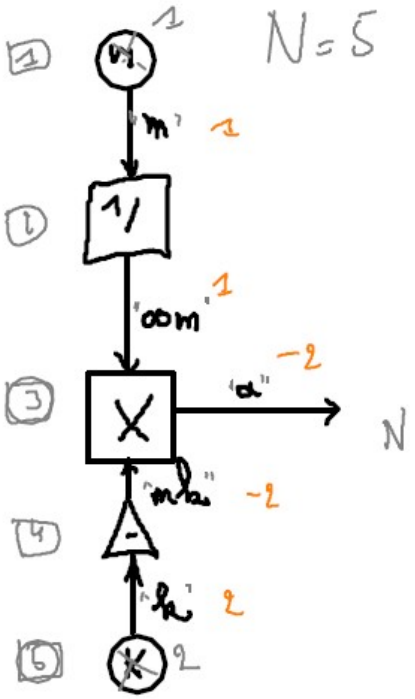
$m_{VA} = m_{VOOM} \times m_{VMK}$

$m_{VMK} = -m_{VK}$

$m_{VK} = 5$

$= (2, 0.5, 2.5, -5, 5) \in \mathbb{R}^5$

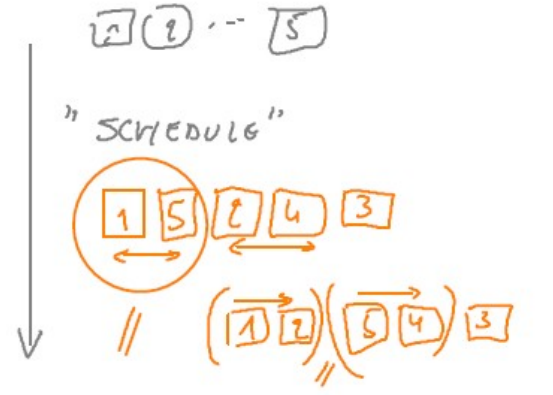
operational semantics
naive/optimal approach



	vm	vcom	va	vmk	vk
	UK	UK	UK	UK	UK
1	1	UK	UK	UK	UK
2	1	1	UK	UK	UK
3	1	1	UK	UK	UK
4	1	1	UK	UK	UK
5	1	1	UK	UK	2
1	1	1	UK	UK	2
2	1	1	UK	UK	2
3	1	1	UK	UK	2
4	1	1	UK	-2	2
5	1	1	UK	-2	2
1	1	1	UK	-2	2
2	1	1	UK	-2	2
3	1	1	-2	-2	2
4					
5					

N

N



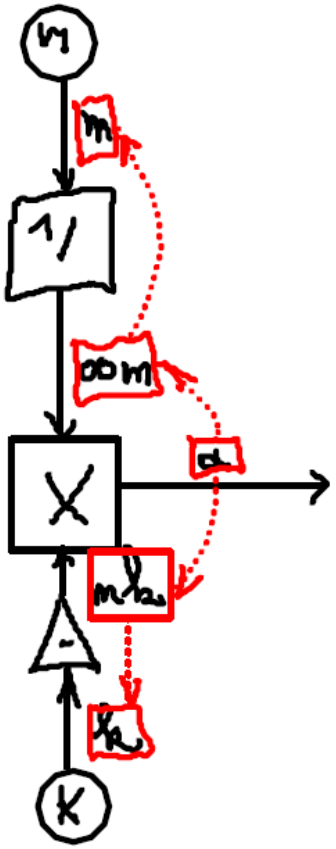
WORST-CASE

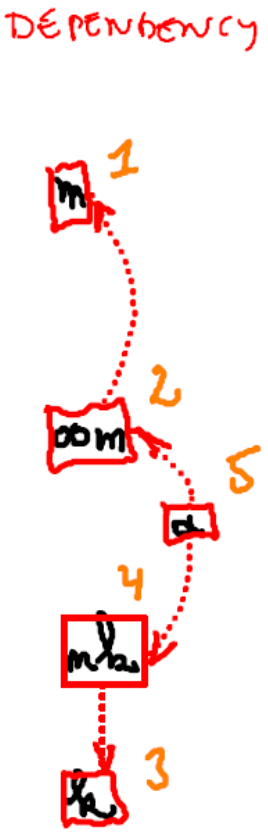
ITER $\Theta(N)\ \Theta(1)$

COMP $\Theta(N^2)\ \Theta(N)$

DISTRIBUTED

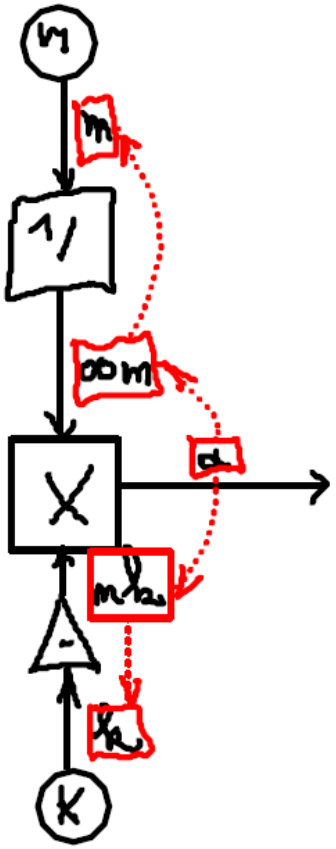
operational semantics scheduling





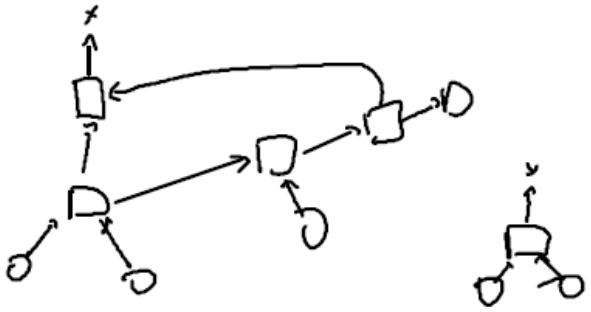
schedule = [m, oom, k, mk, a]

operational semantics scheduling



DEPENDENCY GRAPH

schedule = [m, oom, k, mk, a]



```
operational semantics
```

$O(N)$

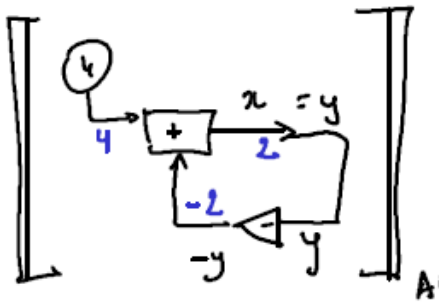
```
depGraph = buildDepGraph(CBD)
schedule = topologicalSort(depGraph)
```

$O(N)$

```
for block in schedule:
  block.compute()
```

$O(N)$

operational semantics
cyclic dependency
aka algebraic loop



:= ALG-COD

$$\begin{cases} x + y = 4 \\ x - y = 0 \end{cases} \qquad \begin{matrix} x = 4 - y \\ y = x \end{matrix}$$

SET LINEAR EQNS.

2 UNKNOWNS

2 EQNS

$$\begin{cases} 1 \times x + 1 \times y = 4 \\ 1 \times x - 1 \times y = 0 \end{cases}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

$$\det\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = -2 \neq 0$$

$$x = \frac{\begin{vmatrix} 4 & 1 \\ 0 & -1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix}} = \frac{-4}{-2} = 2$$

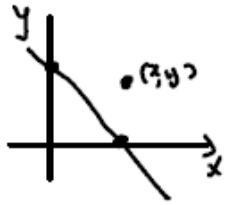
$$y = \frac{\begin{vmatrix} 1 & 4 \\ 1 & 0 \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix}} = \frac{-4}{-2} = 2$$

under-determined set of equations

→ **solution is sub-space**

$$\begin{cases} 2x + 2y = 4 \\ x + y = 2 \end{cases}$$

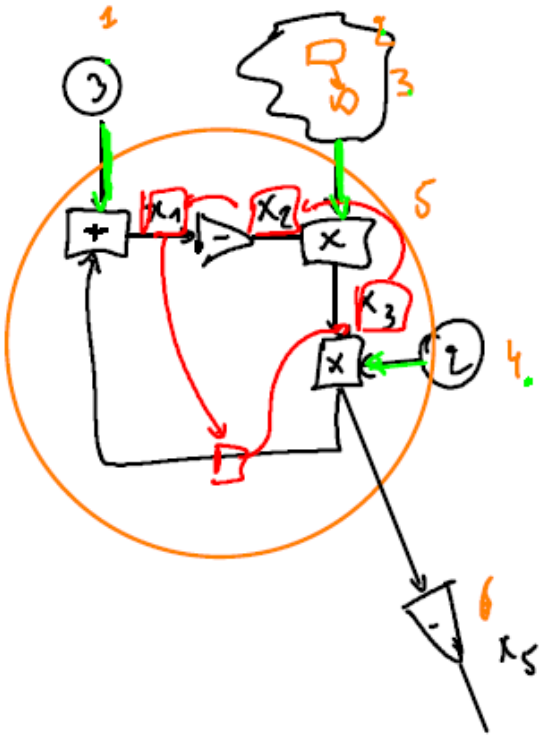
$$y = 2 - x$$



$$\begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

$$\begin{vmatrix} 2 & 2 \\ 1 & 1 \end{vmatrix} = \phi$$

“solving” a linear loop



$$\begin{cases} k_{11}x_1 + k_{21}x_2 + \dots + k_{n1}x_n = K_1 \\ k_{12}x_1 + k_{22}x_2 + \dots + k_{n2}x_n = K_2 \\ \vdots \\ k_{1n}x_1 + k_{2n}x_2 + \dots + k_{nn}x_n = K_n \end{cases}$$

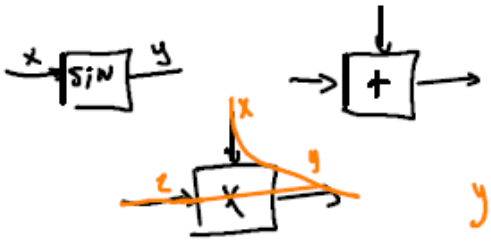
$$\underbrace{\begin{bmatrix} k_{11} & \cdots & k_{1n} \\ \vdots & & \vdots \\ k_{n1} & \cdots & k_{nn} \end{bmatrix}}_{k} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \underbrace{\begin{bmatrix} K_1 \\ \vdots \\ K_n \end{bmatrix}}_{K}$$

unique? $\det(k) \neq 0$

$(x_1, \dots, x_n)$

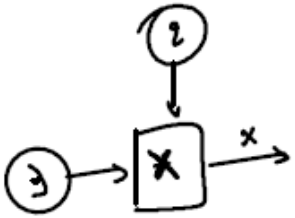
GAUSSIAN ELIM $\Theta(n^3)$

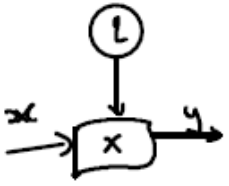
BLOCKS IN "LINEAR LOOP"

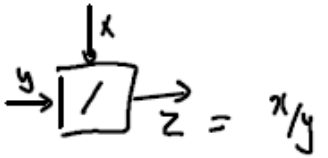


$y = x \times z$

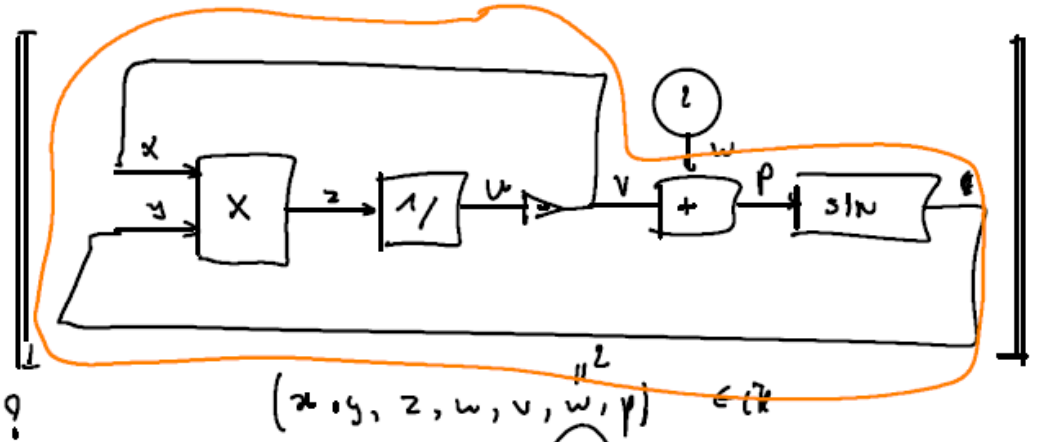
$xy - x \times z = 0$







non-linear loop



UNIQUE ?

COMPLEXITY ? $\mathcal{O}(..)$

$(x, y, z, w, v, w, p) \in \mathbb{R}^7$

$x \cdot y - z = 0$

$z - 1/w = 0$

$\vdots$

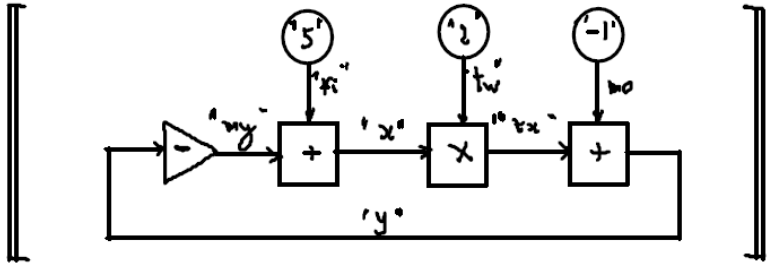
φ

$(x_0, y_0, z_0, w_0, v_0, w_0, p_0)$

$(x_{n-1}, y_{n-1}, z_{n-1}, w_{n-1}, v_{n-1}, w_{n-1}, p_{n-1})$

how to find an algebraic loop?

"algebraic loop"



$:=$

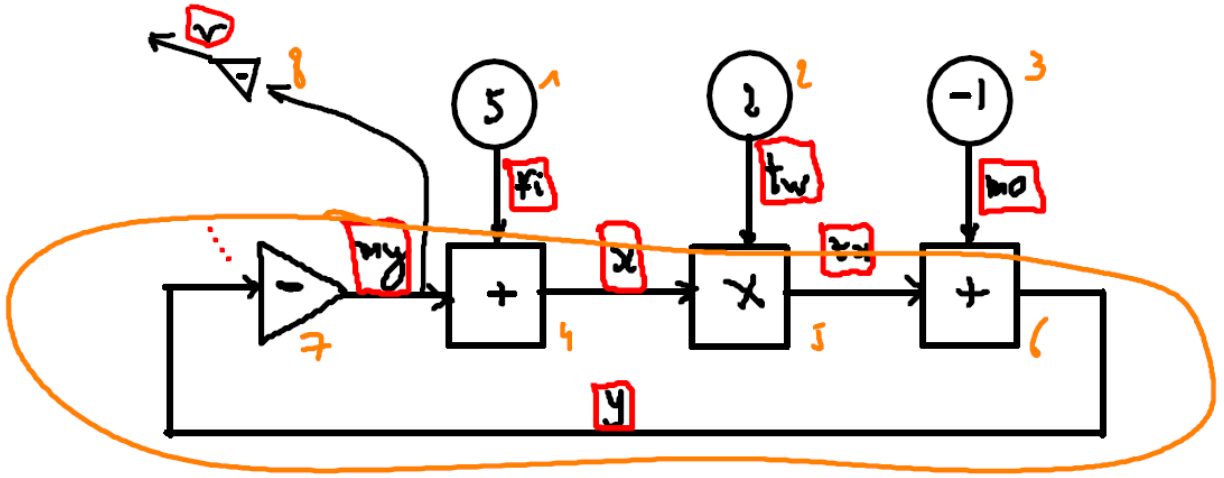
ALG-CBD

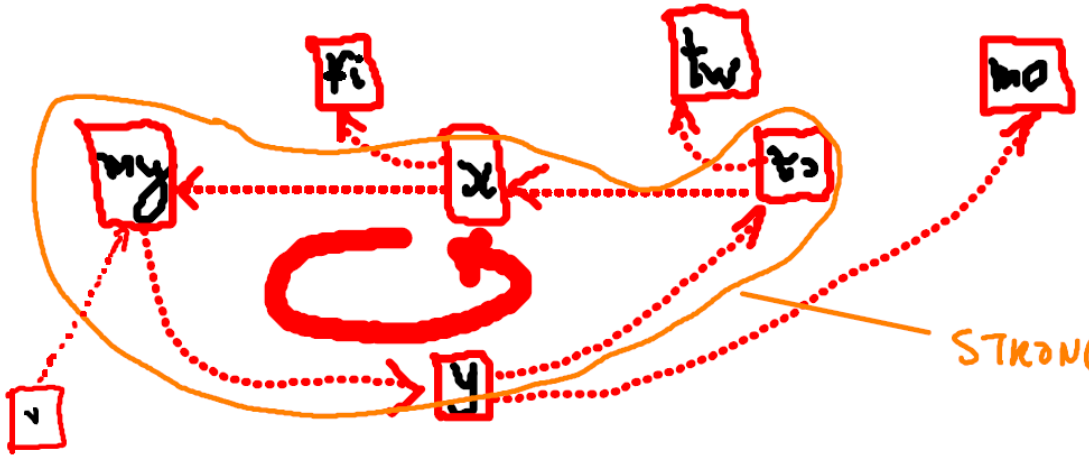
$mvfi, mvx, mvtw, mvtx, mvno, mvy, mvny \in \mathbb{R}$

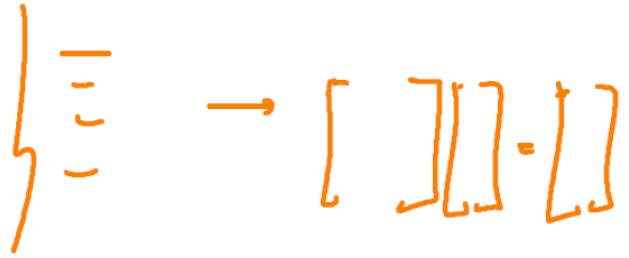
such that

$$\begin{cases} mvfi = 5 \\ mvx = mvfi + mvny \\ mvtw = 2 \\ mvtx = mvtw \times mvx \\ mvno = -1 \\ mvy = mvno + mvx \\ mvny = -mvy \end{cases} \simeq (5, 2, 2, 4, -1, 3, -3) \in \mathbb{R}^7$$

how to find an algebraic loop?







[{1},{2},{3}, {4,5,6,7}, {8}]

M

STRONG COMPONENT

TARJAN 1974

O(N+E)

operational semantics

```
schedule = topologicalSortAndLoopDetect(depGraph(CBD))
```

$\Theta(N+E)$

```
for genBlock in schedule:
  genBlock.compute()
```

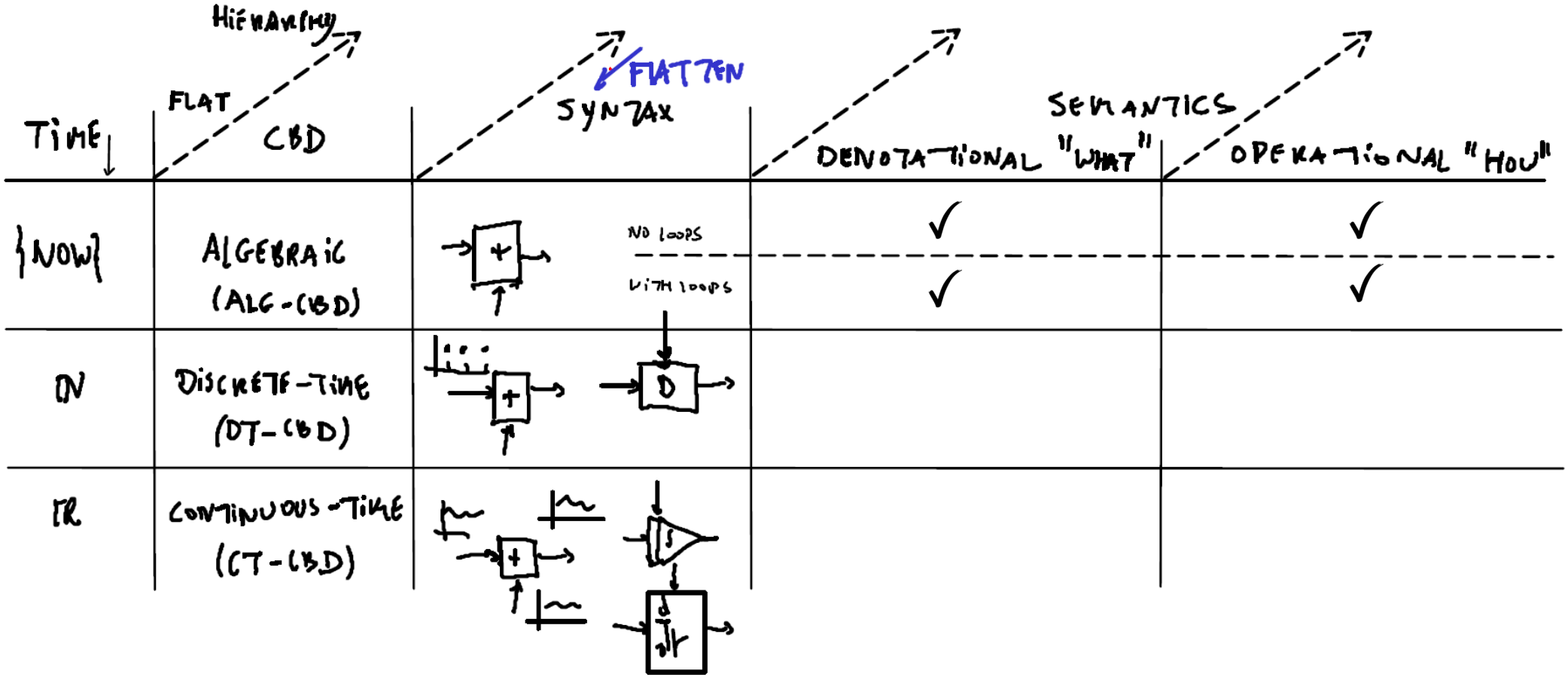
$\Theta(N)$!

some sets of coupled eqns (size M)

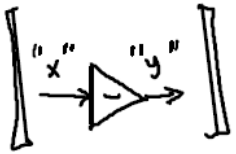
solver complexity $\leq \Theta(M^3)$

usually $M \ll N$

a family of Causal Block Diagram (CBD) formalisms



DISCRETE-TIME CBD (DT-CBD)



$:= \quad [\![y]\!] = -[\![x]\!]$ $\qquad mvy(i) = -_{\mathbb{R}} vx(i), \ \forall i \in \mathbb{N}_\phi$

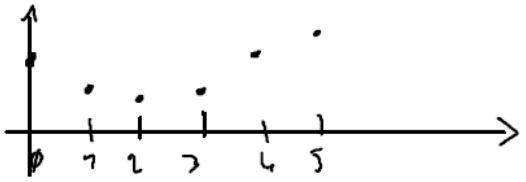
$mvx, mvy \in \mathbb{N}_\phi \to \mathbb{R}$

$- : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

$x_i \qquad i \in \mathbb{N}_\phi$

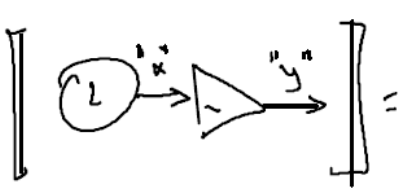
$x(i)$

mvx



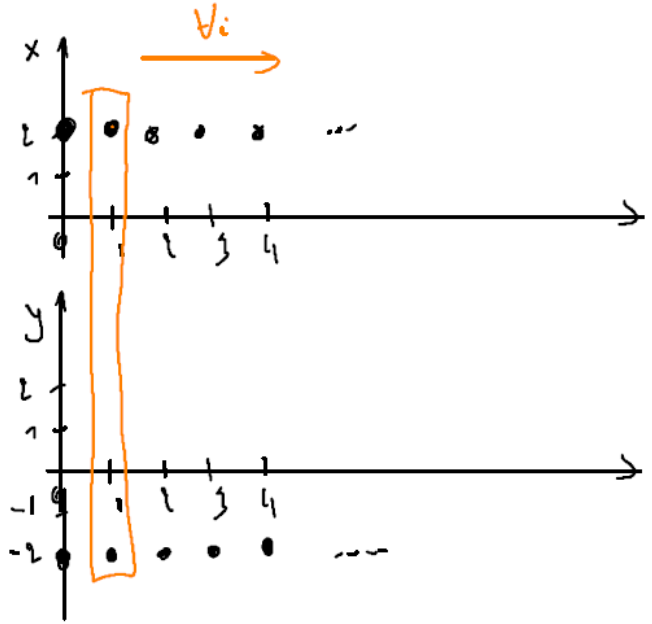
$mvx = \{ \phi \to 10.3, \ 1 \to 5.5, \ 2 \to 3.33, \ \dots \}$

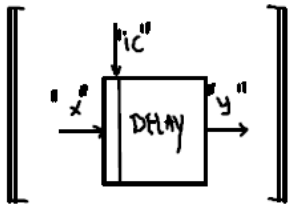
$[(\phi, 10.3), (1, 5.5), (2, 3.33), \dots]$



$x(i) = 2$

$y(i) = -x(i) \qquad \forall i \in \mathbb{N}_\phi$

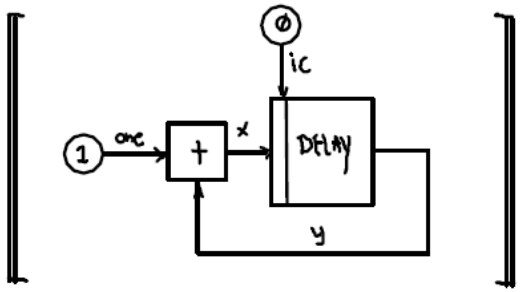




$:=$

$$\begin{cases} \nu_y(i) = \nu_x(i-1), & \forall i \in \mathbb{N} \\ \nu_y(\phi) = \nu_{ic}(\phi) \end{cases}$$

$\nu_{IC}, \nu_y, \nu_x, \nu_{one} \in \mathbb{N}_\phi \to \mathbb{R}$

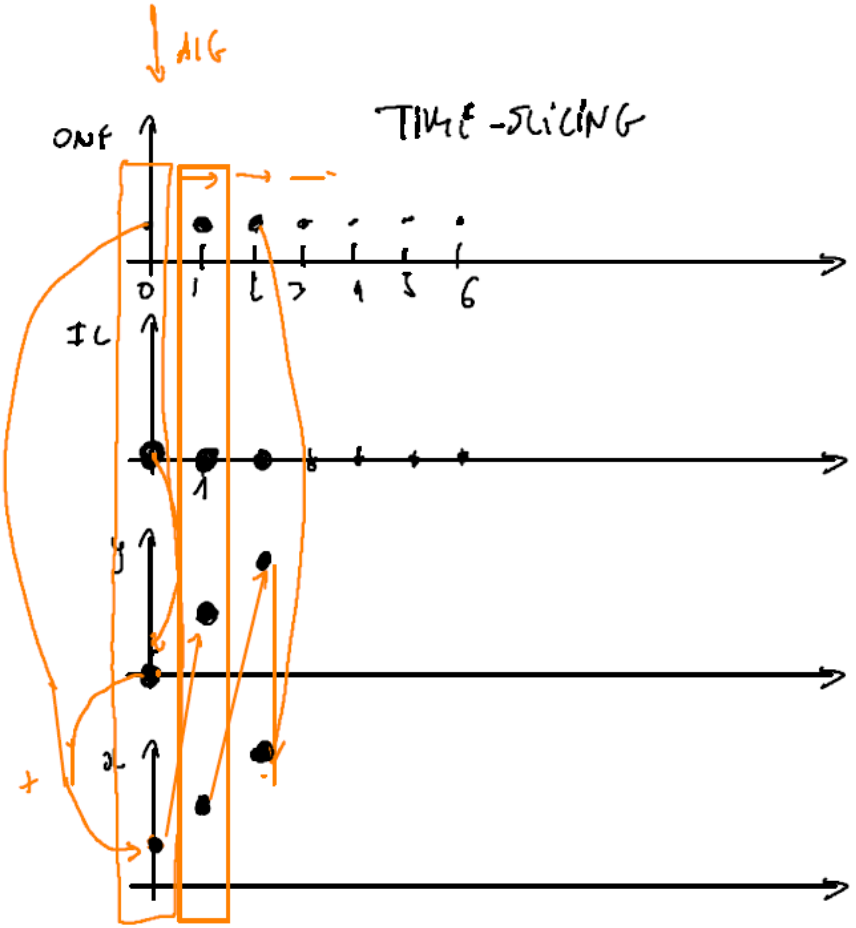


$:=$

$$\begin{cases} \nu_{IC}(i) = \phi, & \forall i \in \mathbb{N}_\phi \\ \nu_y(i) = \nu_x(i-1), & \forall i \in \mathbb{N} \\ \nu_y(\phi) = \nu_{ic}(\phi) \\ \nu_x(i) = \nu_{one}(i) + \nu_y(i), & \forall i \in \mathbb{N}_\phi \\ \nu_{one}(i) = 1, & \forall i \in \mathbb{N}_\phi \end{cases}$$

$= \quad ([0,0,0,\ldots],[0,1,1,\ldots],[1,2,3,\ldots],[1,1,1,\ldots]) \in \mathbb{R}$

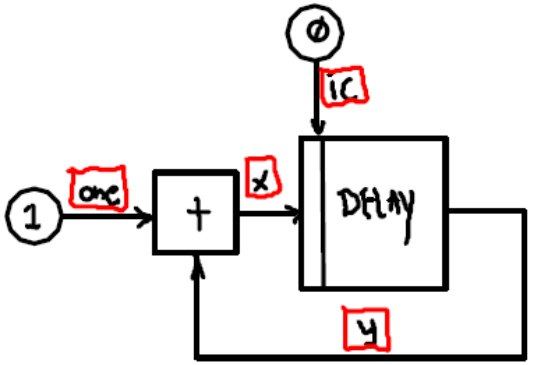
COUNTER

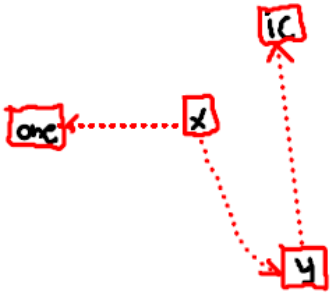


"behaviour trace"

operational semantics

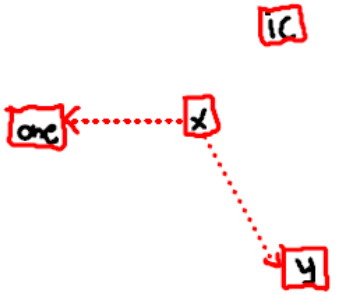
```
i = 0

while (not end_condition(i, ...)):

  depGraph = buildDepGraph(CBD)
  schedule = loopDetectAndTopSort(depGraph)

  for gblock in schedule:
    gblock.compute()

  i++
```





$i = 0$

schedule = [one, IC, y, x]



$\forall i \in \mathbb{N}$

schedule = [y, one, x, IC]

operational semantics

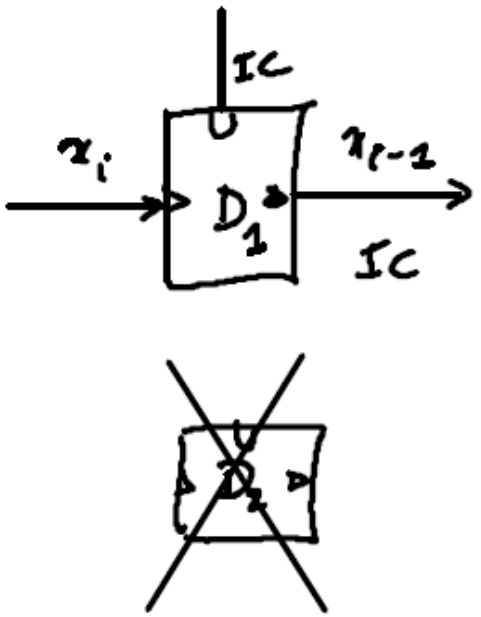
```
i = 0

while (not end_condition(i, ...)):

  depGraph = buildDepGraph(CBD)
  schedule = loopDetectAndTopSort(depGraph)

  for gblock in schedule:
    gblock.compute()

  i++
```

operational semantics

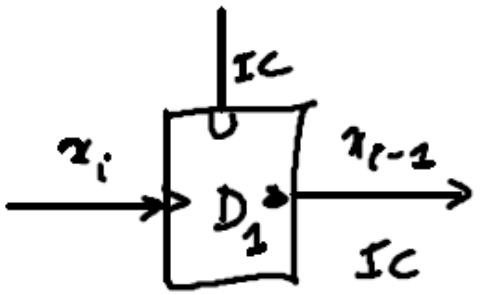
```
i = 0

while (not end_condition(i, ...)):

  depGraph = buildDepGraph(CBD)
  schedule = loopDetectAndTopSort(depGraph)

  for gblock in schedule:
    gblock.compute()

  i++
```

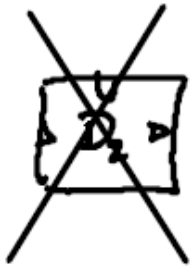
```
i = 0

if (not end_condition(i, ...)):
  depGraph = buildDepGraph(CBD)
  schedule = loopDetectAndTopSort(depGraph)

  for gblock in schedule:
    gblock.compute()
else:
  exit()

i = 1

while (not end_condition(i, ...)):

  depGraph = buildDepGraph(CBD)
  schedule = loopDetectAndTopSort(depGraph)

  for gblock in schedule:
    gblock.compute()

  i++
```

memory over more than 1 time-slice



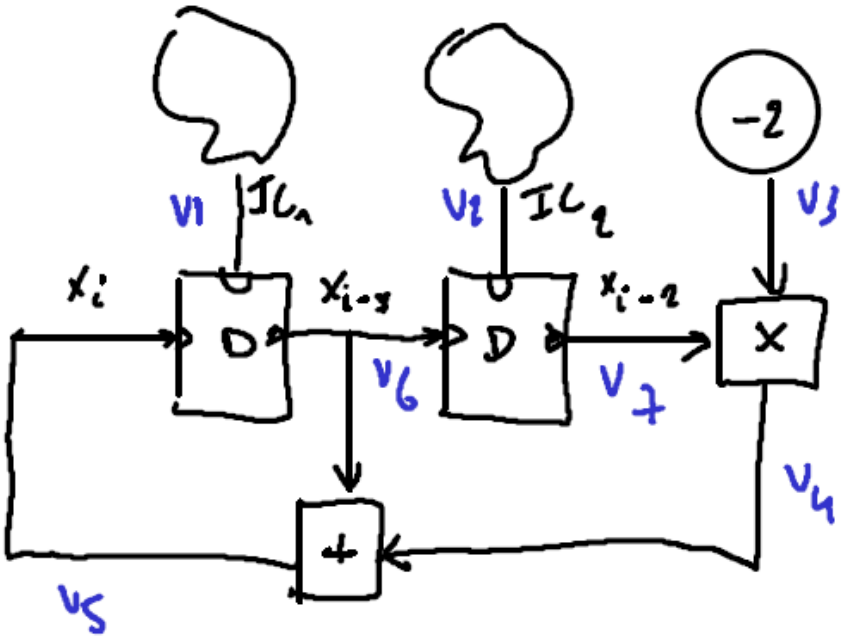
memory over more than 1 time-slice



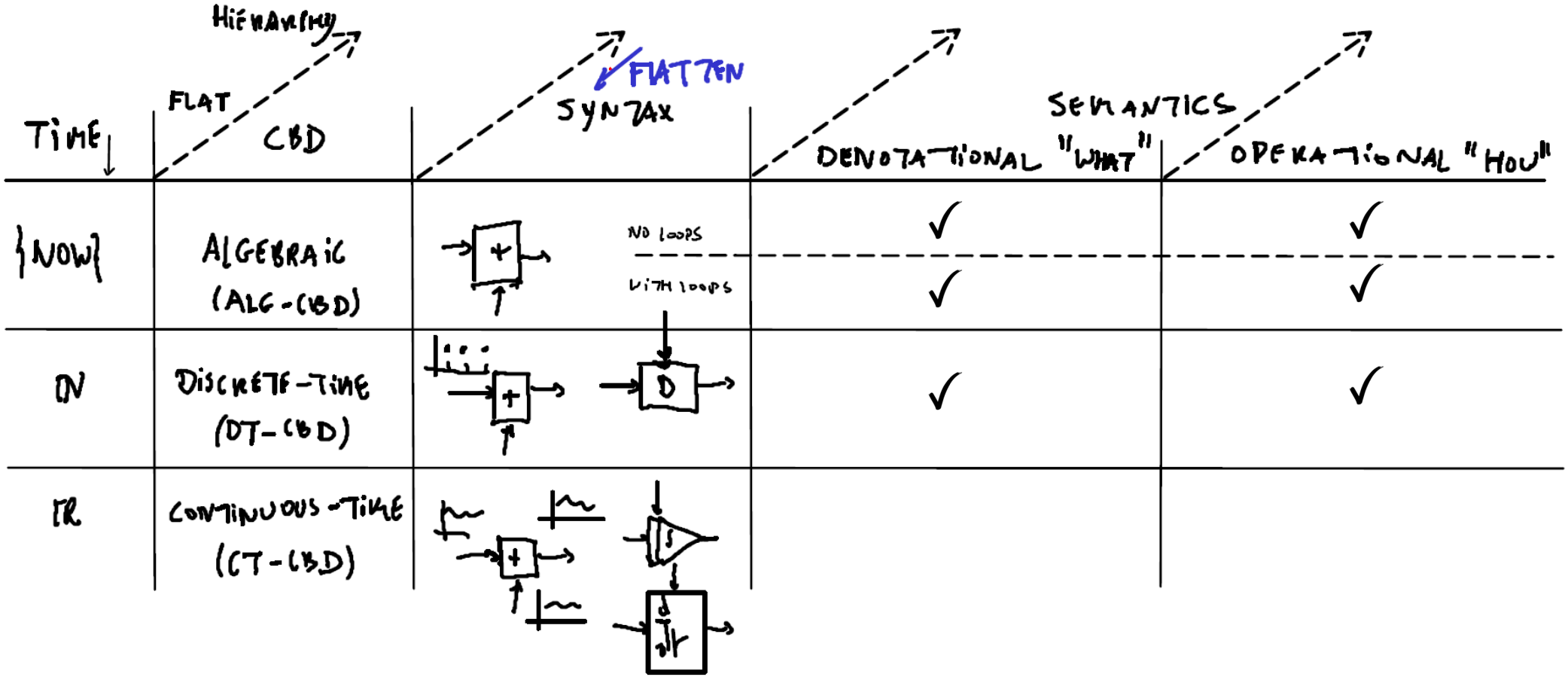


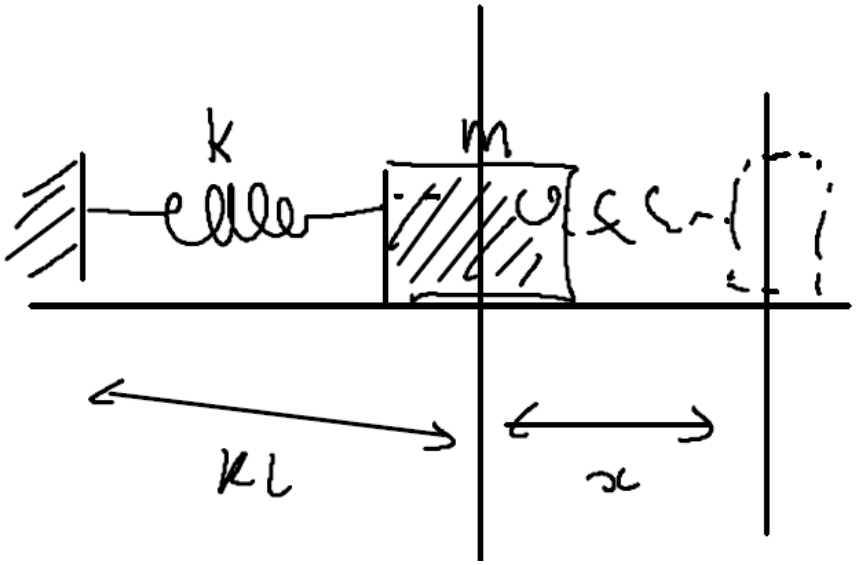
$$x_{i+1} = x_i - 2 \times x_{i-1}$$

$$x_i = x_{i-1} - 2 \times x_{i-2}$$



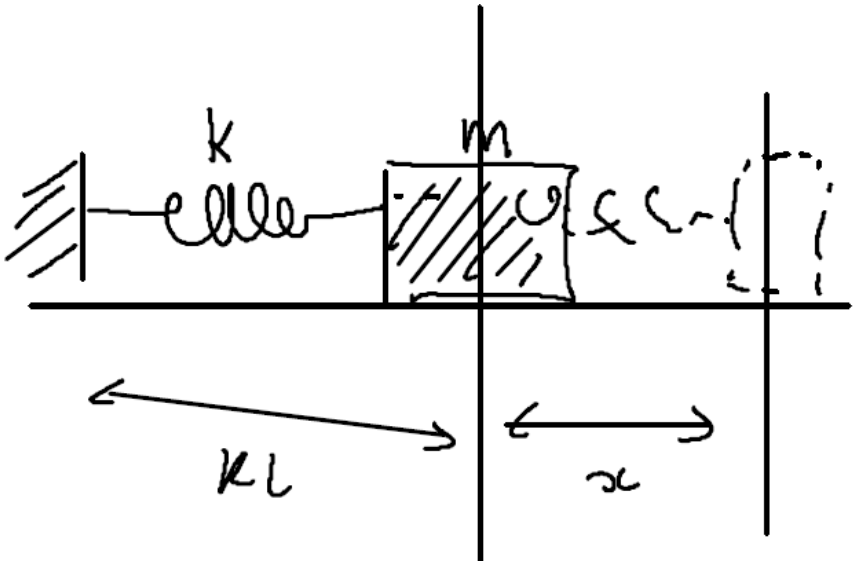
a family of Causal Block Diagram (CBD) formalisms





HOOKE'S LAW

$\underline{F} = -kx$ $\qquad x \ll$



HOOKE'S LAW

$F = -kx \qquad x \ll$

$F = ma$

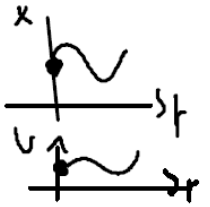
$$\begin{cases} F = m\dfrac{dv}{dt} = m\dfrac{d^2x}{dt^2} \\ v = \dfrac{dx}{dt} \end{cases}$$

ODE
ORDINARY
DIFFERENTIAL
EQNS

$$\begin{cases} \dfrac{dv}{dt} = -\dfrac{k}{m}x, \; v(0) \\ \dfrac{dx}{dt} = v, \; x(0) \end{cases}$$

PDE

$x(t)$, $v(t)$



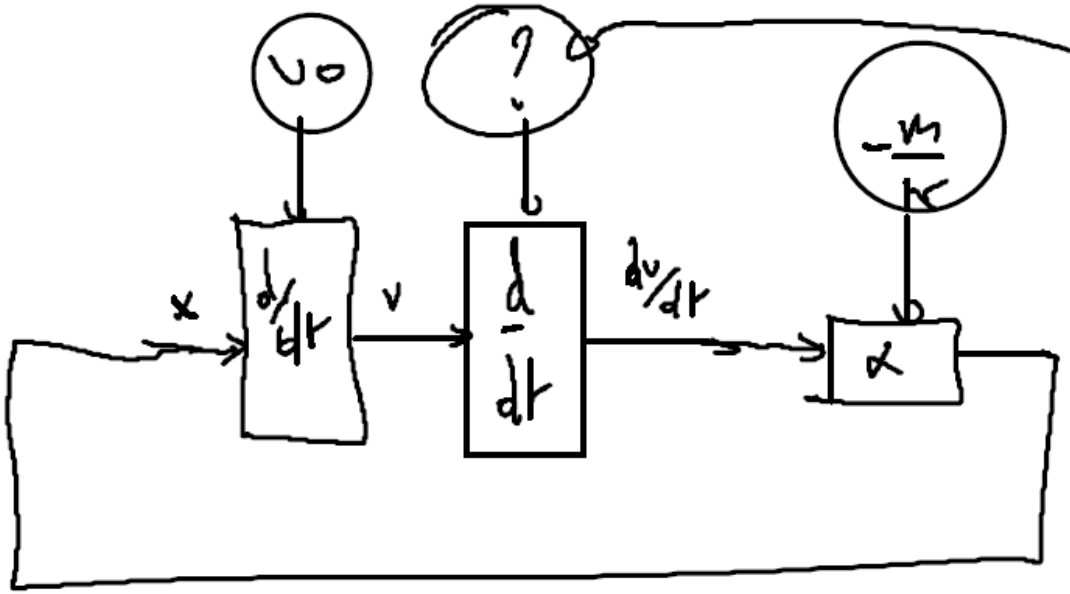
F = m **a** is <u>not</u> “rocket science”

$$v = \frac{d}{dt}x$$

$$a = \frac{d}{dt}v = \frac{d^2}{dt^2}x$$

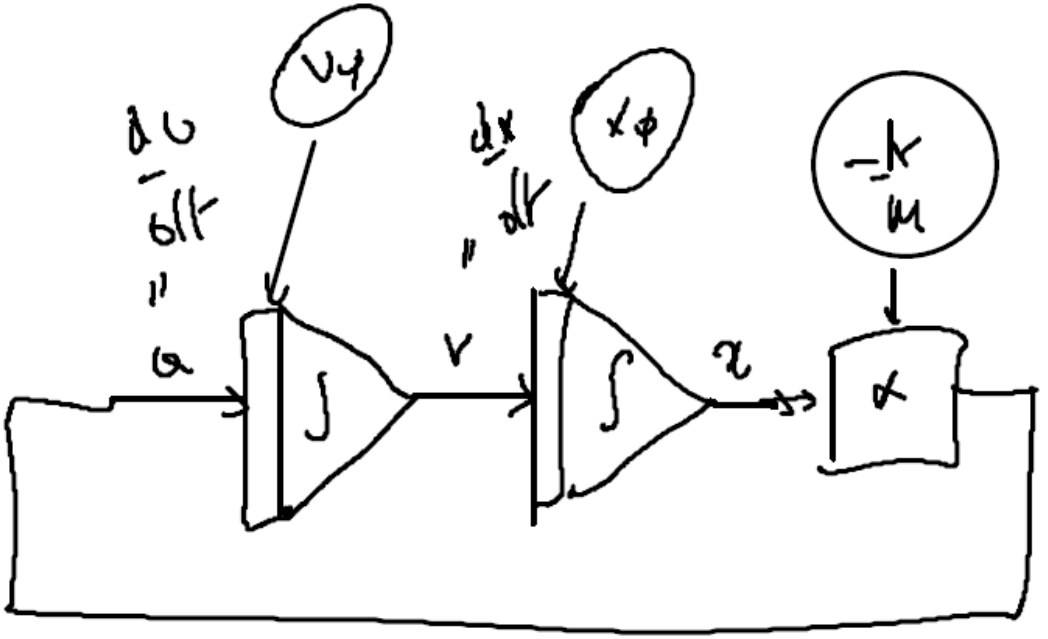
$$F = \frac{dP}{dt} = \frac{d}{dt}(mv)$$

$$= \underbrace{\frac{dm}{dt}\cdot v} + m\,\overbrace{\frac{dv}{dt}}^{a}$$



$$x = -\frac{m}{k}\frac{dv}{dt}$$

$$\frac{dv}{dt}(0) = -\frac{k}{m}x(0)$$

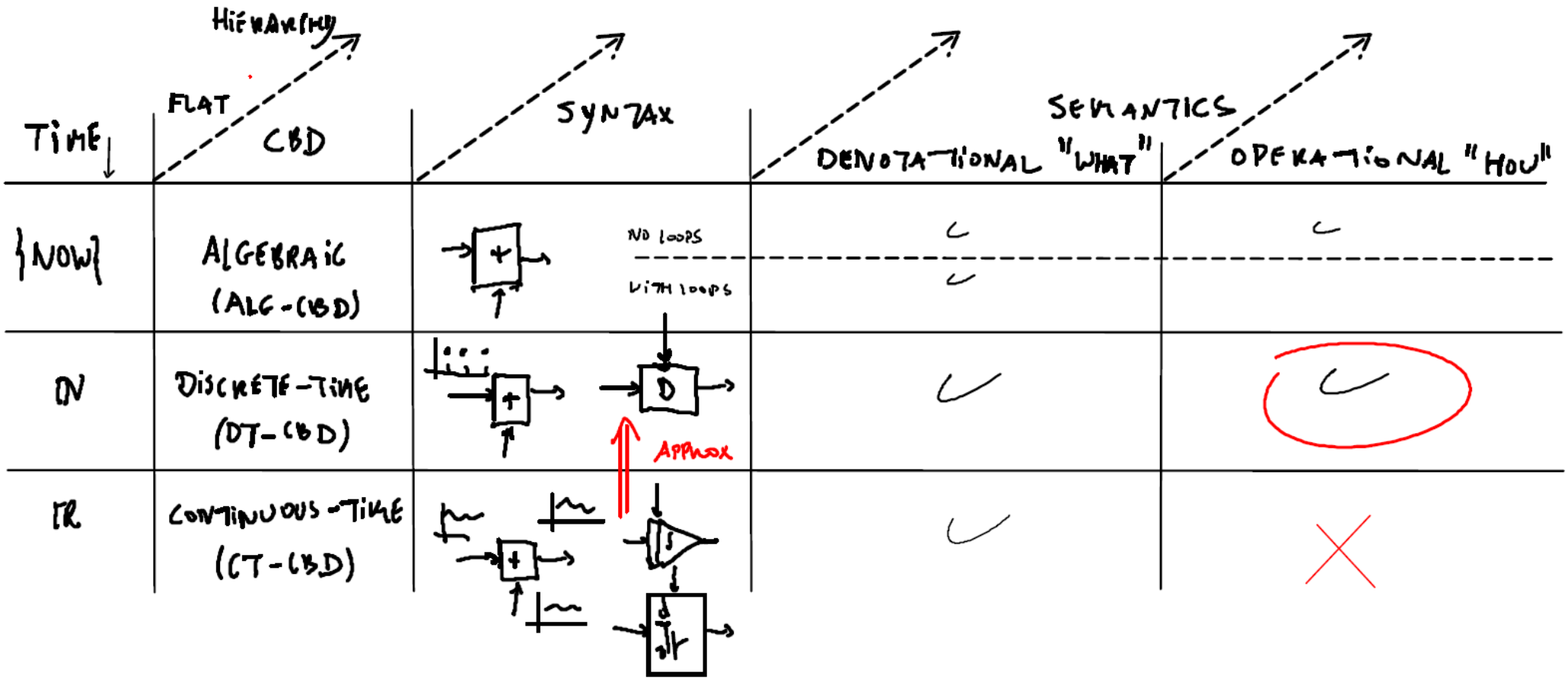


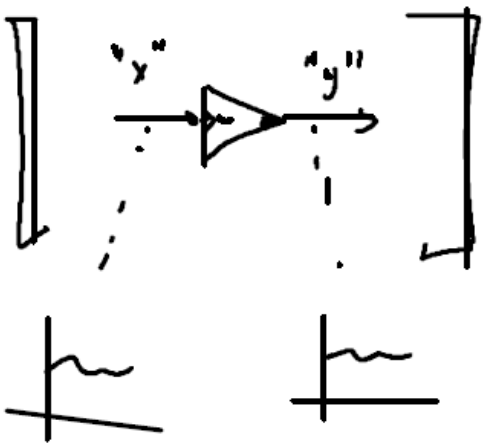
$$\lim_{\Delta t \to \phi}$$

Physical Systems Modelling

- Problem-Specific (technological)
- Domain-Specific (e.g., translational mechanical)
- (general) Laws of Physics
- Power Flow/Bond Graphs (physical: energy/power)
- Computationally a-causal
  (Mathematical and Object-Oriented) ← **Modelica**
- Causal Block Diagrams (data flow)
- Numerical (Discrete) Approximations
- Computer Algorithmic + Numerical
  (Floating Point vs. Fixed Point)
- As-Fast-As-Possible vs. Real-time (XiL)
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)
- Dynamic Structure

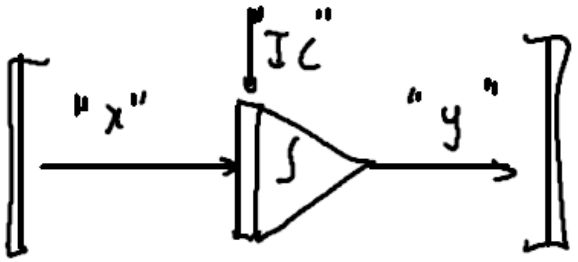
a family of Causal Block Diagram (CBD) formalisms



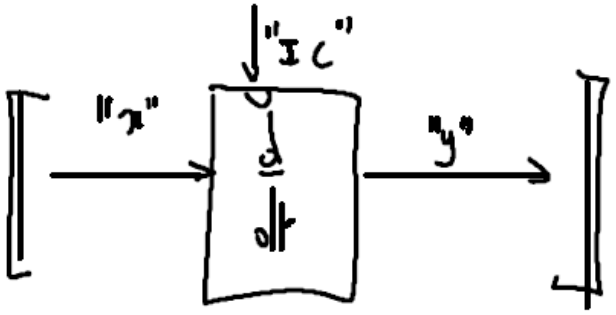


$:=$ $\quad \mathrm{val}_y(t) = -\mathrm{val}_x(t), \quad \forall t \in \mathbb{R}$

$-: \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

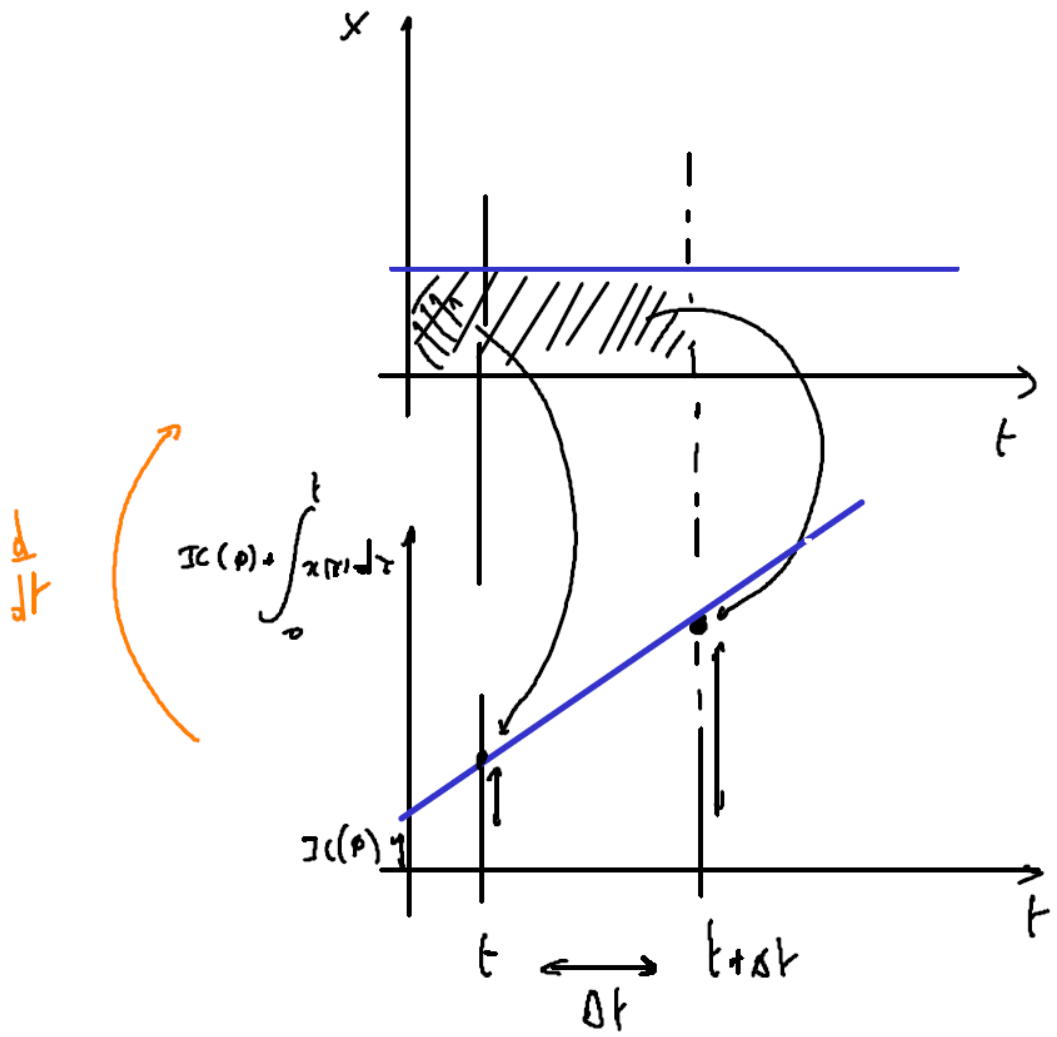


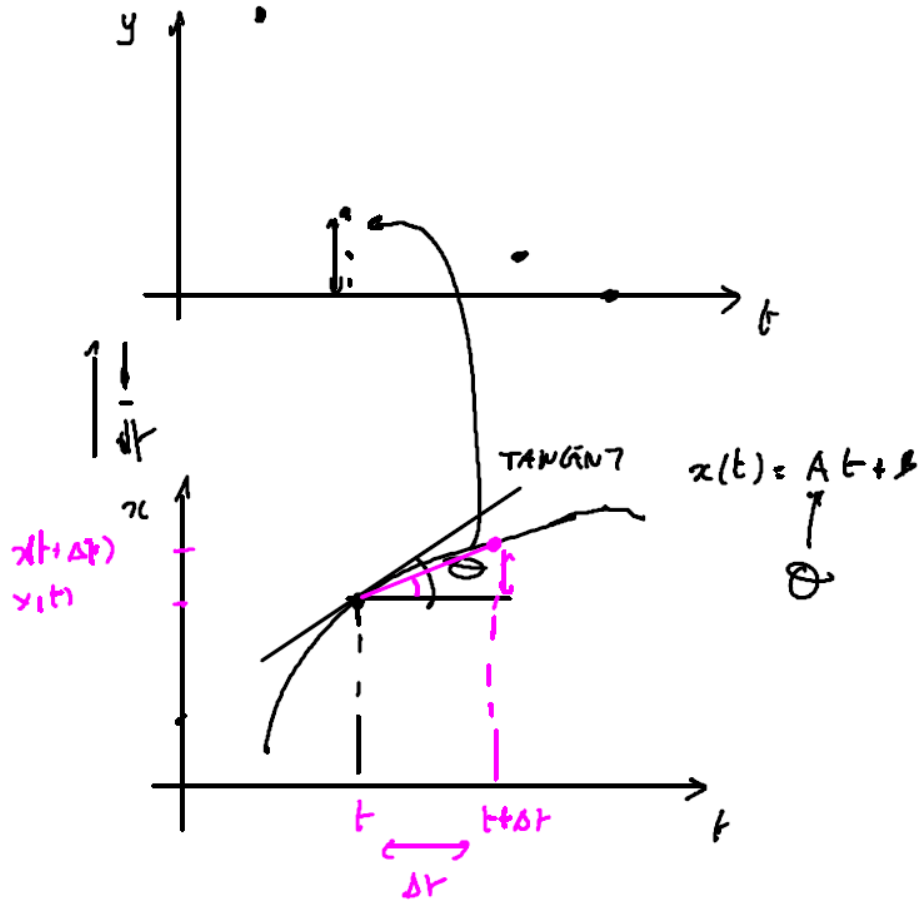
$:=$ $\quad \mathrm{val}_y(t) = \underbrace{\mathrm{val}_y(\phi)}_{IC(\phi)} + \int_\phi^t x(\tau)\, d\tau, \quad \forall t \in \mathbb{R}$



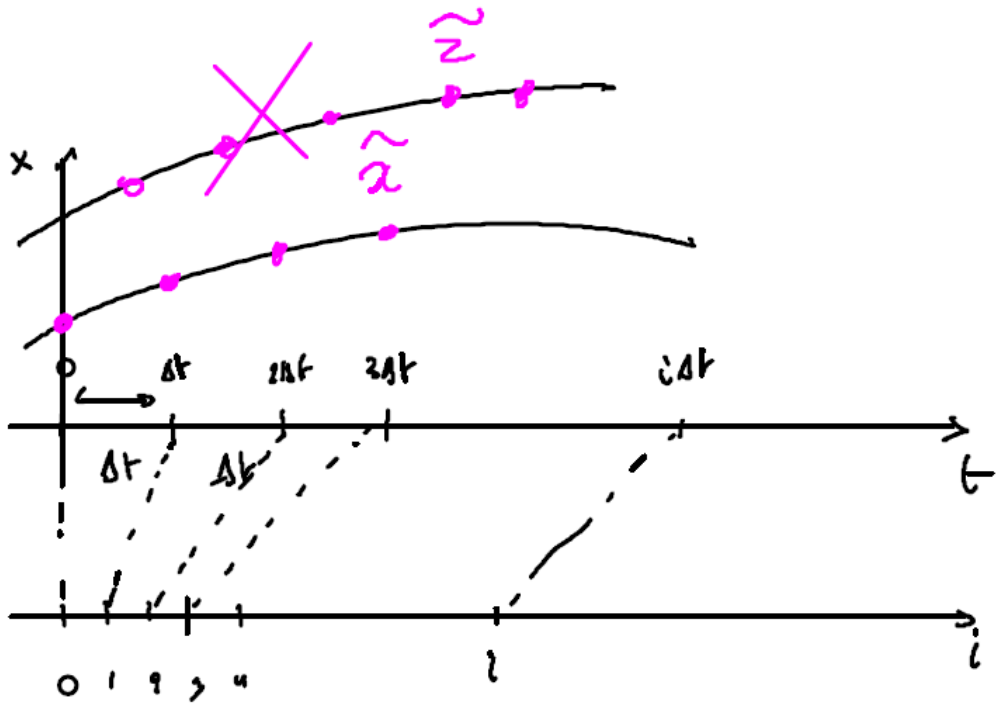
$:=$ $\quad \mathrm{val}_y(t) = \frac{d\ \mathrm{val}_x(t)}{dt}, \quad \forall t \in (\phi, \infty)$

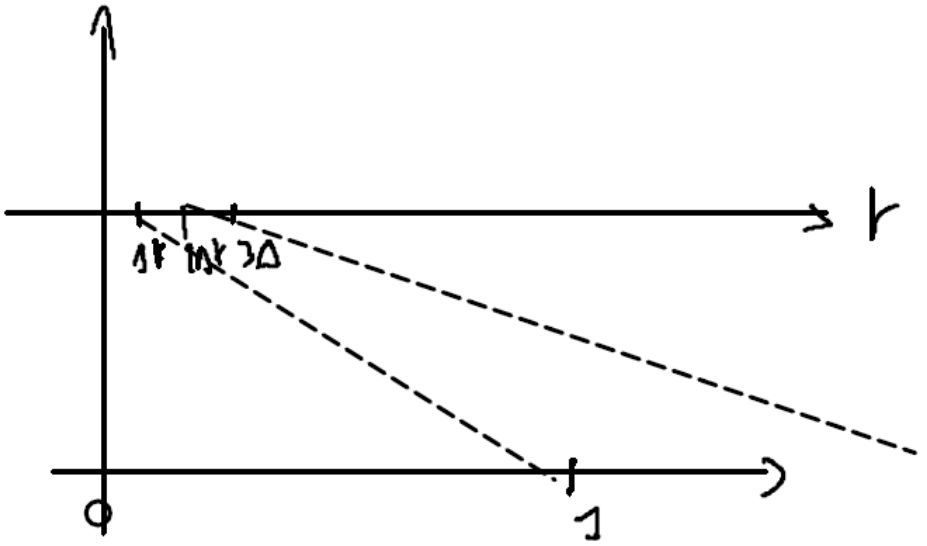
$\mathrm{val}_y(\phi) = IC(\phi)$

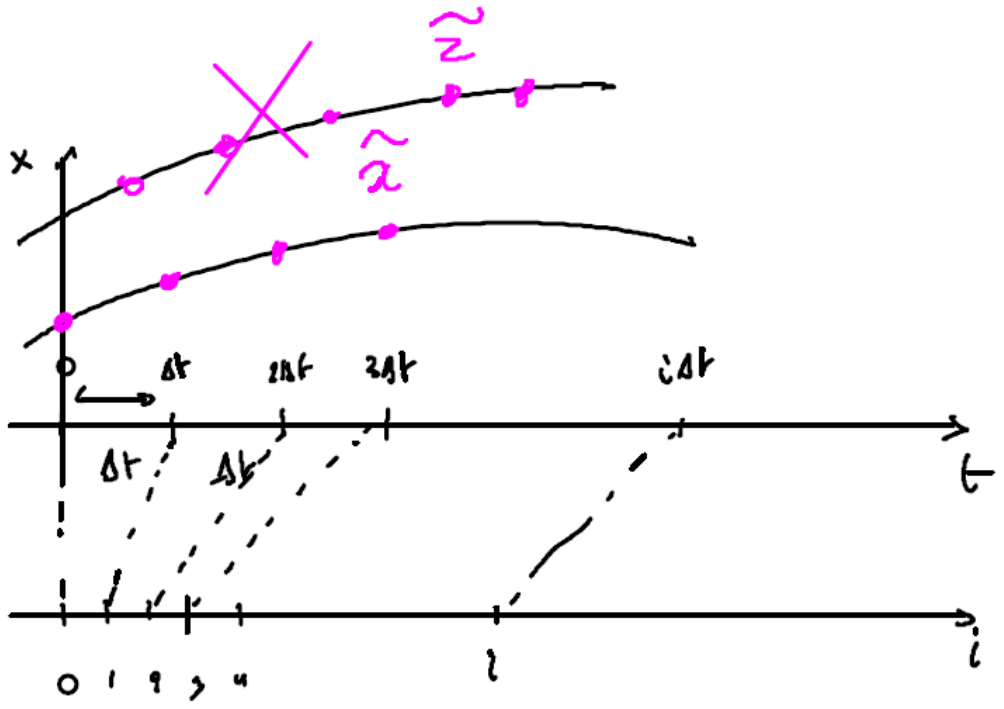


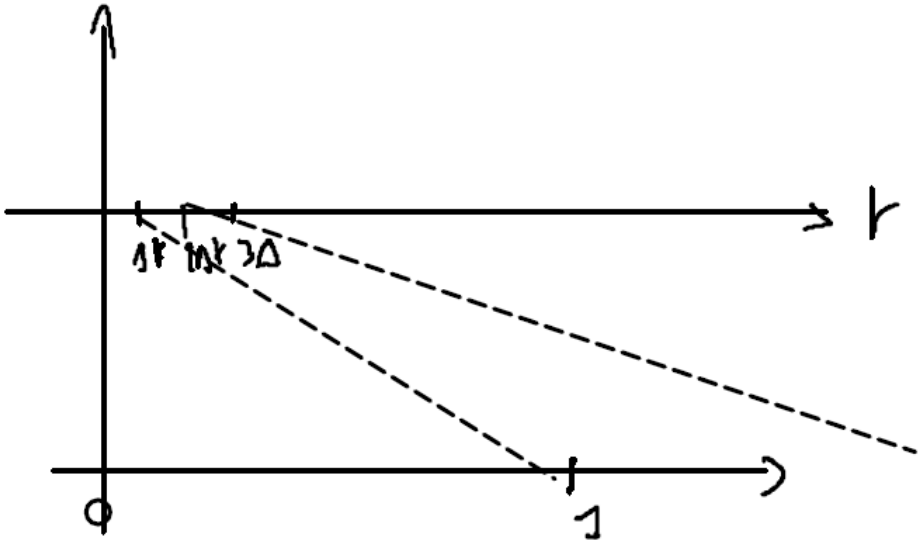


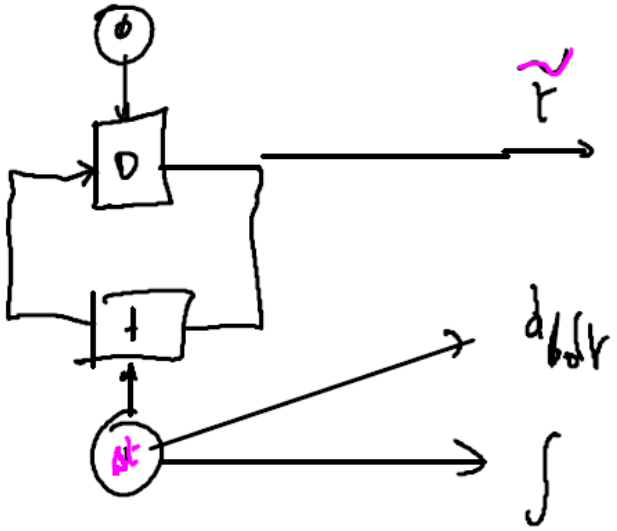
$$\lim_{\Delta t \to 0} \frac{x(t+\Delta t) - x(t)}{\Delta t}$$


x
Δt
2Δt
3Δt
iΔt
t
i
0
1
2
3
4
l


r
1
0


x
t
i
Δt
2Δt
3Δt
iΔt
0 1 2 3 4

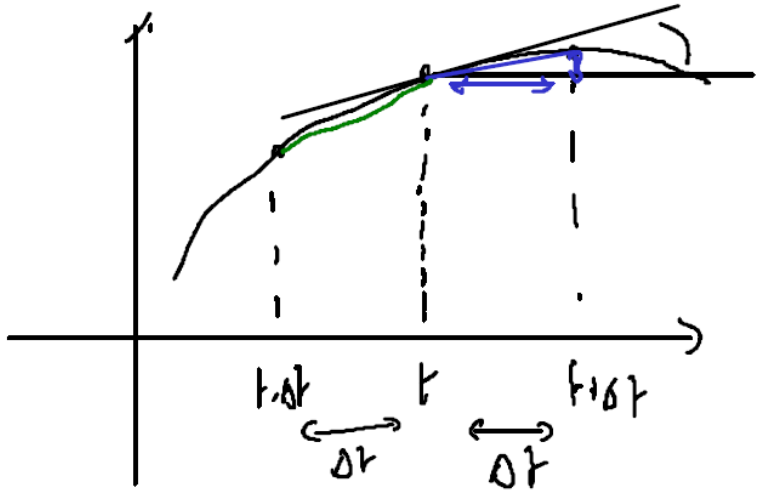

r
1
0


0
D
1
Δt

$$\tilde{y}(t) = \frac{\tilde{x}(t) - \tilde{x}(t-\Delta t)}{\Delta t}$$

$$\tilde{y}(t) = \frac{\tilde{x}(t+\Delta t) - \tilde{x}(t)}{\Delta t}$$

$\Delta t \ll$



$$\lim_{\Delta t \to 0}$$

$$\bar{y}(t) = \frac{d\bar{x}}{dt}(t) = \frac{\bar{x}(t+\Delta t) - \bar{x}(t)}{\Delta t}$$

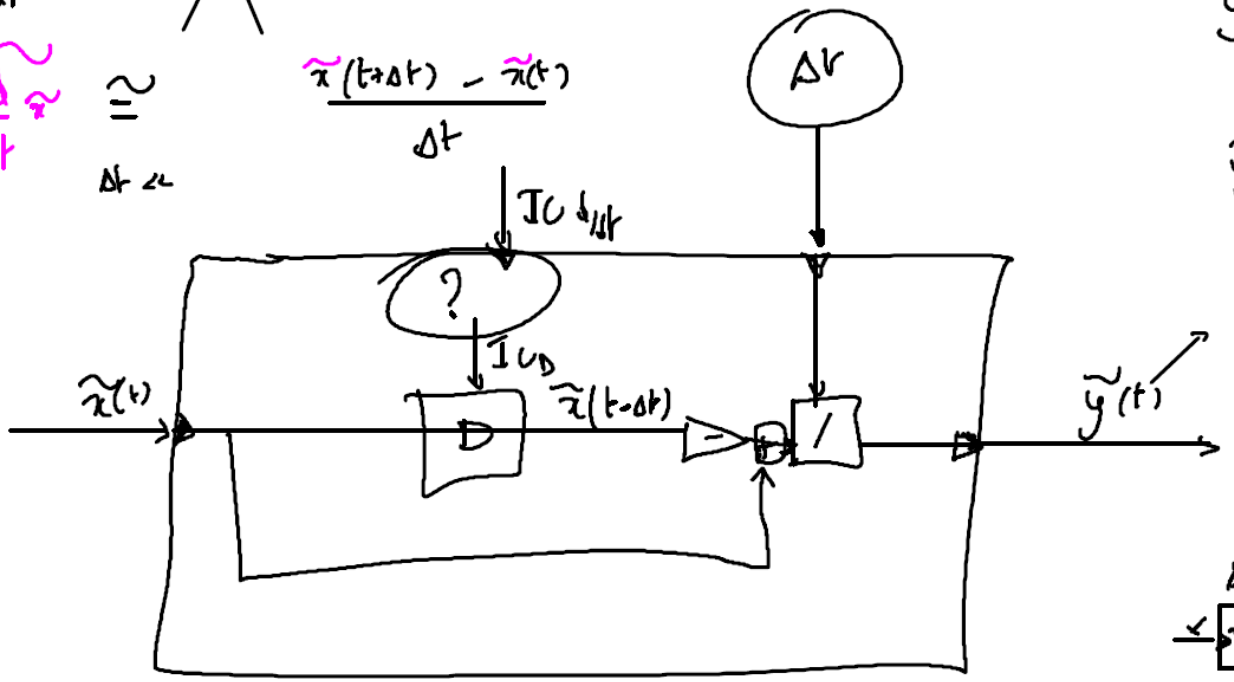
$$\hat{y}(t) = \frac{d\hat{x}}{dt}(t) = \frac{\hat{x}(t) - \hat{x}(t-\Delta t)}{\Delta t}$$

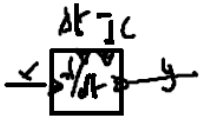
$\frac{d}{dt}x(t) = \lim_{\Delta t \to 0} \frac{x(t+\Delta t) - x(t)}{\Delta t}$

$\tilde{y}(t) \simeq \frac{d}{dt}\tilde{x} \simeq \frac{\tilde{x}(t+\Delta t) - \tilde{x}(t)}{\Delta t}$ $\Delta t \ll$

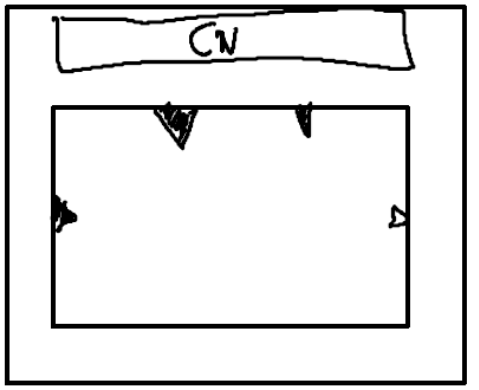
$\tilde{y}(t+\Delta t) = \frac{x(t+\Delta t) - x(t)}{\Delta t}$

$\tilde{y}(t) = \frac{\tilde{x}(t) - \tilde{x}(t-\Delta t)}{\Delta t}$



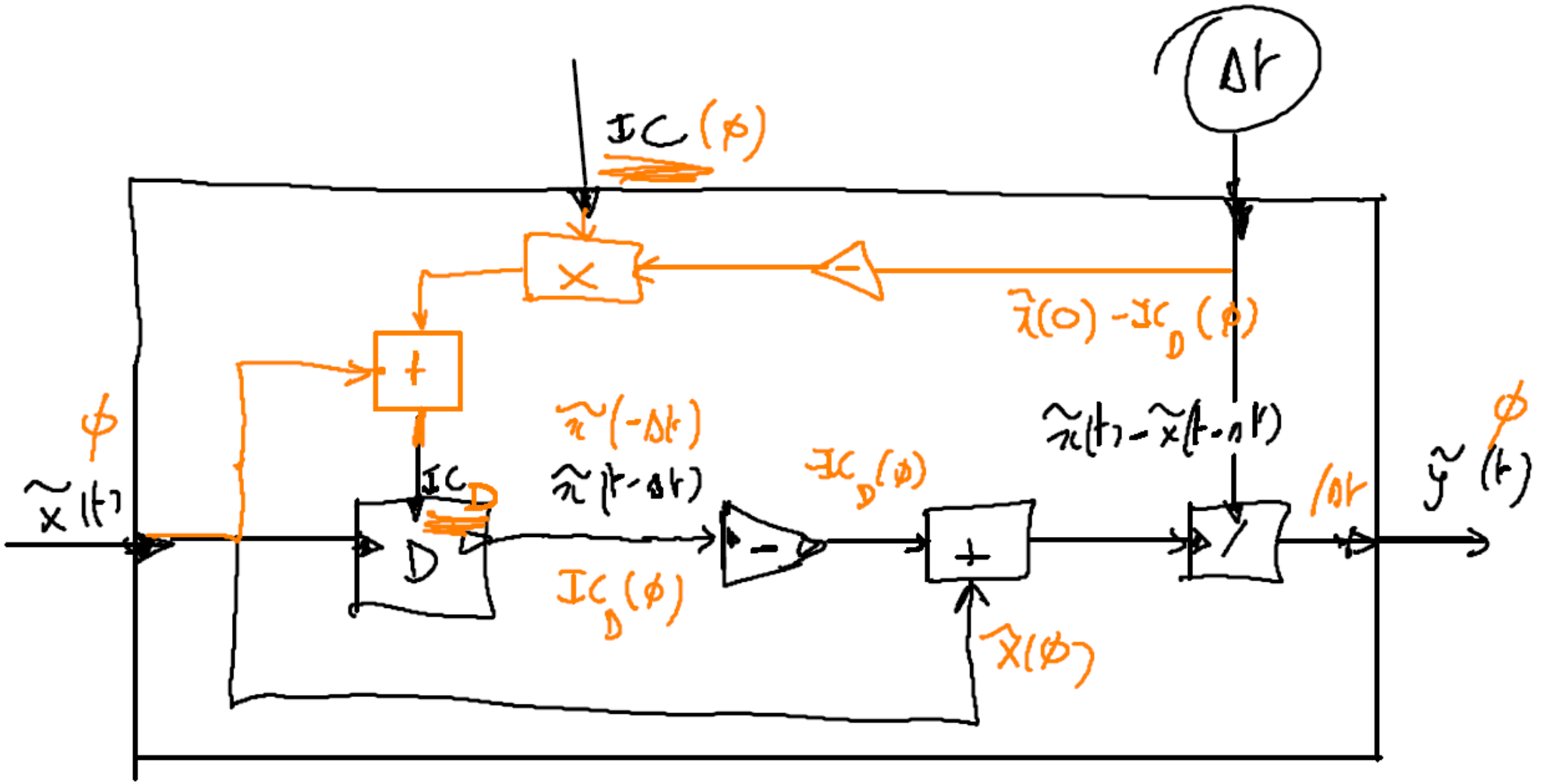


$y = \frac{d}{dt}x \qquad y(0) = IC(0)$



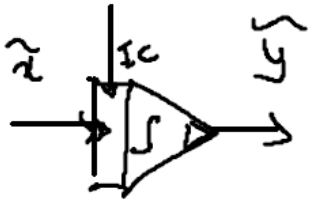
ALL BLOCKS : = Δt

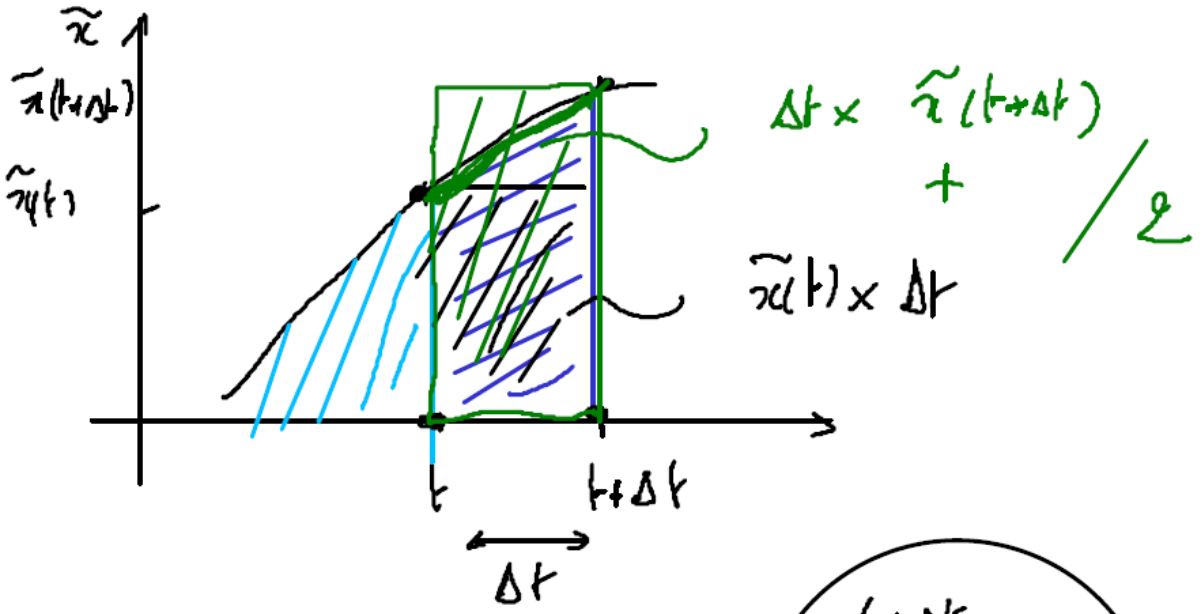
SYNCHRONOUS BLOCK DIAGRAMS



$$\frac{\tilde{x}(0) - IC_D(0)}{\Delta t} = \underset{\Delta t \to 0}{IC_d(0)} \qquad \tilde{y}(\phi) = \underset{\Delta t \to 0}{IC_d(0)}$$

$$\tilde{x}(0) - \underset{\Delta t \to 0}{IC_f(0)} \cdot \Delta t = IC_D(0)$$



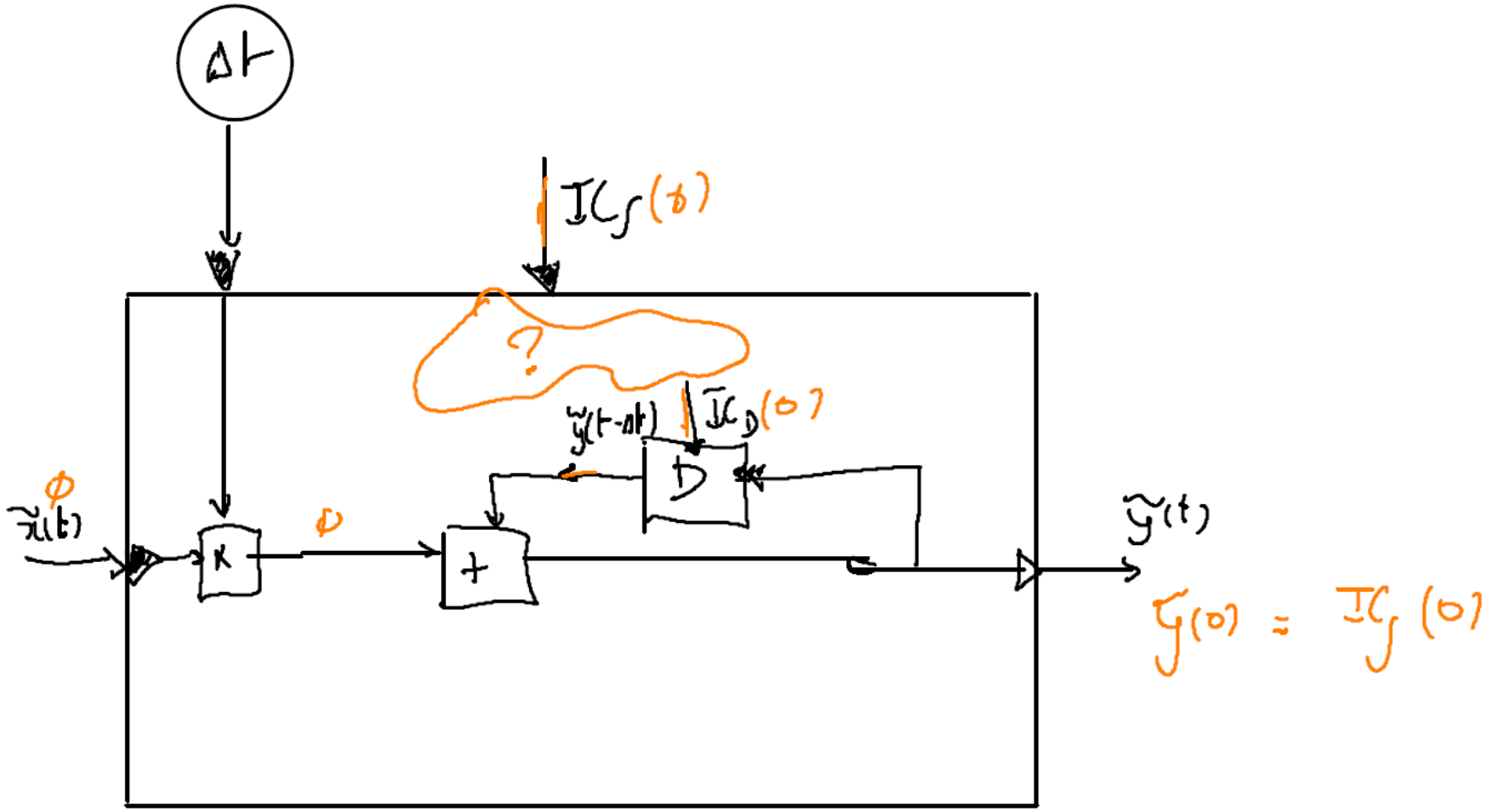


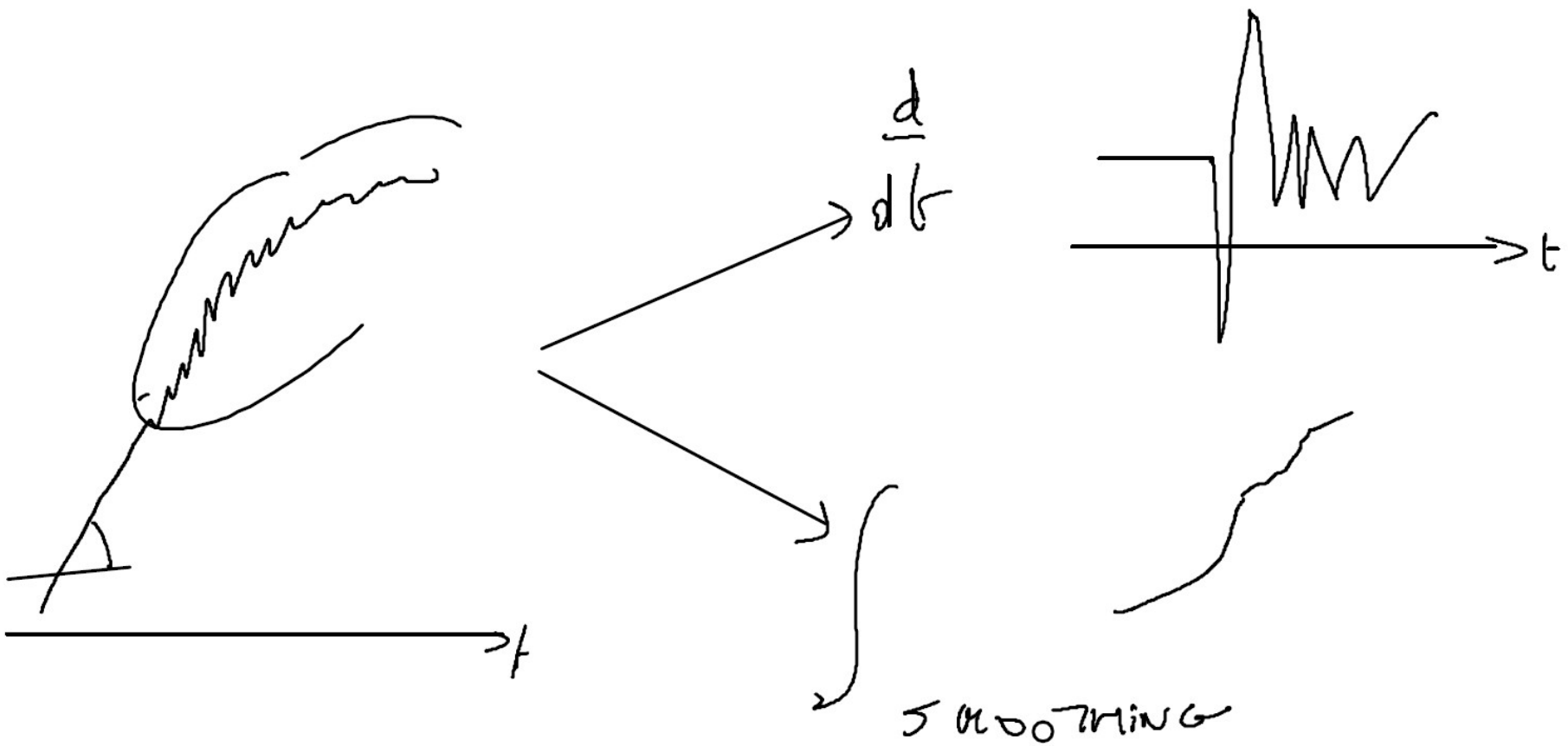
$$\tilde{y}(t+\Delta t) = \tilde{y}(t) + \int_{t}^{t+\Delta t} \tilde{x}(\tau)\, d\tau$$

$$\lim_{\Delta t \to 0} \Delta t\ \tilde{x}(t) \qquad \Delta t\ \tilde{x}(t+\Delta t)$$

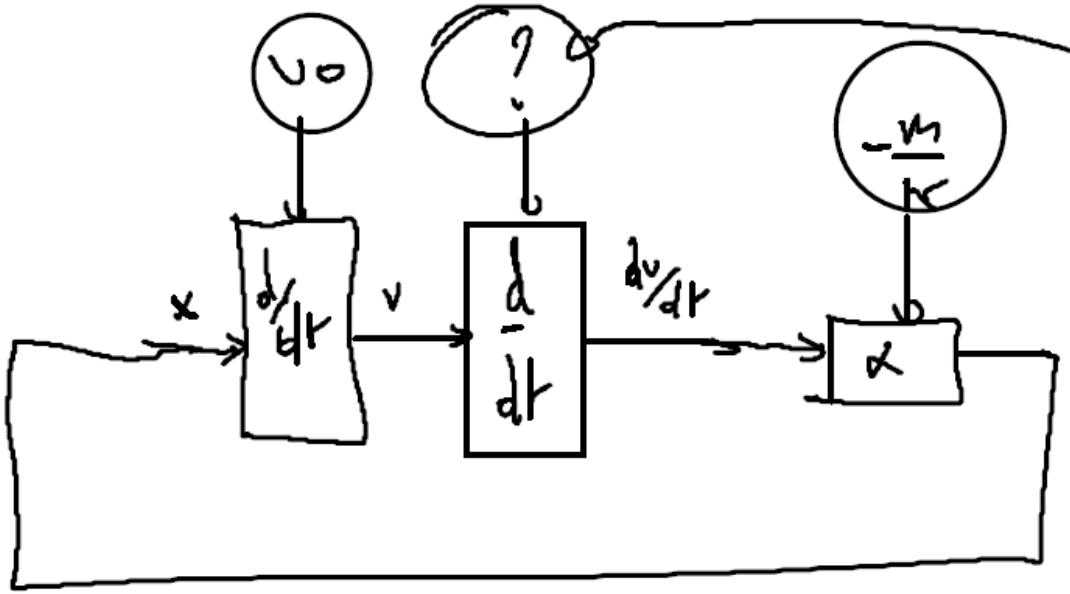
$$\tilde{y}(t) \approx y(t-\Delta t) + \Delta t\ \tilde{x}(t)$$

Approx


Δt
IC_f(t)
?
ỹ(t-Δt)
IC_D(0)
D
φ
x̃(t)
K
D
+
ỹ(t)
y(0) = IC_f(0)



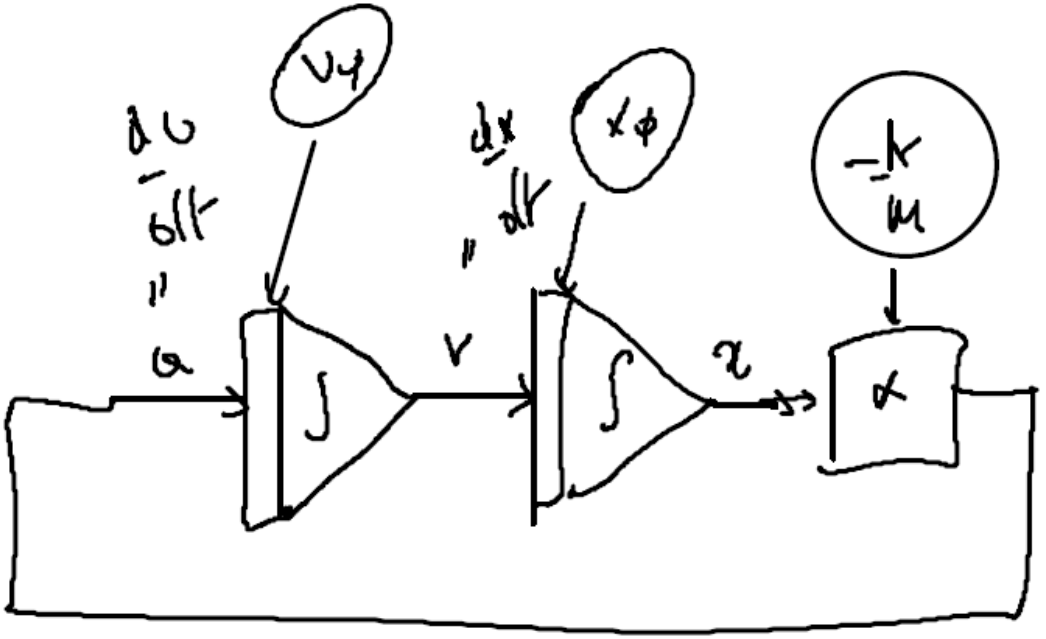
→ If choice: prefer intergration over differentiations



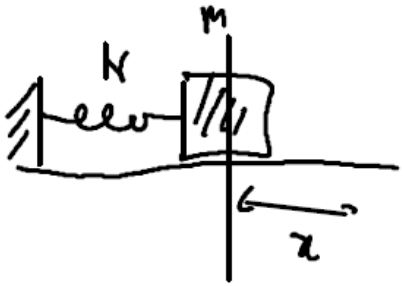
$$x = -\frac{m}{k}\frac{dv}{dt}$$

$$\frac{dv}{dt}(0) = -\frac{k}{m}x(0)$$

→ If choice: prefer integration over differentiation

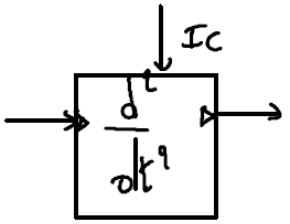


$$\lim_{\Delta t \to 0}$$

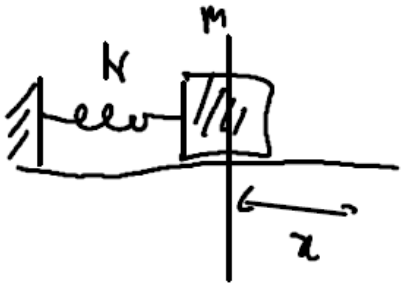


HARMONIC EQN.

$$\frac{d^2x}{dt^2} = -x, \qquad x(0) = x_0, \quad \frac{dx}{dt}(0) = v(0) = v_0 \qquad \text{ODE}$$

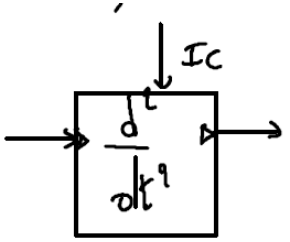


$$\begin{cases} \dfrac{dx}{dt} = v, & x(0) = x_0 \\ \dfrac{dv}{dt} = -x, & v(0) = v_0 \end{cases}$$



HARMONIC EQN.

$$\frac{d^2x}{dt^2} = -x, \qquad x(0) = x_0, \quad \frac{dx}{dt}(0) = v(0) = v_0 \qquad \text{ODE}$$



$$\begin{cases} \dfrac{dx}{dt} = v, & x(0) = x_0 \\ \dfrac{dv}{dt} = -x, & v(0) = v_0 \end{cases}$$

HIGHER-ORDER ODES

$$\frac{d^k x}{dt^k} = \underbrace{\frac{d}{dt}\left(\frac{d}{dt}\left(\cdots \frac{d}{dt}x\right)\right)}_{k}$$

$$\begin{cases} \dfrac{dx}{dt} = x\_der\_1 \\ \dfrac{dx\_der\_1}{dt} = x\_der\_2 \\ \vdots \\ \dfrac{dx\_der\_k-1}{dt} = x\_der\_k \end{cases}$$

$$\frac{d^2x}{dt^2} = -x \quad , \quad x(0) = \phi \quad , \quad \frac{dx}{dt}(0) = 1$$

$$\frac{dx}{dt} = x$$

$$x(t) = e^x + C$$

$$\begin{cases} x(t) = A \sin(t) + B \cos(t) \left(+ C\right) \\ x(0) = \phi \quad , \quad \frac{dx}{dt}(0) = 1 \end{cases}$$

$$\frac{dx}{dt} = A\cos(t) - B\sin(t)$$

$$\frac{d^2x}{dt^2} = -A\sin(t) - B\cos(t) = -(A\sin(t) + B\cos(t)) = -x(t) \qquad \text{q.e.d.}$$

$$x(0) : \quad B = \phi$$

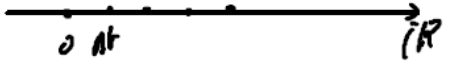
$$\frac{dx}{dt}(0) : \quad A = 1$$

$$\begin{cases} x(t) = \sin(t) \\ \frac{dx}{dt}(t) = \cos(t) \end{cases}$$

$\tilde{x}(\tilde{t})$

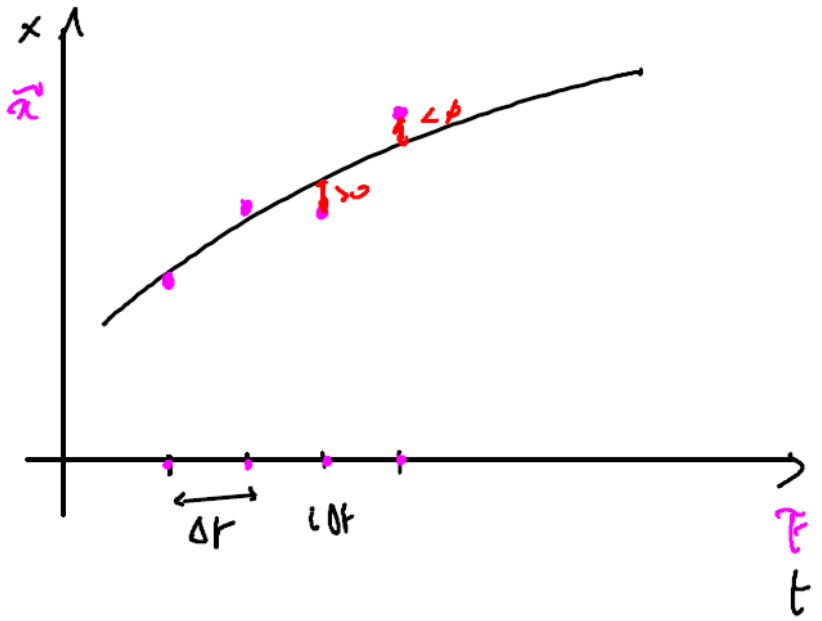
$\frac{d\tilde{x}}{d\tilde{t}} = \tilde{v}(\tilde{t})$

$\tilde{t} = i \times \Delta t$



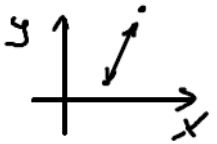
Error $\sim$ $e(i\Delta t) = x(i\Delta t) - \tilde{x}(i\Delta t)$

Total error $\sim \sum_{i=0}^{i\_end} e(i\Delta t)$



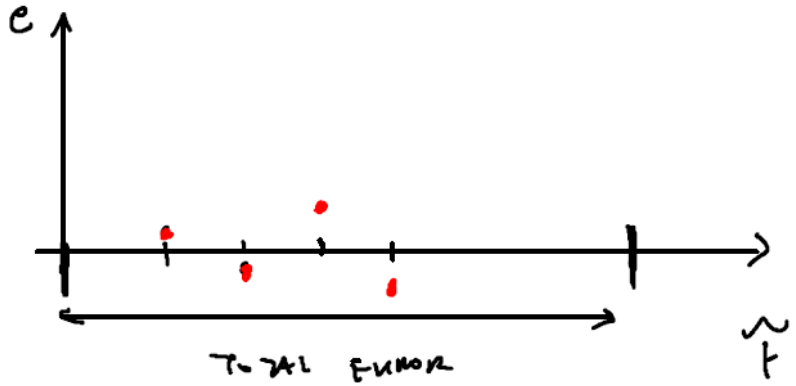
$e(i\Delta t) = |x(i\Delta t) - \tilde{x}(i\Delta t)|$

$e(i\Delta t) = (x(i\Delta t) - \tilde{x}(i\Delta t))^2$



"SUM OF SQUARES"

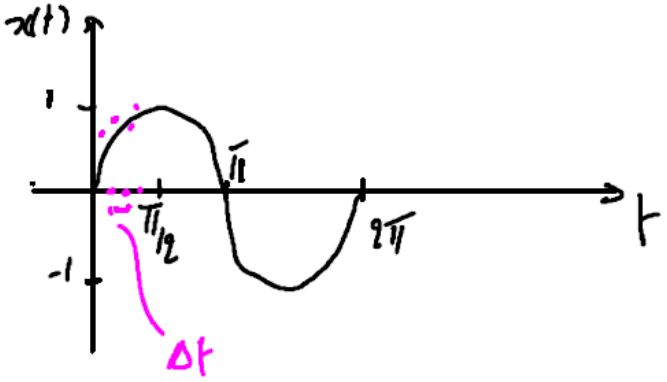
Total error $= \sum_{i=\phi}^{i\_end} (x(i\Delta t) - \tilde{x}(i\Delta t))^2$

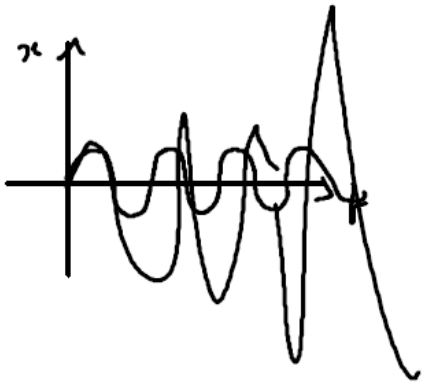


$= \int (x(\tau) - \tilde{x}(\tau))^2 \, d\tau$

$x(\tau)$ (BD)

$\tilde{x}(\tau)$ (BD)



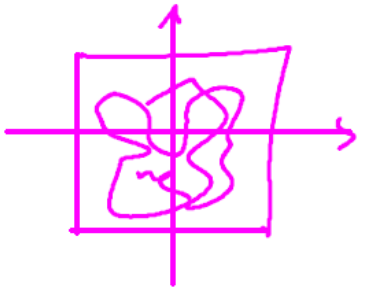


UNSTABLE

SYSTEM ?

NUMERICAL ?

BOUNDED REGION



PHASE PLOT ~ STABILITY

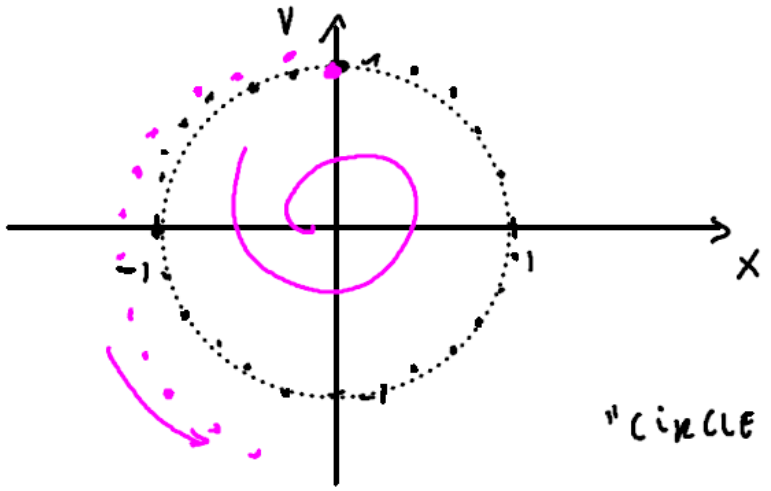
- SYSTEM (SOS)
- NUMERICAL

~ Δt, SCHEME

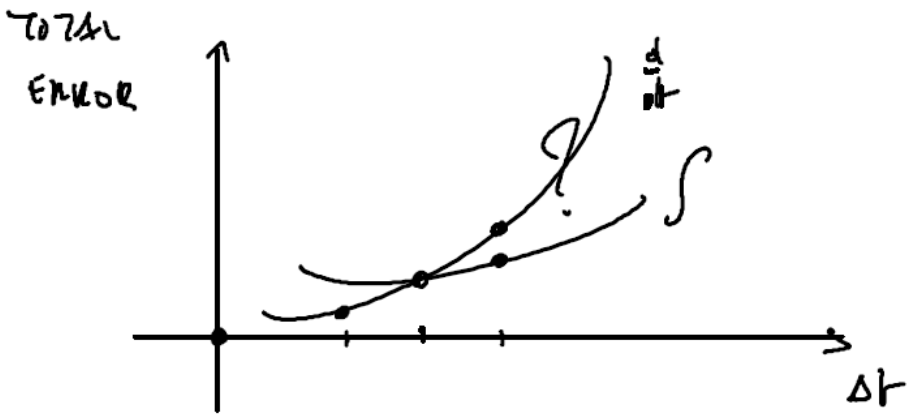
$(x(t), v(t))$

$x(0) = 0$, $v(0) = 1$

$x(\frac{\pi}{2}) =$, $\sim (\frac{\pi}{2}) =$



"CIRCLE TEST"



DISCRETIZATION SCHEME FIXED

$$\text{Total Error}\ (\Delta t) \overset{?}{=} \mathcal{O}(\Delta t)$$

$$x(t+\Delta t) = x(t) + \frac{x'(t)\Delta t}{1!} + \frac{x''(t)}{2!}\Delta t^2 + \frac{x^{(3)}(t)}{3!}\Delta t^3 + \dots$$

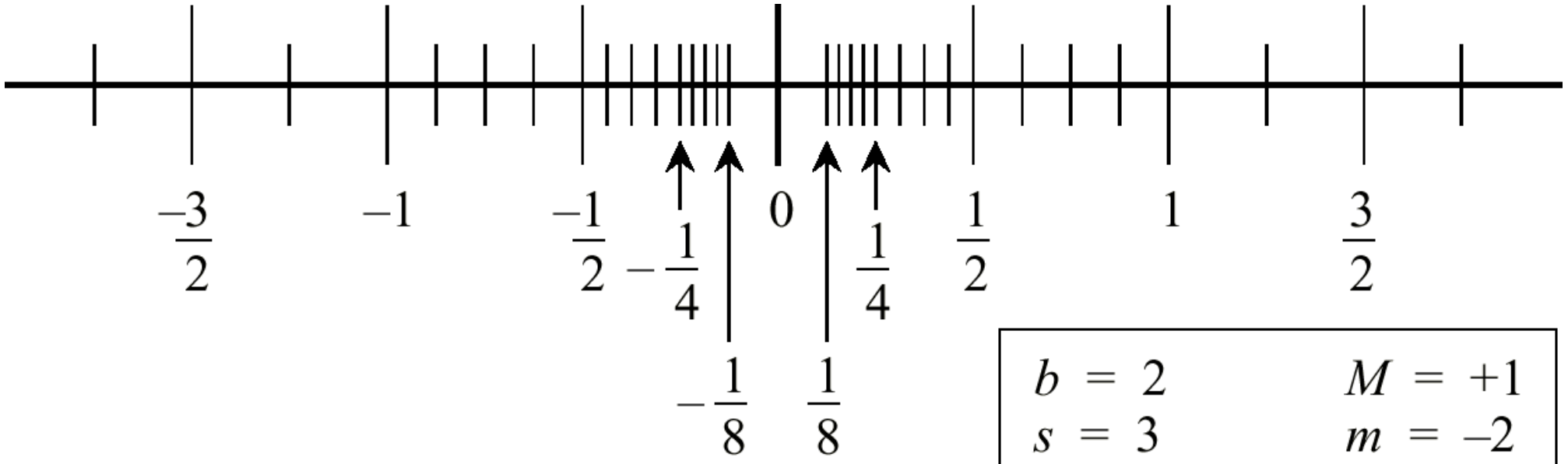
TAYLOR EXPANSION

$$\longrightarrow \mathcal{O}(\Delta t^2)$$

Physical Systems Modelling

- Problem-Specific (technological)
- Domain-Specific (e.g., translational mechanical)
- (general) Laws of Physics
- Power Flow/Bond Graphs (physical: energy/power)
- Computationally a-causal
  (Mathematical and Object-Oriented) ← **Modelica**
- Causal Block Diagrams (data flow)
- Numerical (Discrete) Approximations
- Computer Algorithmic + Numerical
  (Floating Point vs. Fixed Point)
- As-Fast-As-Possible vs. Real-time (XiL)
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)
- Dynamic Structure

Example Floating Point Format



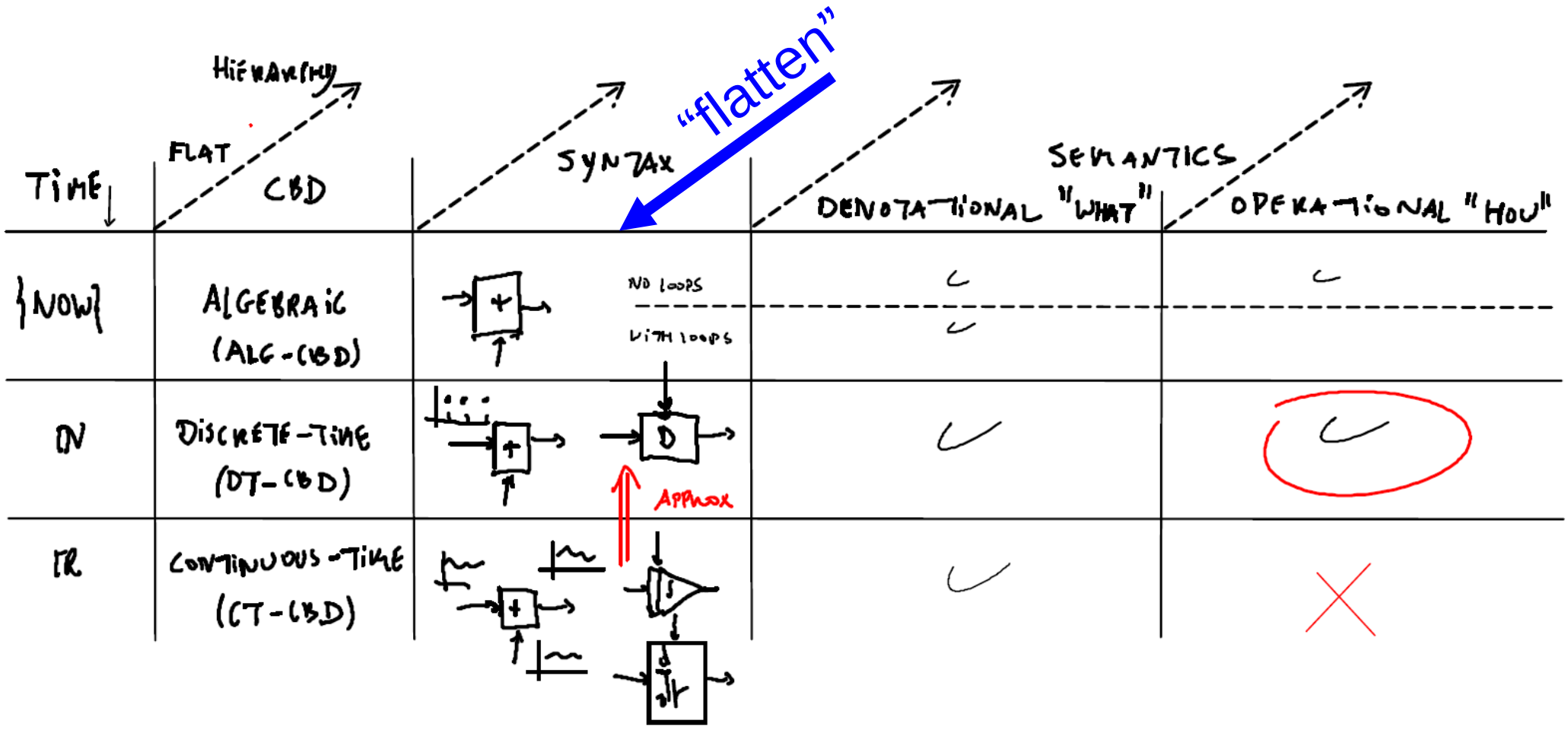
- Smallest non-zero positive number = $b^m \times b^{-1} = 1/8$
- Largest non-zero positive number = $b^M \times (1 - b^{-s}) = 7/4$
- Smallest gap = $b^m \times b^{-s} = 1/32$
- Largest gap = $b^M \times b^{-s} = 1/4$
- Number of representable numbers = $2\times((M-m)+1)\times(b-1)\times b^{s-1}+1 = 33$
  ... fits into available bits? Optimal number of bits?
- Note: fill the gap around 0: **de-normalized**

Physical Systems Modelling

- Problem-Specific (technological)
- Domain-Specific (e.g., translational mechanical)
- (general) Laws of Physics
- Power Flow/Bond Graphs (physical: energy/power)
- Computationally a-causal
  (Mathematical and Object-Oriented) ← **Modelica**
- Causal Block Diagrams (data flow)
- Numerical (Discrete) Approximations
- Computer Algorithmic + Numerical
  (Floating Point vs. Fixed Point)
- As-Fast-As-Possible vs. Real-time (XiL)
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)
- Dynamic Structure

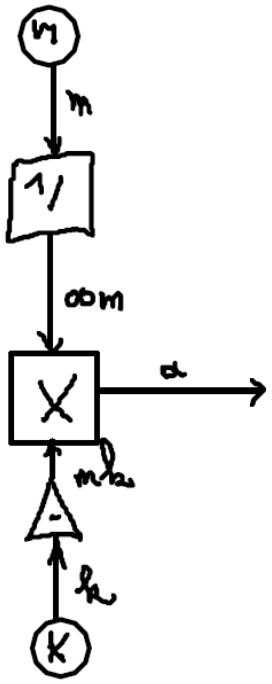
++ hierarchy

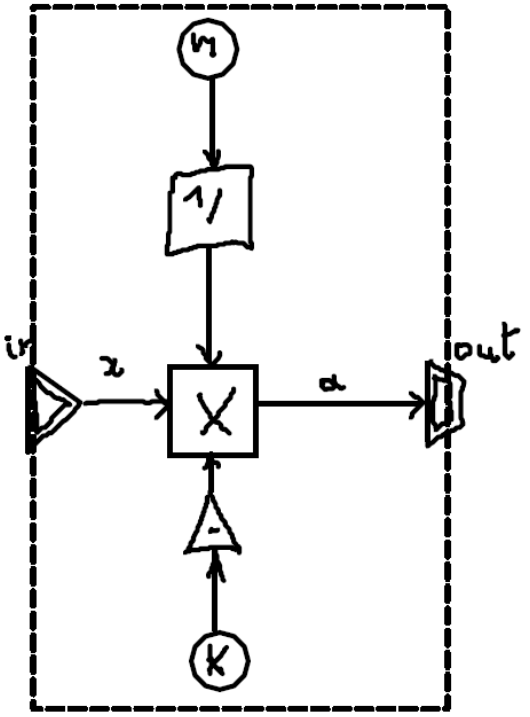
adding hierarchy to CBD formalisms



adding hierarchy to CBD formalisms

modularizing: introducing ports/interfaces

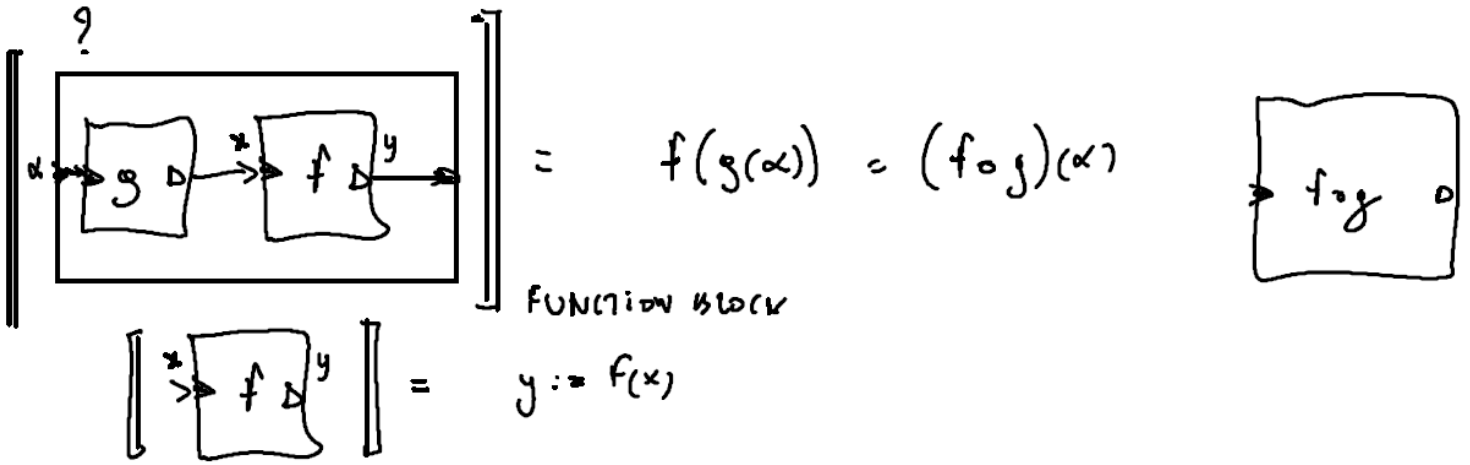




adding hierarchy to CBD formalisms

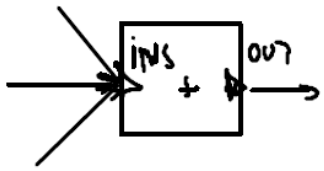
semantics?

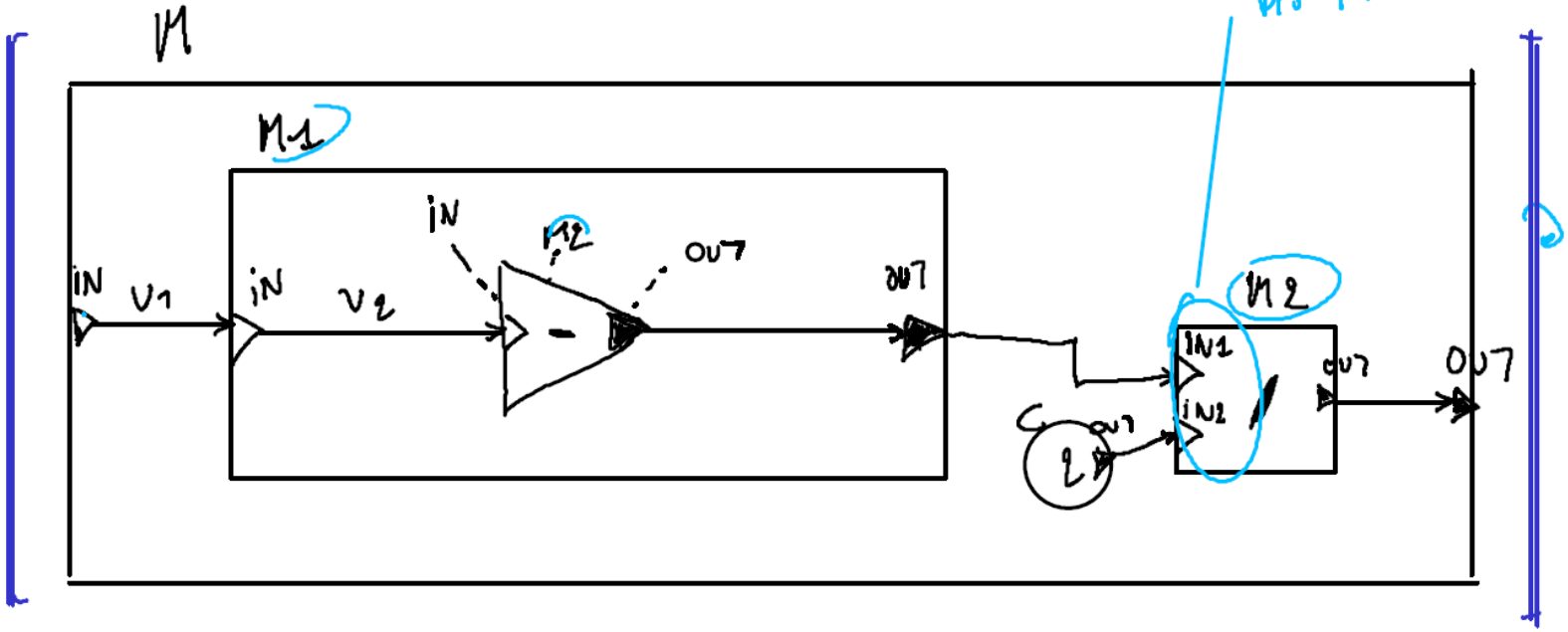
HIERARCHY

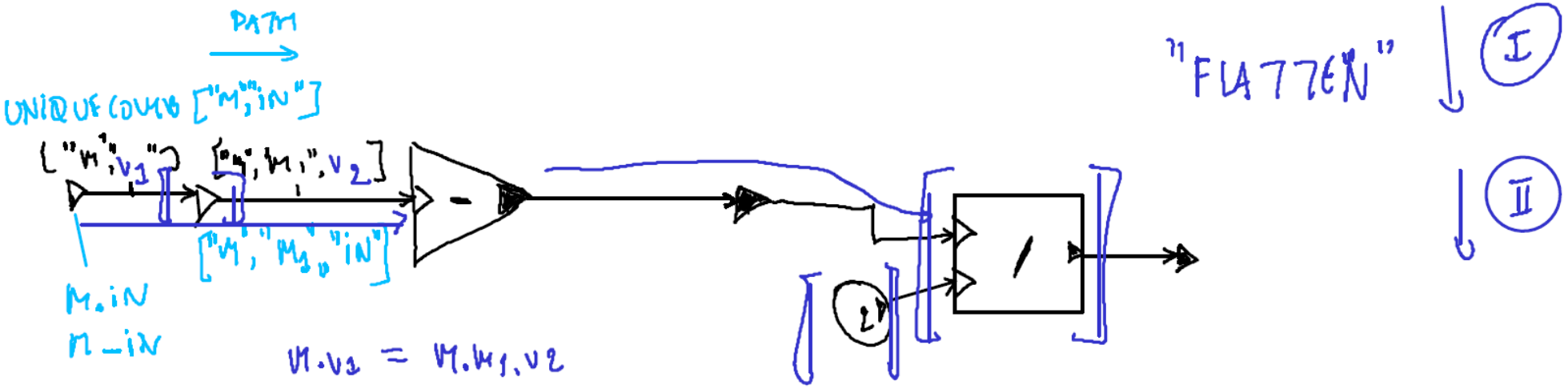


adding hierarchy to CBD formalisms

“flattening”



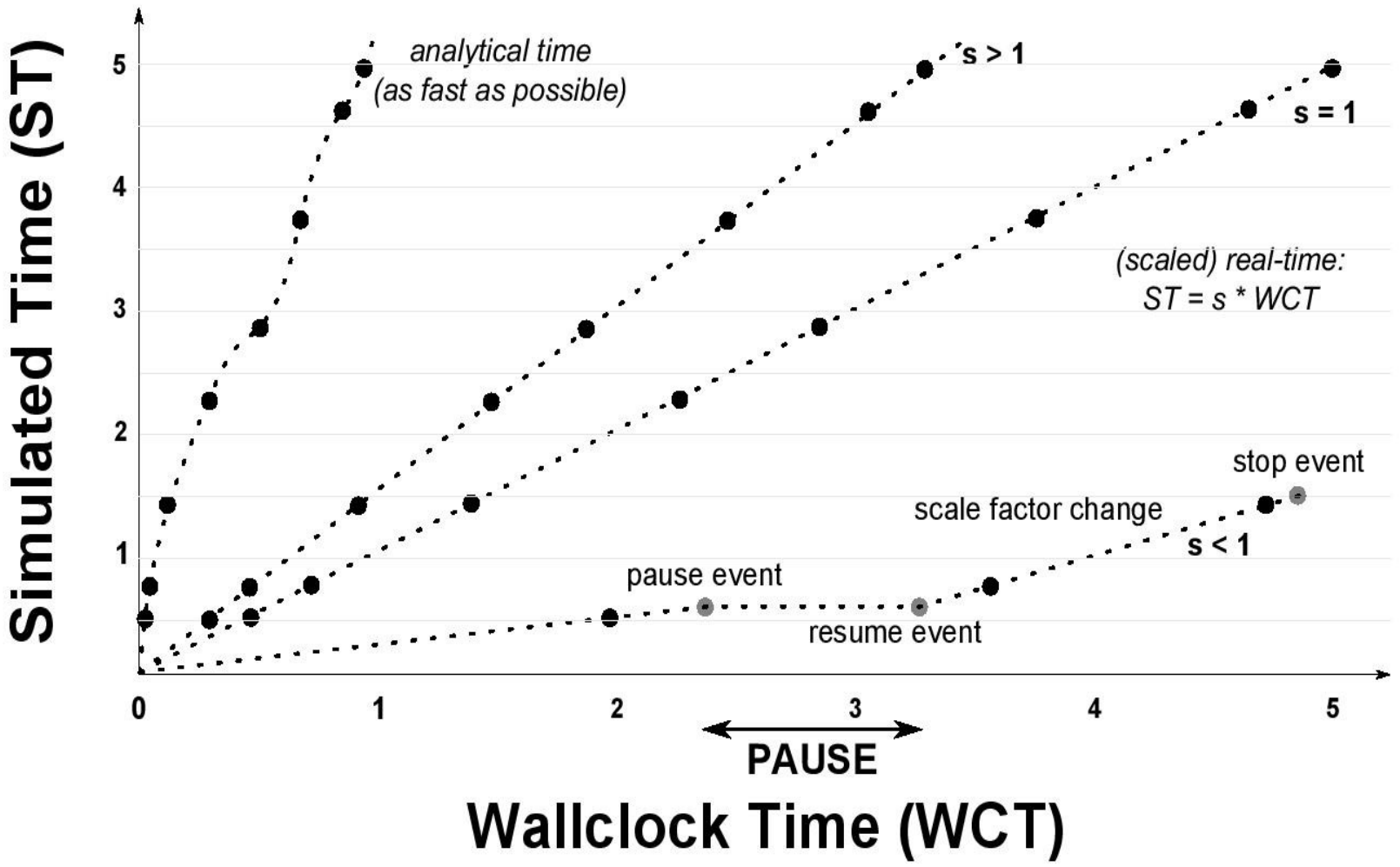




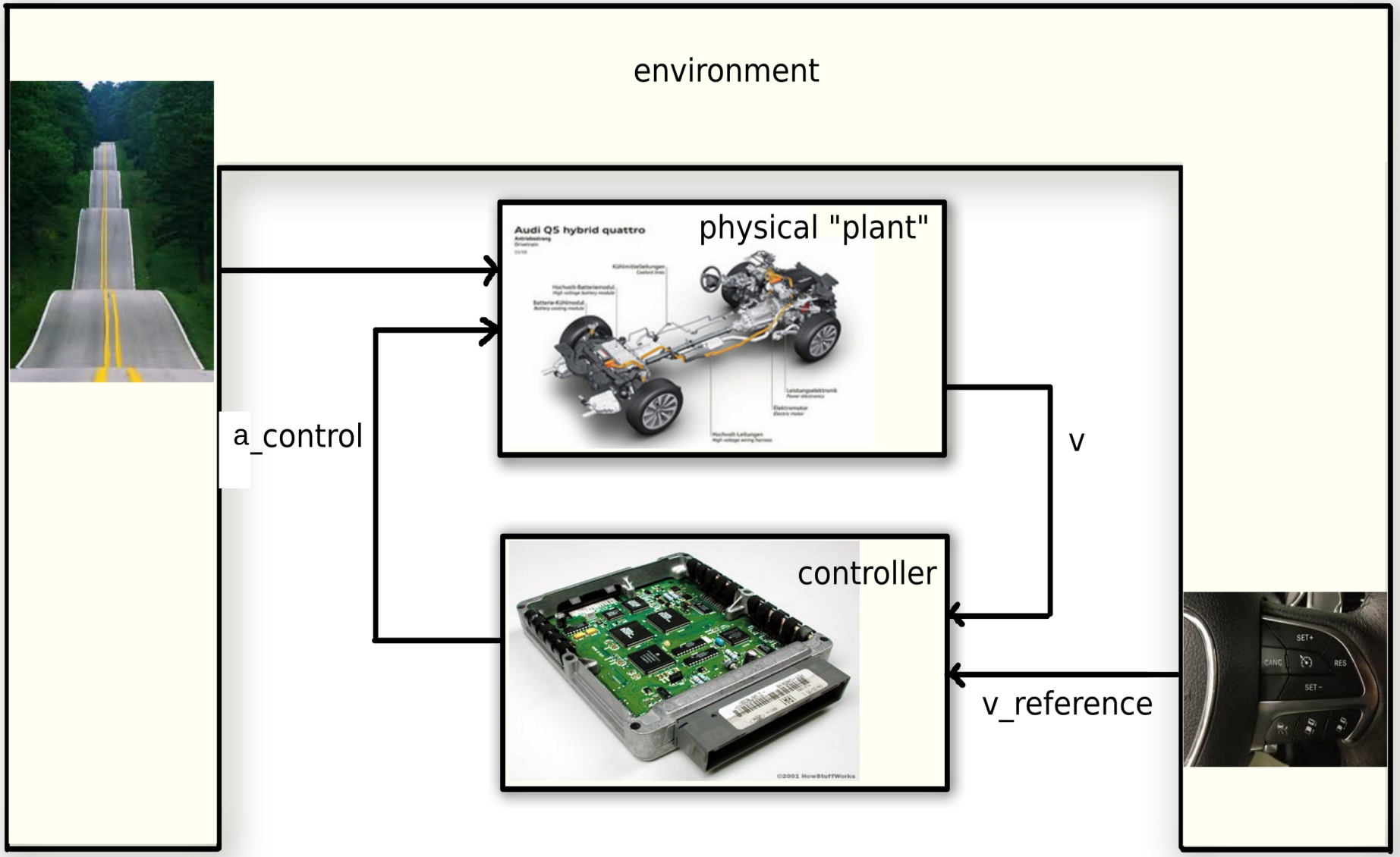
Physical Systems Modelling

- Problem-Specific (technological)
- Domain-Specific (e.g., translational mechanical)
- (general) Laws of Physics
- Power Flow/Bond Graphs (physical: energy/power)
- Computationally a-causal
  (Mathematical and Object-Oriented) ← **Modelica**
- Causal Block Diagrams (data flow)
- Numerical (Discrete) Approximations
- Computer Algorithmic + Numerical
  (Floating Point vs. Fixed Point)
- As-Fast-As-Possible vs. Real-time (XiL)
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)
- Dynamic Structure

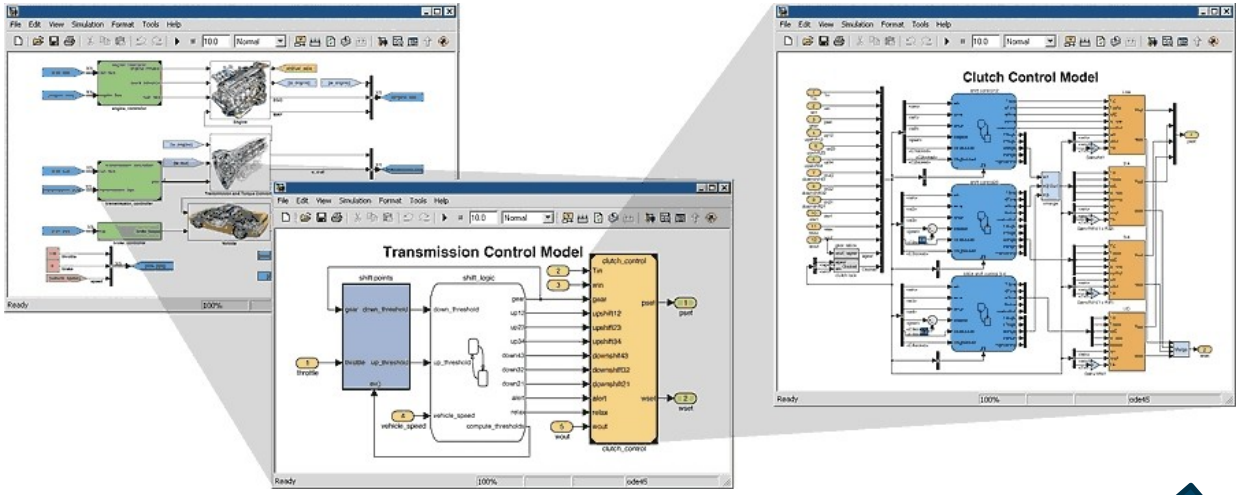
As-Fast-As-Possible vs. Real-time

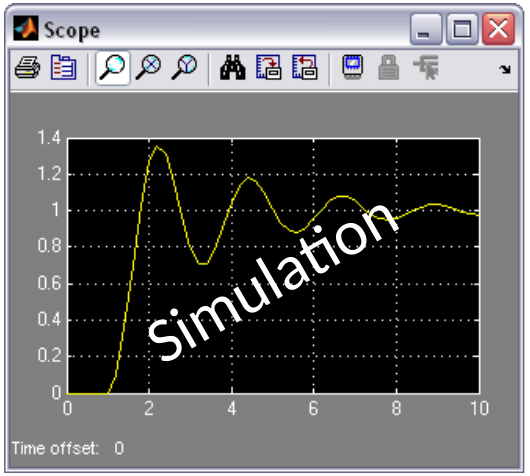


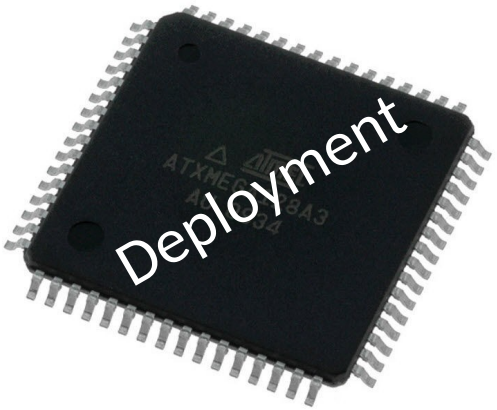
Model-Based Systems Engineering (MBSE)



Model-Based Systems Engineering (MBSE)

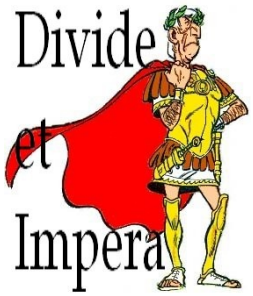


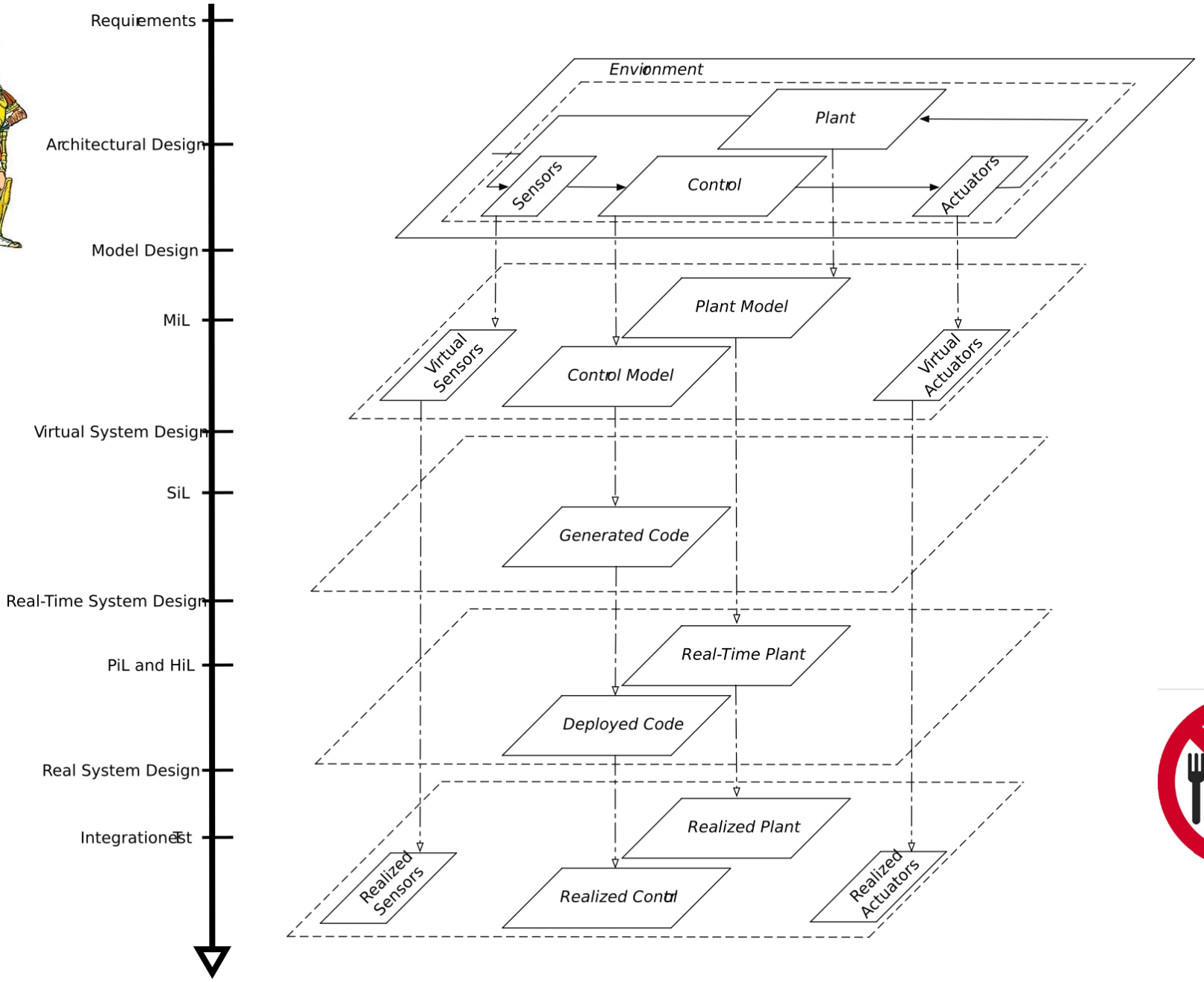




MiL, HiL, SiL, ...

XiL: X = Model, Software, Processor, Hardware





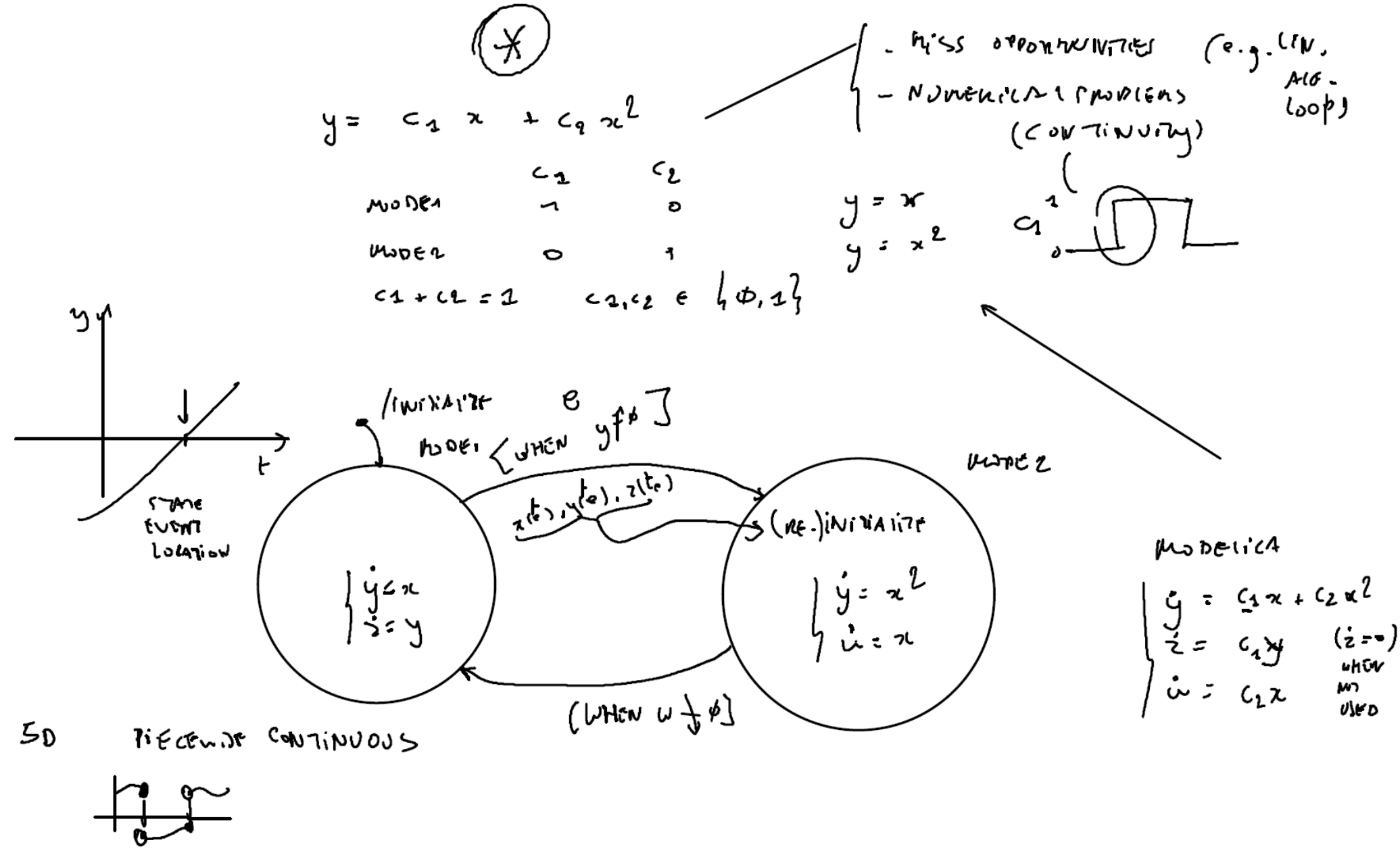
vertical consistency!

Ken Vanherpen. A contract-based approach for multi-viewpoint consistency in the concurrent design of cyber-physical systems. PhD thesis University of Antwerp. 2018.

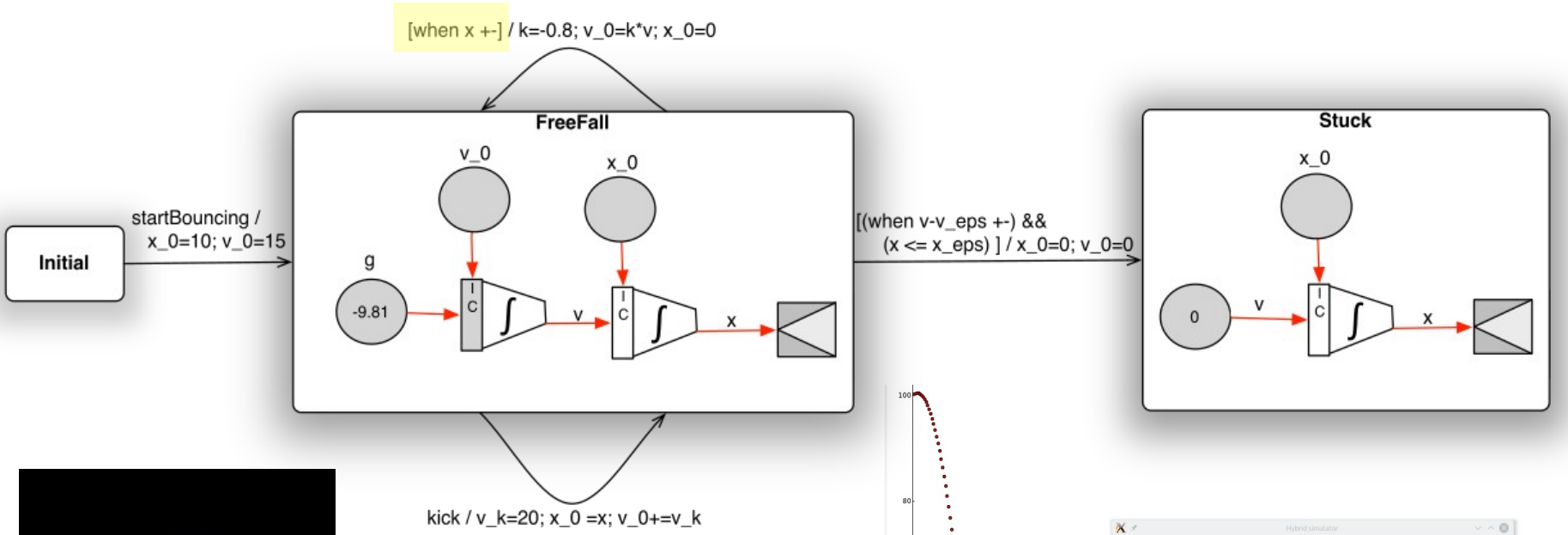
Physical Systems Modelling

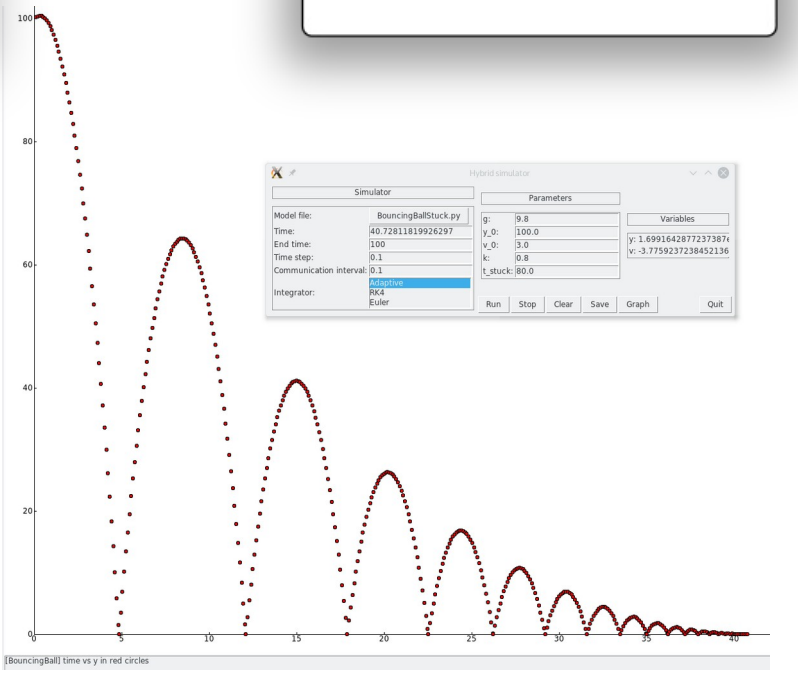
- Problem-Specific (technological)
- Domain-Specific (e.g., translational mechanical)
- (general) Laws of Physics
- Power Flow/Bond Graphs (physical: energy/power)
- Computationally a-causal
  (Mathematical and Object-Oriented) ← **Modelica**
- Causal Block Diagrams (data flow)
- Numerical (Discrete) Approximations
- Computer Algorithmic + Numerical
  (Floating Point vs. Fixed Point)
- As-Fast-As-Possible vs. Real-time (XiL)
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)
- Dynamic Structure

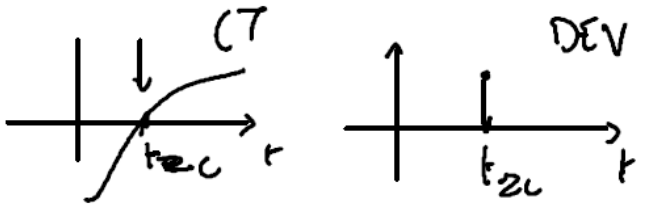
Hybrid (discrete-continuous) modelling/simulation



“bouncing ball” hybrid abstraction

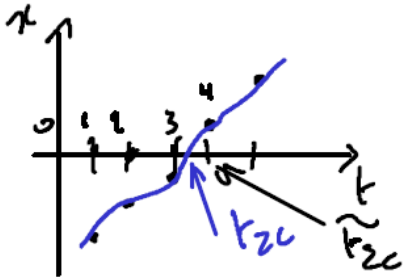


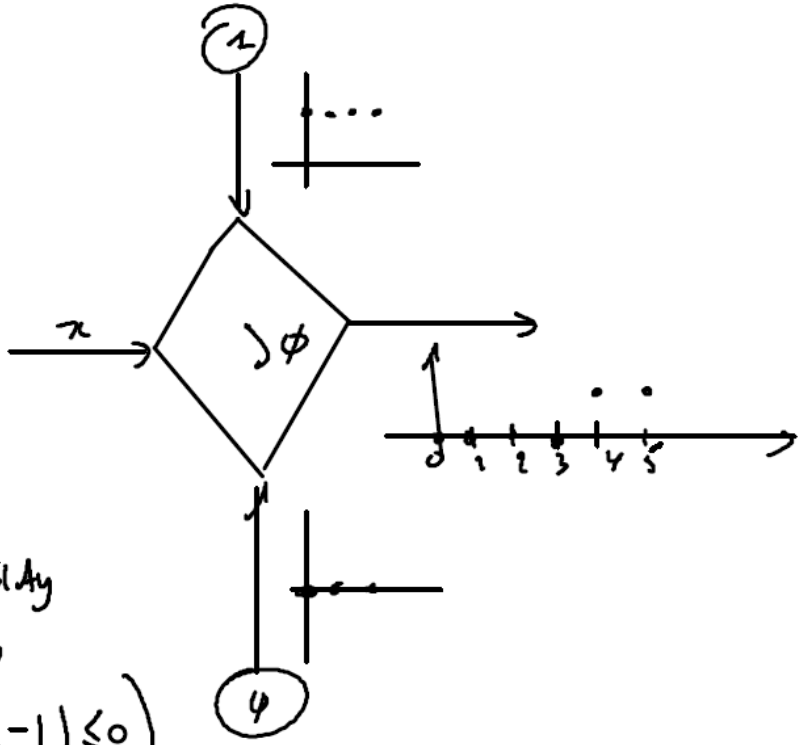




WHEN : (↓R → ↑R) WHEN ($x^{(k)} \neq \phi$)

→ DEV SIGNAL

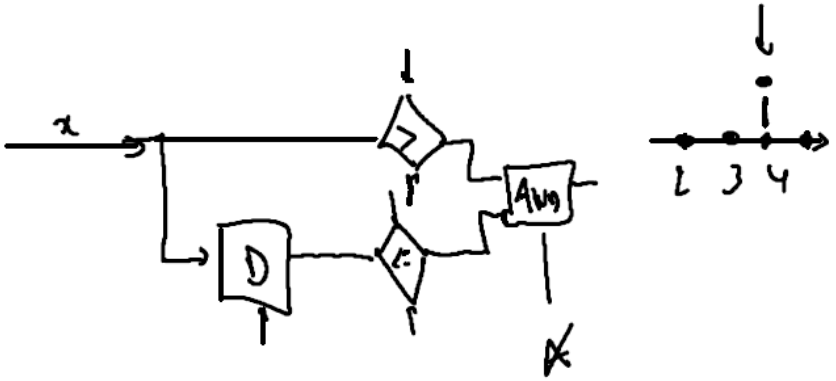




DELAY

DISCR

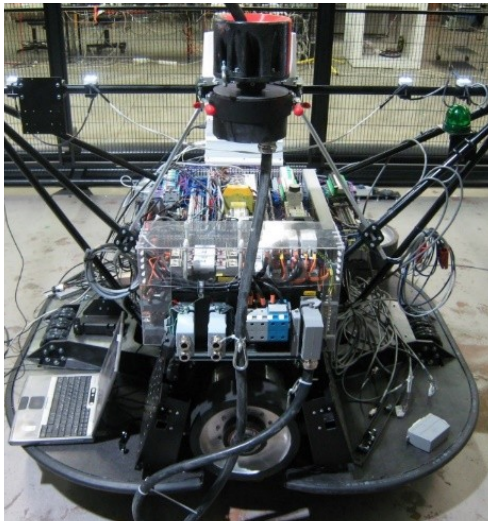
$\tilde{t}_{zc}$? : $(\tilde{x}(\tilde{t}_{zc}) > 0) \,\&\, (\tilde{x}(\tilde{t}_{zc}-1) \leq 0)$

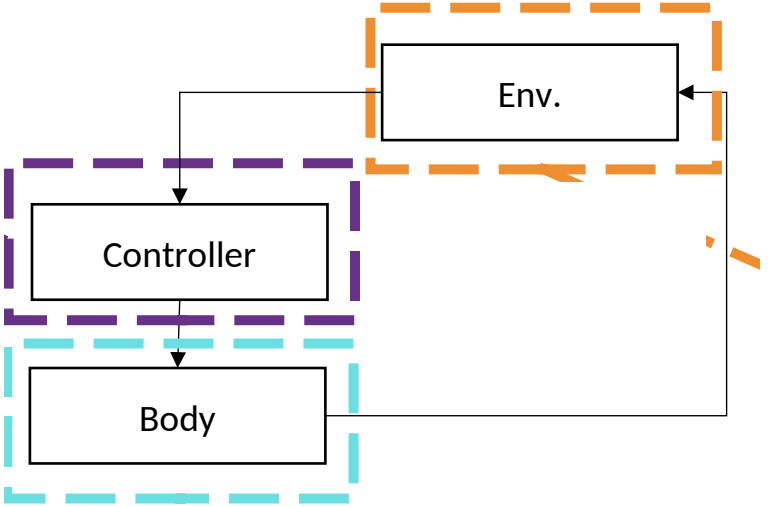


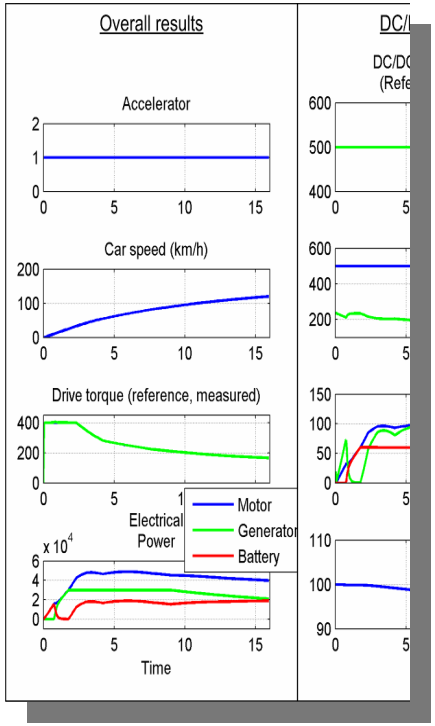
Physical Systems Modelling

- Problem-Specific (technological)
- Domain-Specific (e.g., translational mechanical)
- (general) Laws of Physics
- Power Flow/Bond Graphs (physical: energy/power)
- Computationally a-causal
  (Mathematical and Object-Oriented) ← **Modelica**
- Causal Block Diagrams (data flow)
- Numerical (Discrete) Approximations
- Computer Algorithmic + Numerical
  (Floating Point vs. Fixed Point)
- As-Fast-As-Possible vs. Real-time (XiL)
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)
- Dynamic Structure

problem: **full-system analysis**

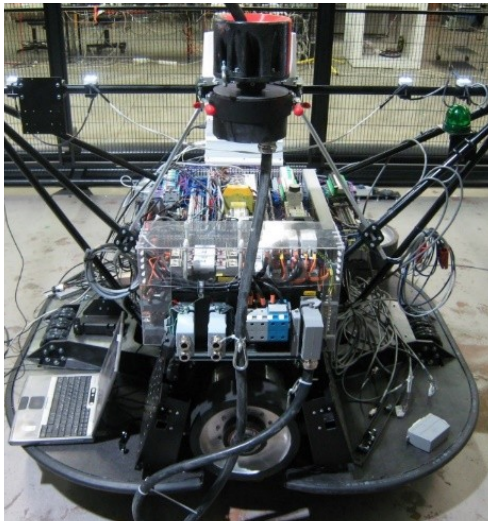


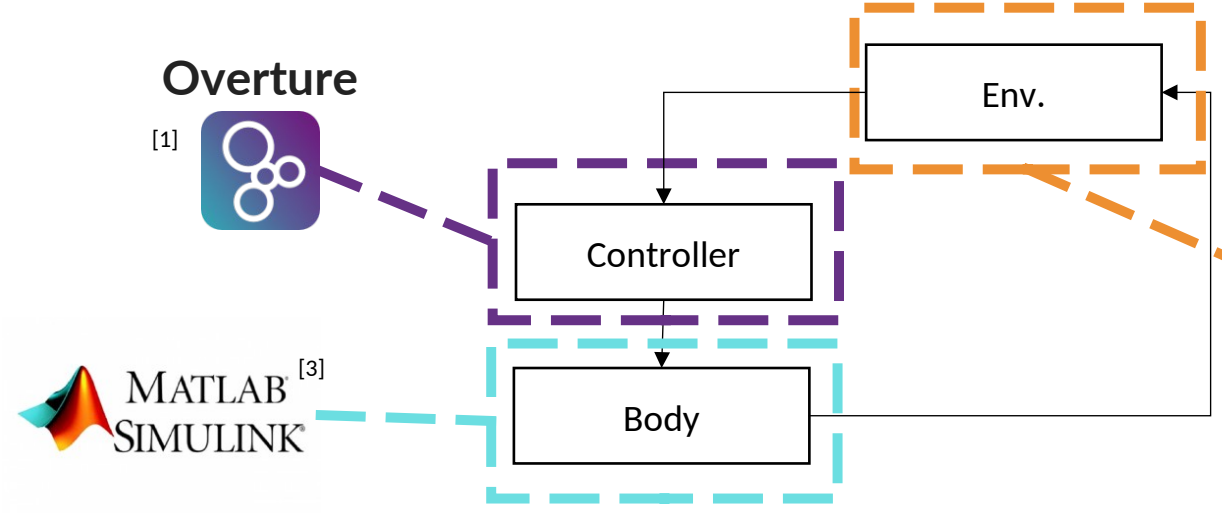


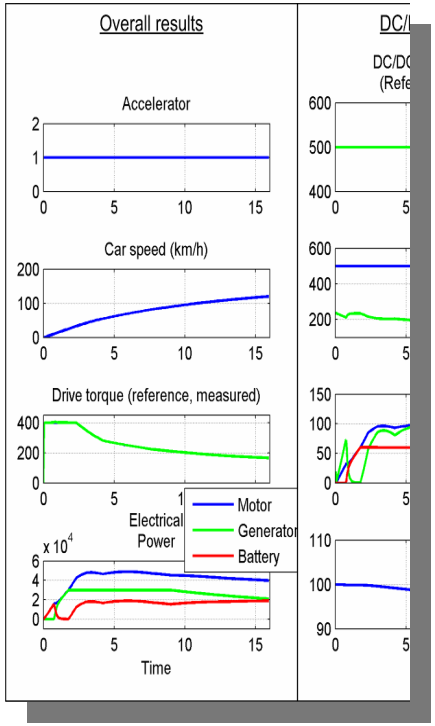


problem: **full-system analysis**

(also when IP protected)

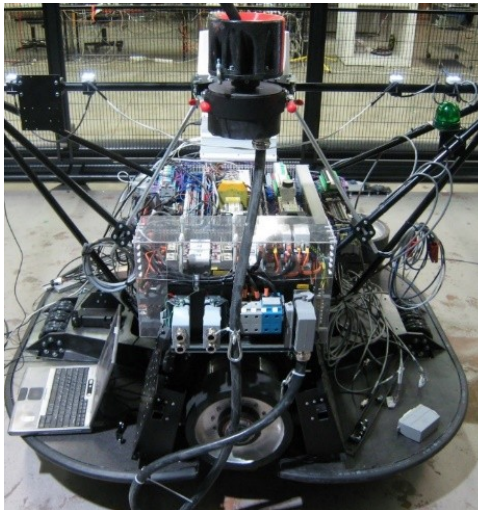


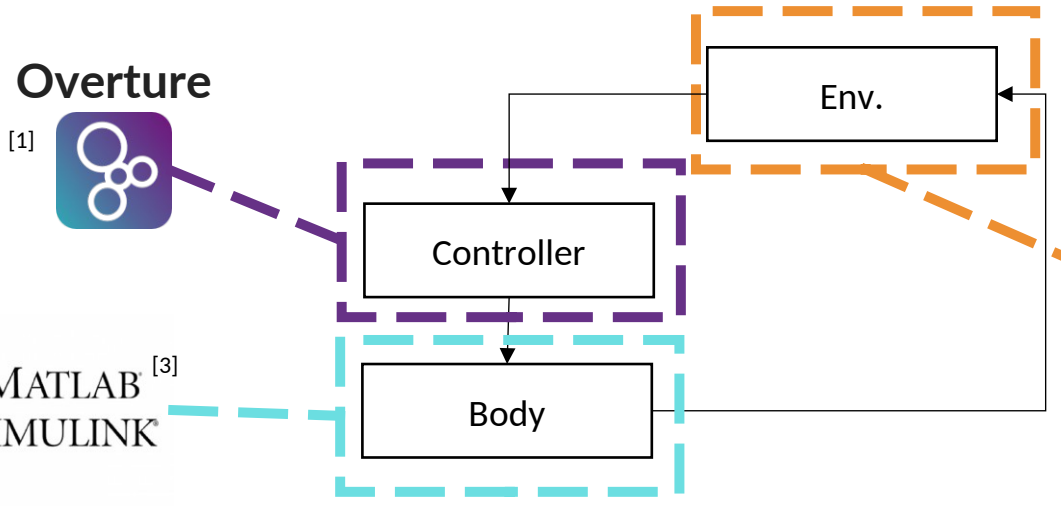




problem: **full-system analysis**

(also when IP protected)

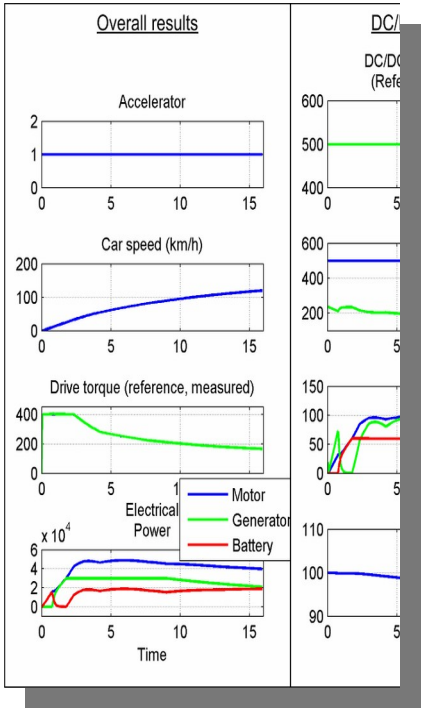




solution: combine

sub-system **simulators**

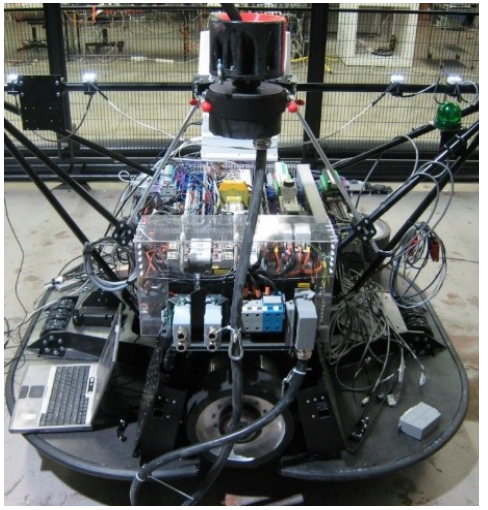
aka **co-simulation**

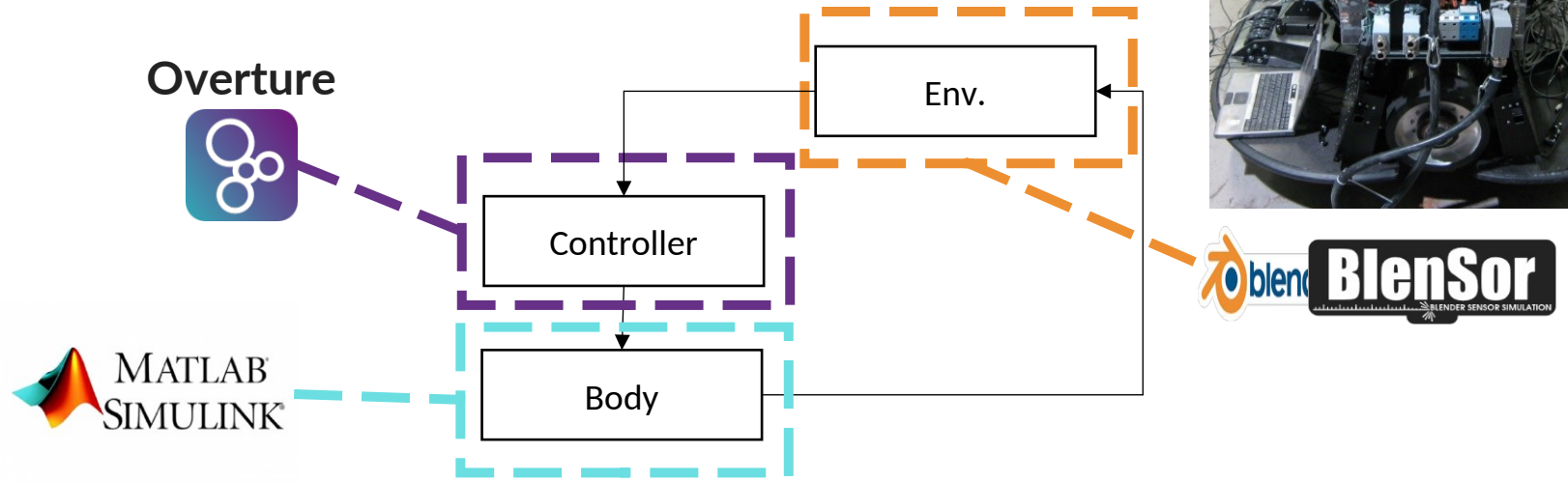


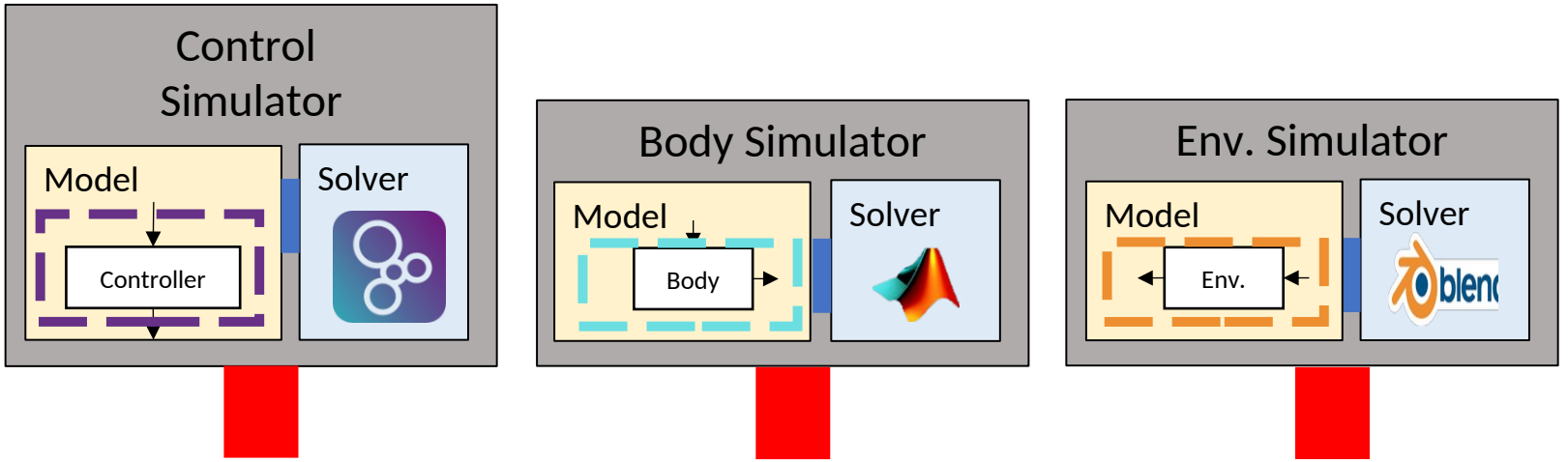
Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, Hans Vangheluwe
Co-Simulation: A Survey. ACM Comput. Surv.51(3): 49:1-49:33 (2018)

co-simulation: how?

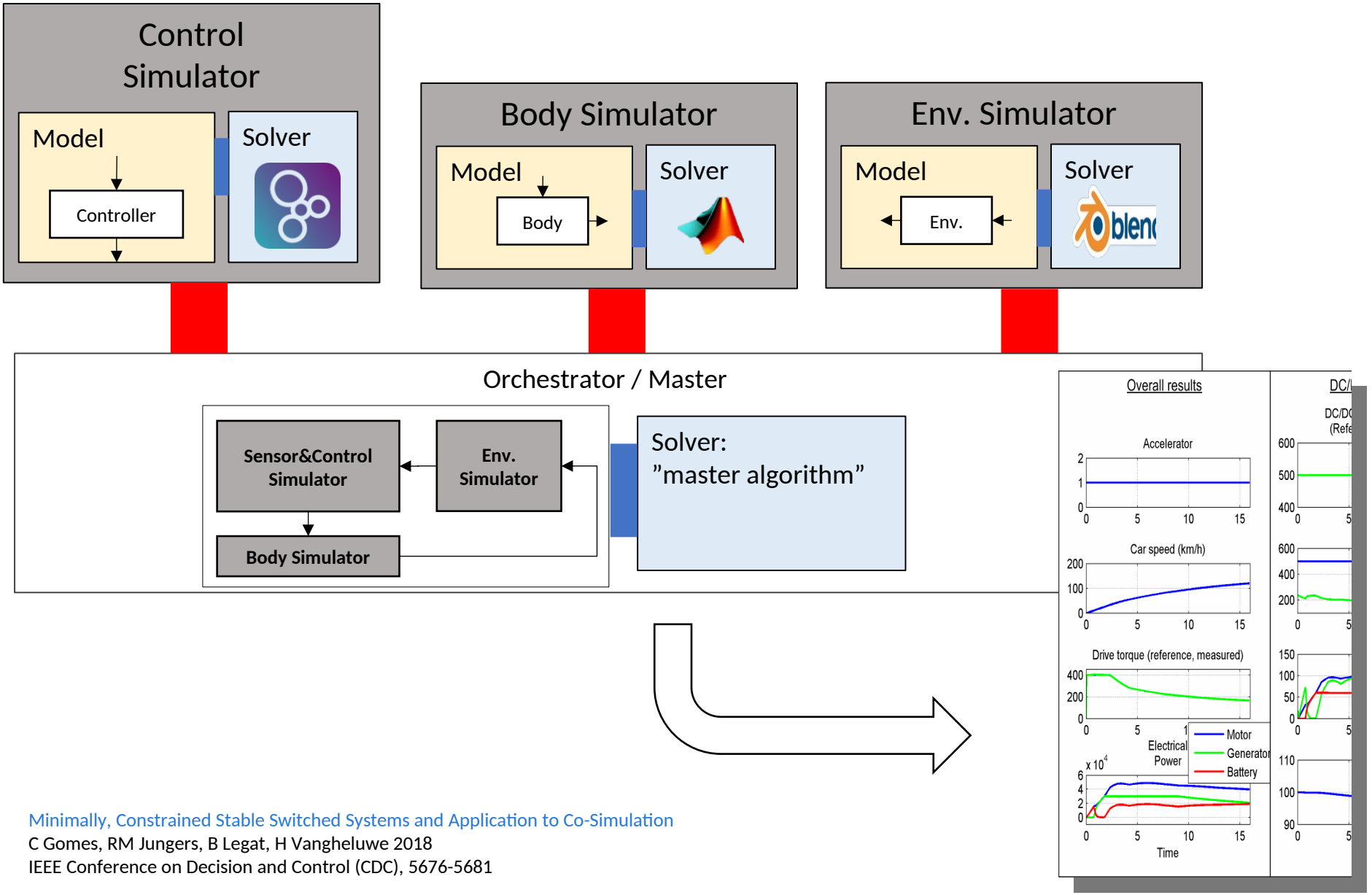
(when IP protected)





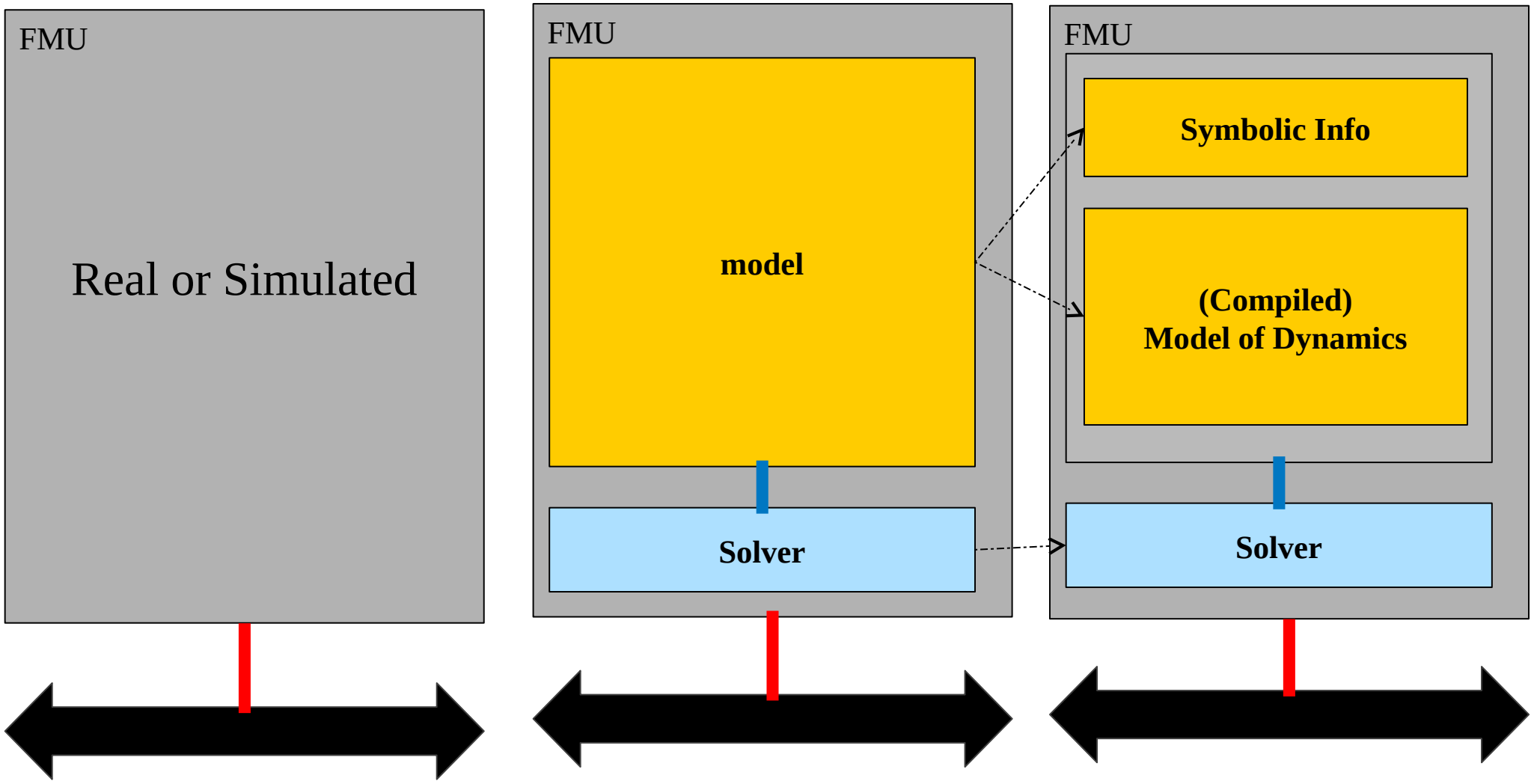


co-simulation: how?
(when IP protected)

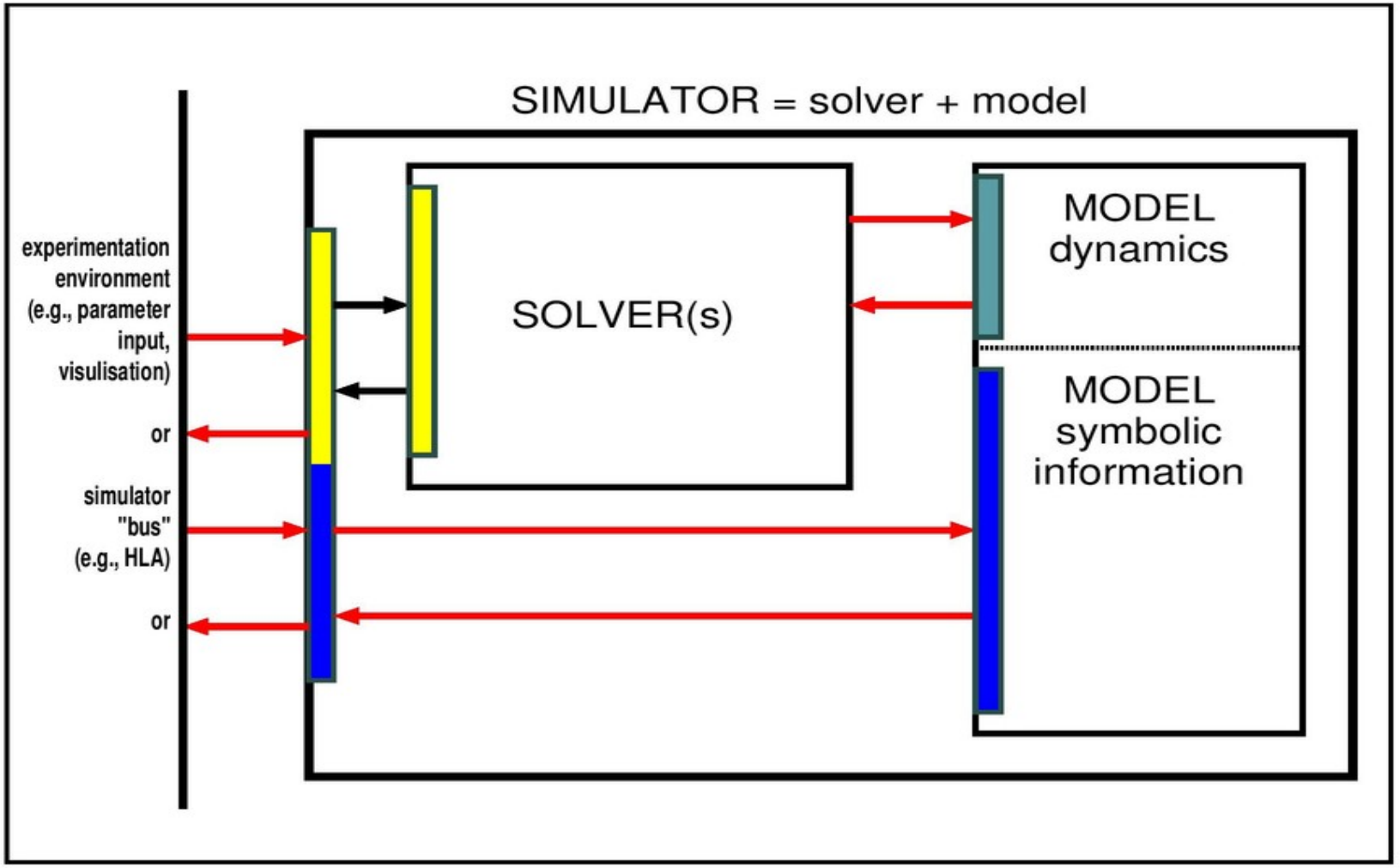


Minimally, Constrained Stable Switched Systems and Application to Co-Simulation
C Gomes, RM Jungers, B Legat, H Vangheluwe 2018
IEEE Conference on Decision and Control (CDC), 5676-5681

Deconstructing a Simulator: Functional Mockup Unit (FMU)



Model-Solver Interface
Simulator-Environment Interface



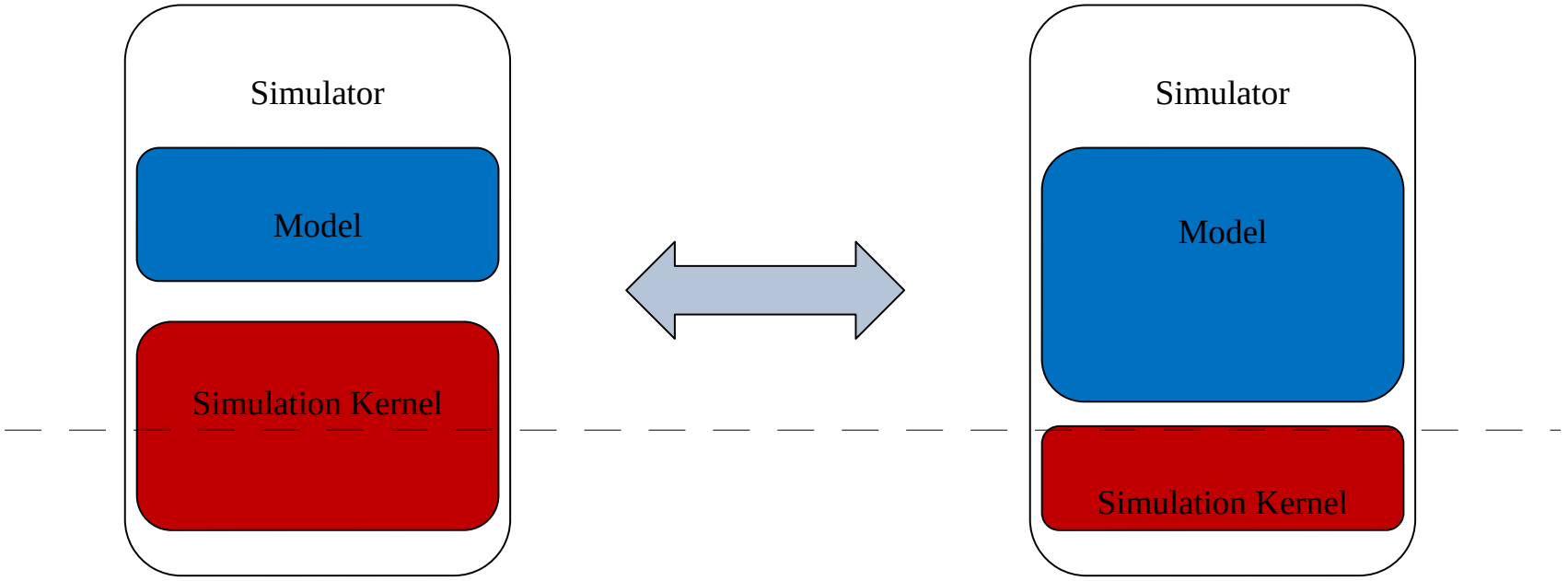
DSblock

Martin Otter and Hilding Elmquist.
The DSblock interface for exchanging model components. Eurosim '95 Simulation Congress. pp. 505- 510. 1995.

MSL-EXEC

Henk Vanhooren, Jurgen Meirlaen, Youri Amerlinck, Filip Claeys, Hans Vangheluwe, and Peter A. Vanrolleghem.
WEST: Modelling biological wastewater treatment. Journal of Hydroinformatics , 5(1):27--50, 2003.

meaningful operational semantics (Models of Computation)



Traditional Simulator

Explicit Computational Semantics “inline integration” (Cellier)

The leading standard to exchange dynamic simulation models

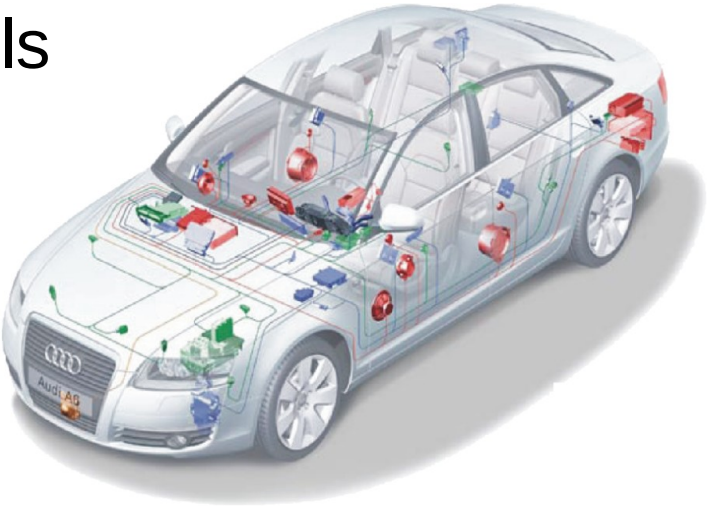
The Functional Mock-up Interface is a free standard that defines a container and an interface to exchange dynamic simulation models using a combination of XML files, binaries and C code, distributed as a ZIP file. It is supported by 180+ tools and maintained as a Modelica Association Project.

Why FMI | Complete Package 3.0.1 | Specification 3.0.1 | Implementers' Guide

https://fmi-standard.org/

Functional Mock-up Interface (FMI)

- **XML** + Binary Representation for Models
  - Standard
  - Modelling Tool Independent
  - +/- Black box …

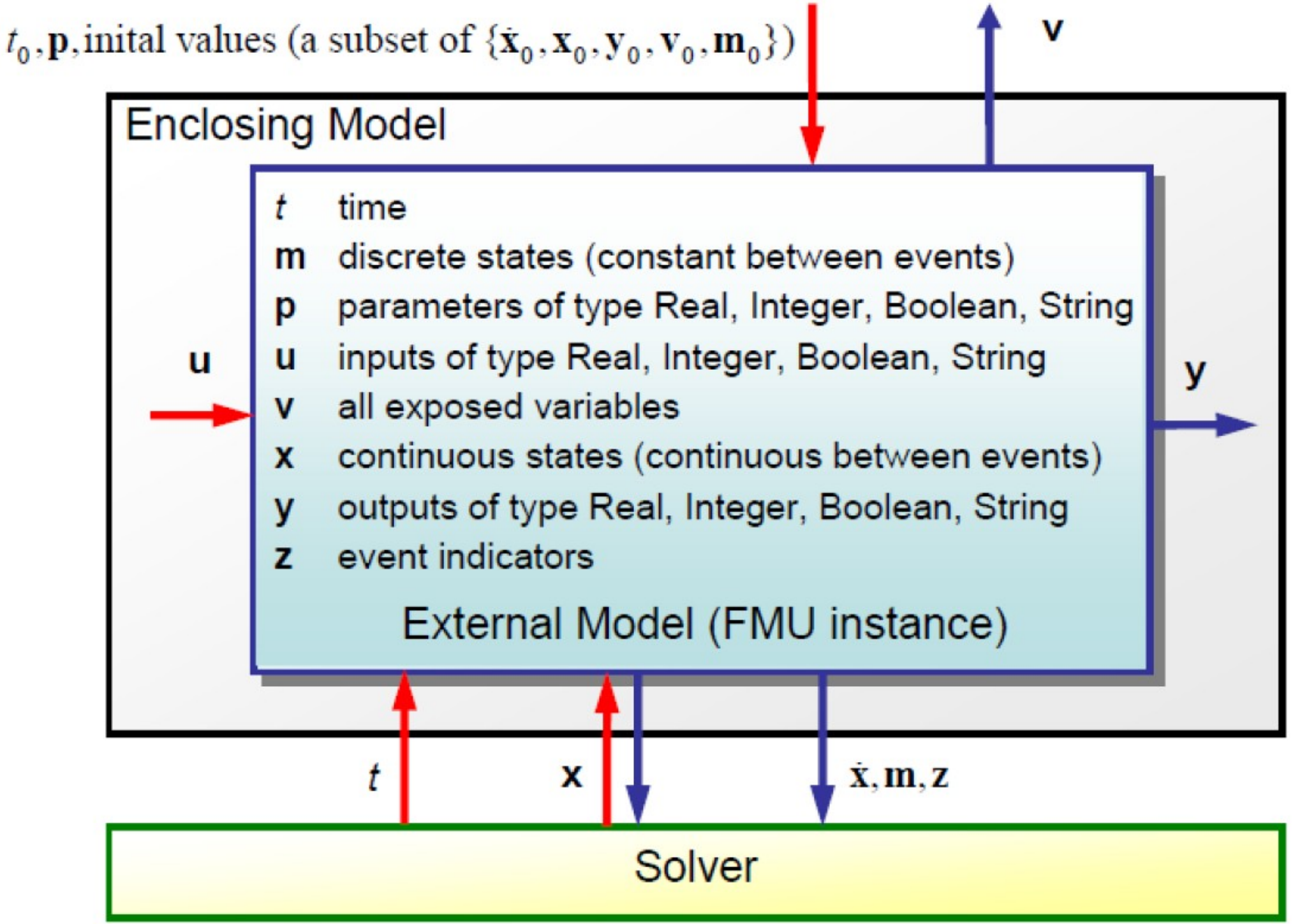


Symbolic information: FMU - XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<fmiModelDescription fmiVersion="1.0"
 modelName="BackEulerExmpl"
 modelIdentifier="BackEulerExmpl"
 guid="{7c4e810f-3da3-4a00-8276-176fa3c9f000}"
 numberOfContinuousStates="1"
 numberOfEventIndicators="0">
 <ModelVariables>
  <ScalarVariable
        name="Delay" valueReference="0">
    <Real start="0.0" fixed="true" />
  </ScalarVariable>
  <ScalarVariable
        name="stepSize" valueReference="1">
    <Real start="0.1" fixed="true" />
  </ScalarVariable>
  <ScalarVariable
        name="Product" valueReference="2">
    <Real start="0.0" fixed="true" />
  </ScalarVariable>
  <ScalarVariable
        name="Negator" valueReference="3">
    <Real start="0.0" fixed="true" />
  </ScalarVariable>
  <ScalarVariable
        name="Adder" valueReference="4">
    <Real start="0.0" fixed="true" />
  </ScalarVariable>
 </ModelVariables>
</fmiModelDescription>
```

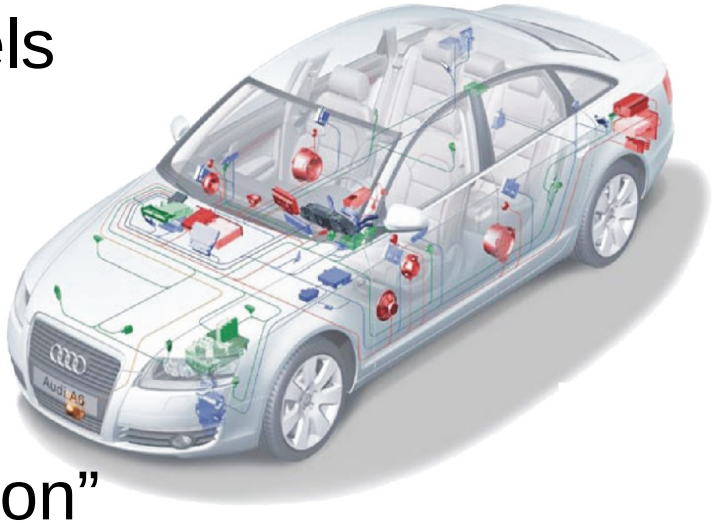
Behaviour information: FMU - C

```c
fmi2Status fmi2DoStep(fmi2Component fc , fmi2Real currentCommPoint, fmi2Real commStepSize, fmi2Boolean
    noPrevFMUState)
{
    FMUInstance* fi = (FMUInstance *)fc;
    fmi2Status simStatus = fmi2OK;
    printf("%s in fmiDoStep()\n",fi->instanceName);
    fi->currentTime = currentCommPoint + commStepSize;
    printf("Motor_in: %f\n", fi->r[_motor_in]);
    printf("slave CBD_PART2 now at time: %f\n", fi->currentTime);

    fi->r[_position] =  fi->r[_position] + fi->r[_velocity] * commStepSize;
    fi->r[_velocity] = fi->r[_velocity] + fi->r[_acceleration_after_friction] * commStepSize;
    fi->r[_friction] = fi->r[_velocity] * 5.81;
    fi->r[_motor_acceleration] = fi->r[_motor_in] * 40;
    fi->r[_acceleration_after_friction] = fi->r[_motor_acceleration] - fi->r[_friction];

    return simStatus;
}

fmi2Status fmi2GetReal(fmi2Component fc, const fmi2ValueReference vr[], size_t nvr, fmi2Real value[])
{
    FMUInstance* comp = (FMUInstance *)fc;
    int i;
    for (i = 0; i < nvr; i++)
    {
        value[i] = comp->r[(vr[i])];
    }
    return fmi2OK;
}
```

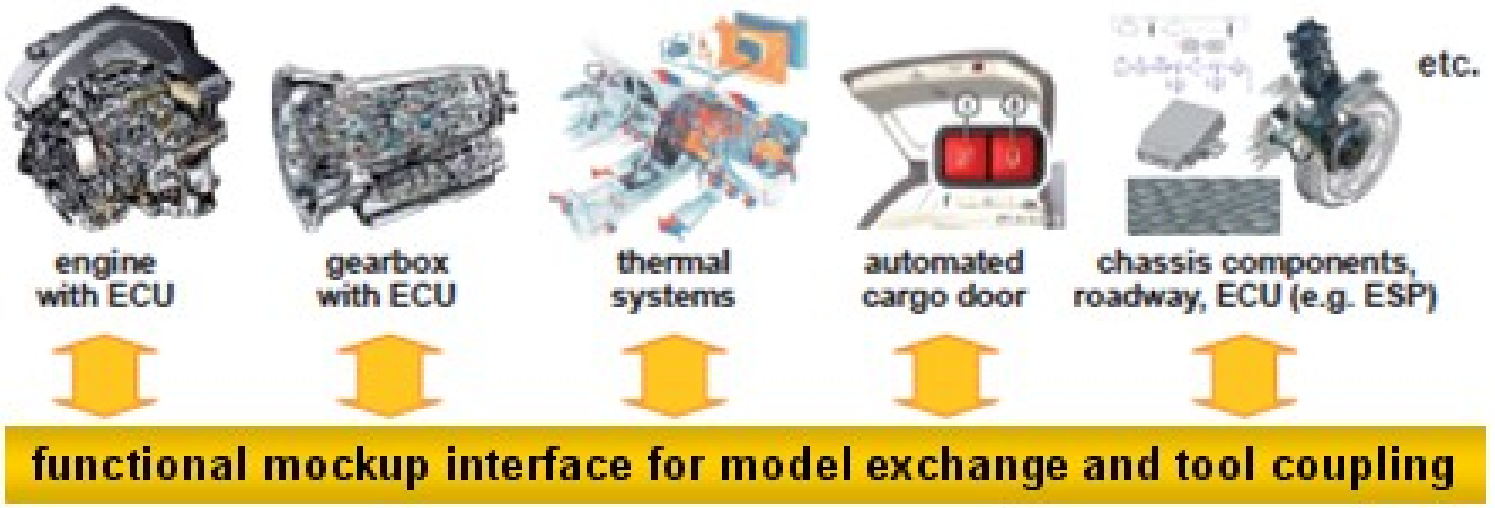

$t_0, \mathbf{p}$, inital values (a subset of $\{\dot{\mathbf{x}}_0, \mathbf{x}_0, \mathbf{y}_0, \mathbf{v}_0, \mathbf{m}_0\}$)
v
Enclosing Model
t time
m discrete states (constant between events)
p parameters of type Real, Integer, Boolean, String
u inputs of type Real, Integer, Boolean, String
v all exposed variables
x continuous states (continuous between events)
y outputs of type Real, Integer, Boolean, String
z event indicators
u
y
External Model (FMU instance)
t
x
$\dot{\mathbf{x}}, \mathbf{m}, \mathbf{z}$
Solver

Functional Mock-up Interface (FMI)

- **XML** + Binary Representation for Models
  - Standard
  - Modelling Tool Independent
  - +/- Black box …

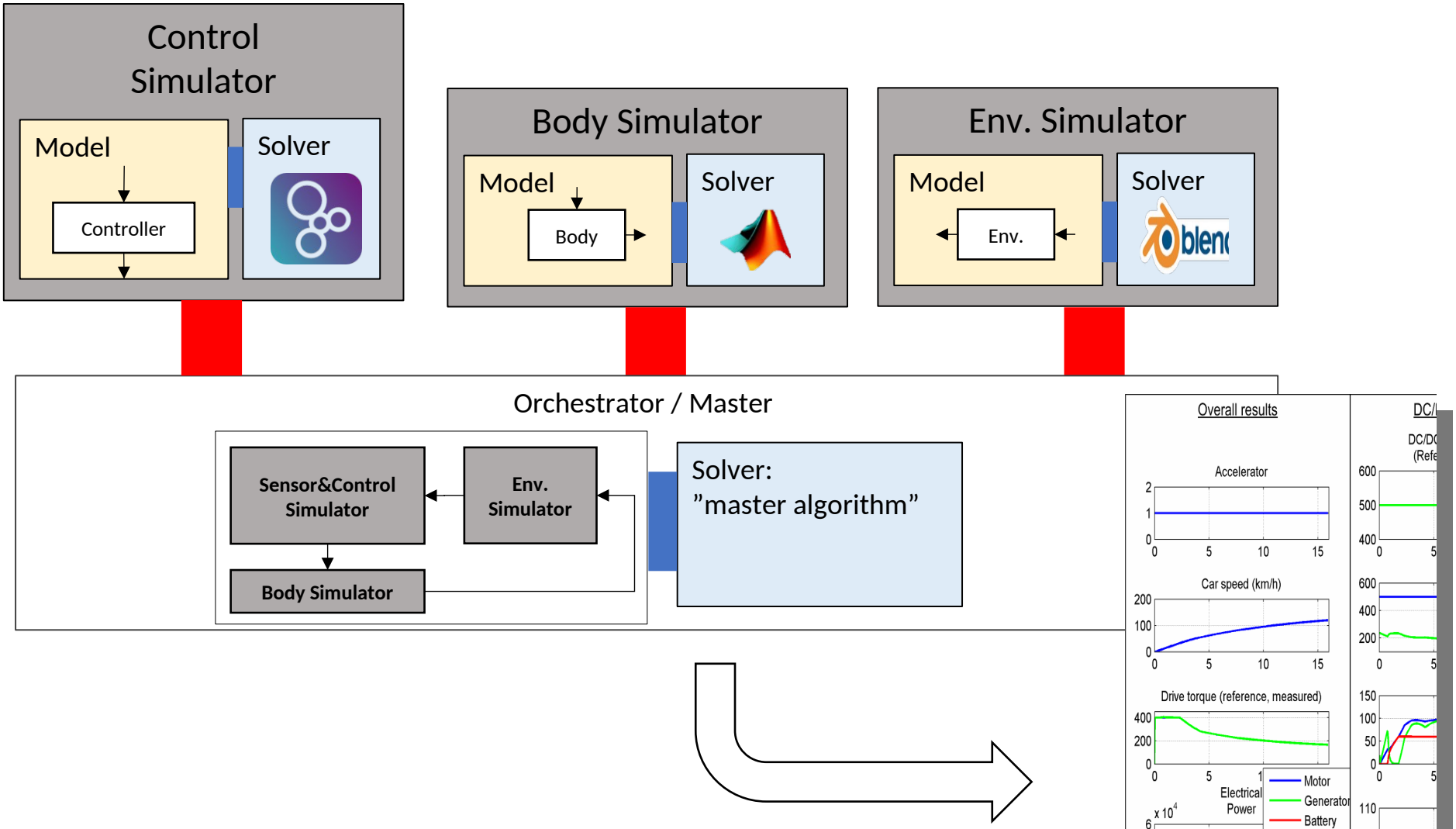


- Composed FMUs still need “orchestration”



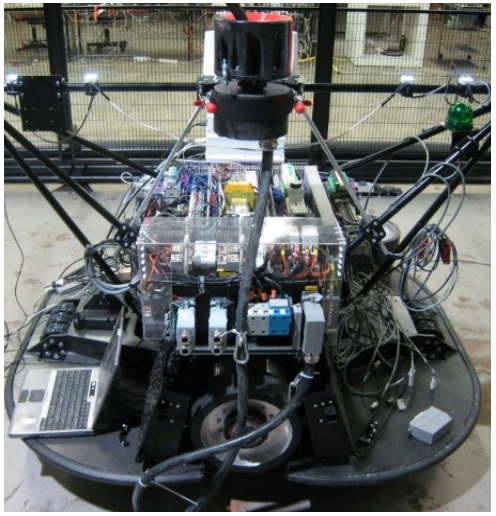
www.fmi-standard.org

co-simulation: how?
(when IP protected)



Minimally, Constrained Stable Switched Systems and Application to Co-Simulation
C Gomes, RM Jungers, B Legat, H Vangheluwe 2018
IEEE Conference on Decision and Control (CDC), 5676-5681

Co-simulation: how?



Overture

Env.

Controller

Body

MATLAB SIMULINK

BlenSor

FUNCTIONAL MOCK-UP INTERFACE

[1]

Control Simulator

FMU.zip
|-- .dll or .so
|-- .xml

Body Simulator

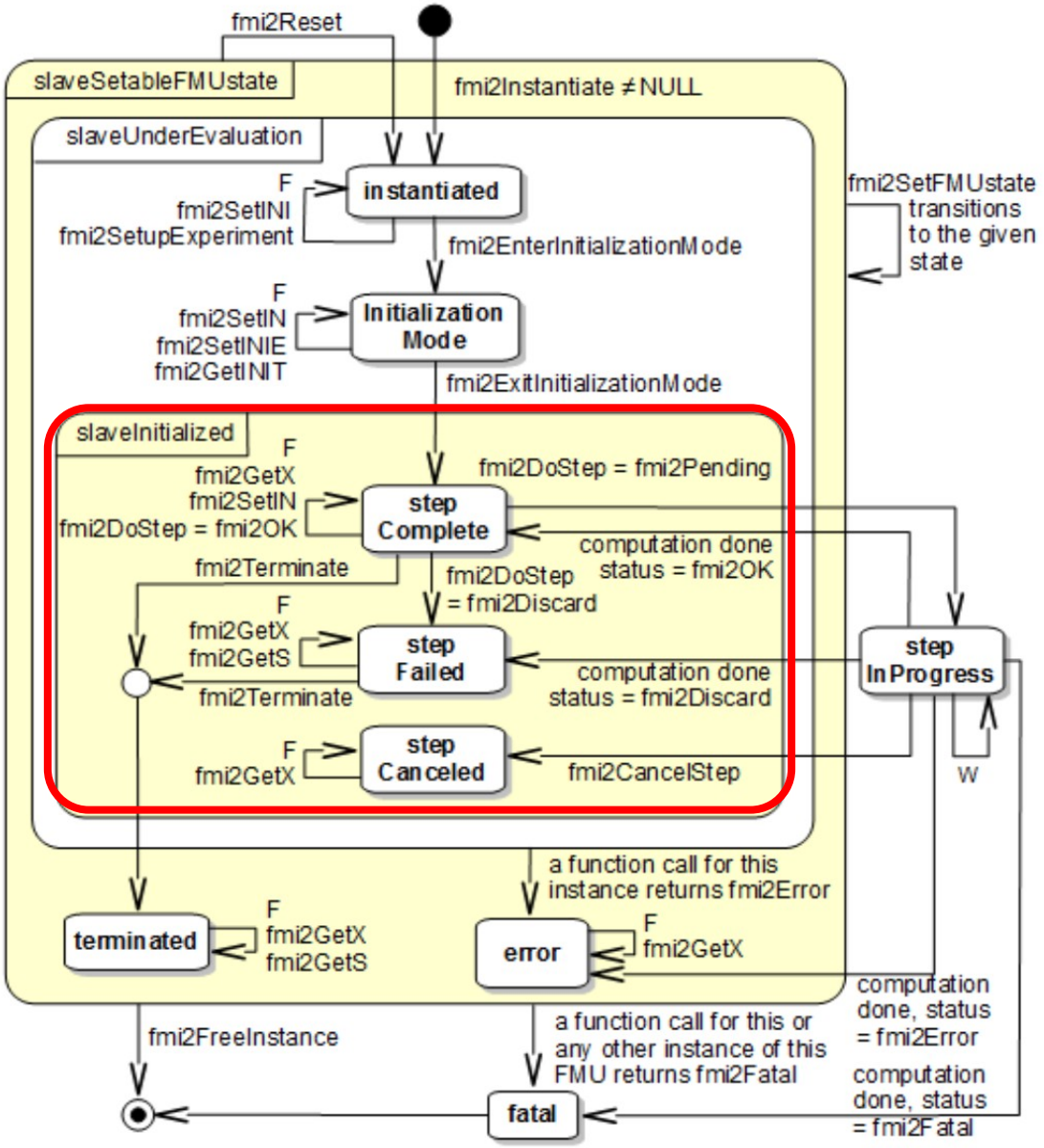
Env. Simulator

fmi2DoStep(…),
fmi2SetReal(…), …

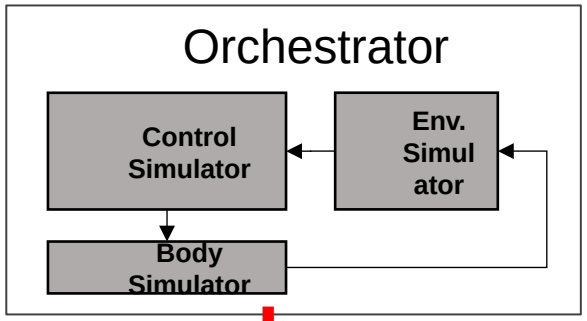
[1] www.fmi-standard.org

Co-simulation master

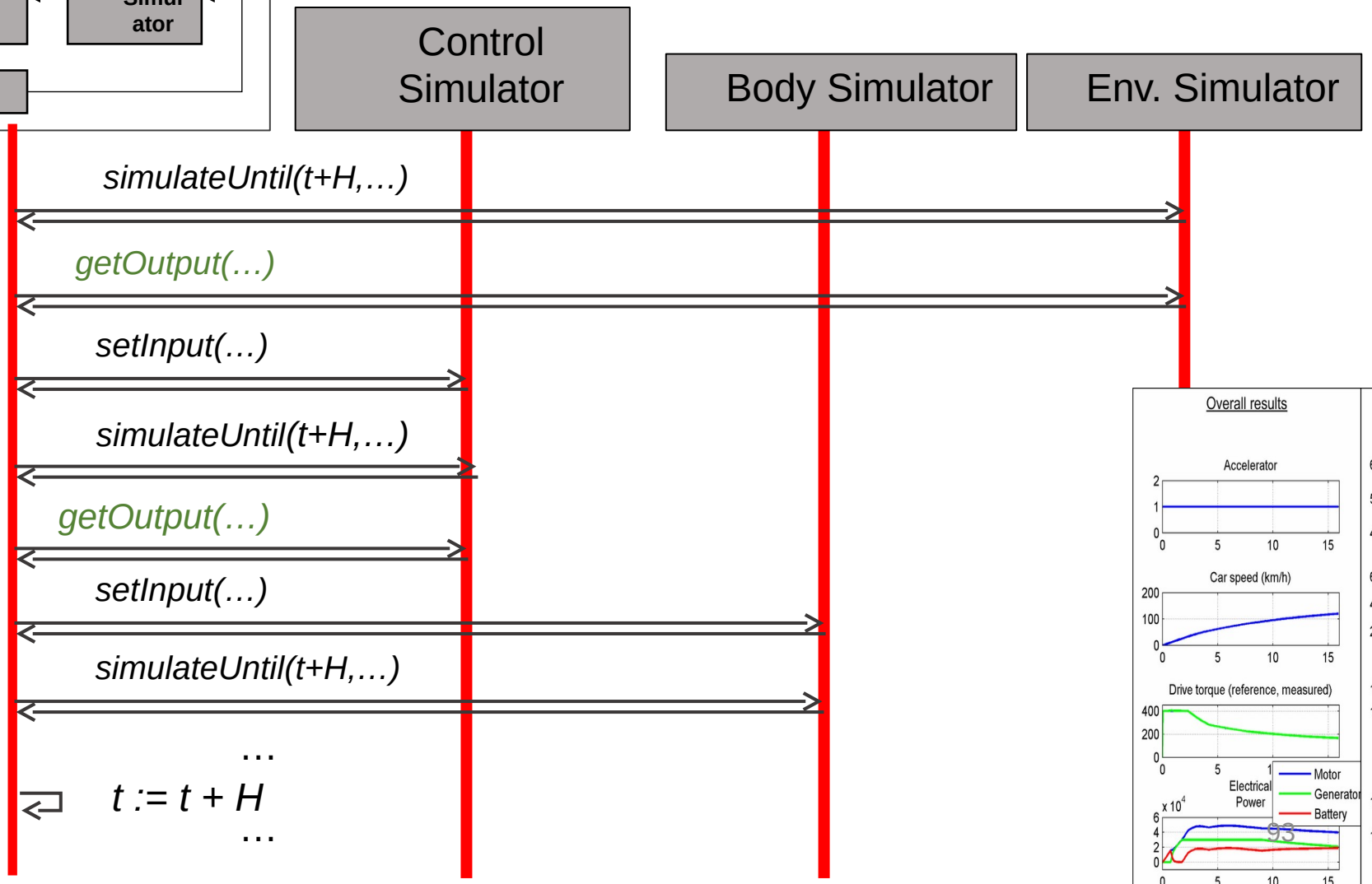
State machine calling sequence Master-Slave



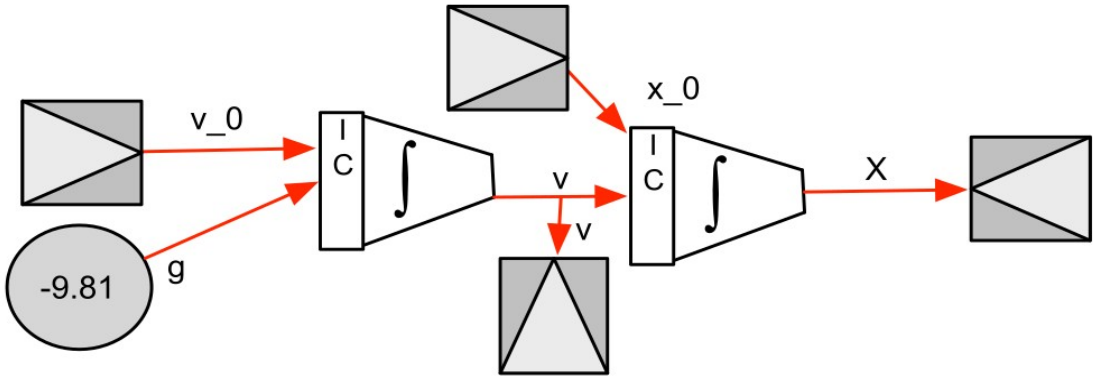
Co-simulation: how?



Gu, B., & Asada, H. H. (2001). Co-simulation of algebraically coupled dynamic subsystems. In *American Control Conference, 2001. Proceedings of the 2001* (Vol. 3, pp. 2273–2278 vol.3). http://doi.org/10.1109/ACC.2001.946089



Looking inside a simulator/solver for Causal Block Diagrams (CBDs)

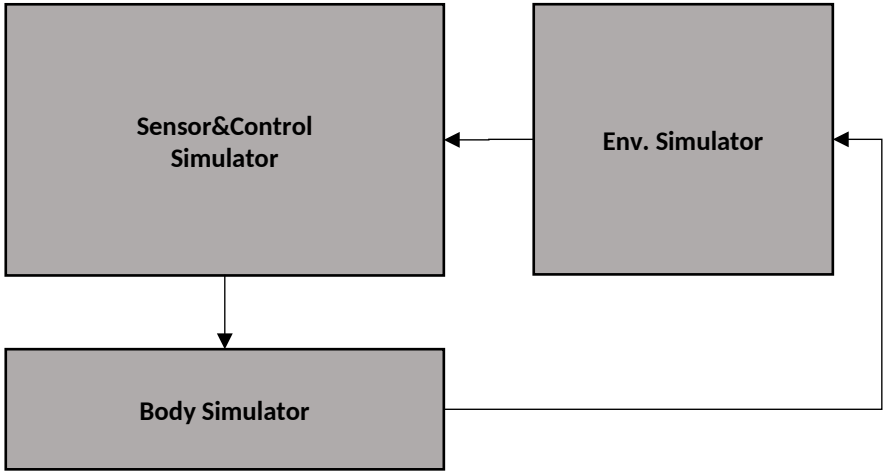


$logicalTime \leftarrow 0$
while not end_condition **do**
    $schedule \leftarrow LOOPDETECT(DEPGRAPH(cbd))$
    for gblock **in** schedule **do**
        $COMPUTE(gblock)$
    end for
    $logicalTime \leftarrow logicalTime + \Delta t$
end while

CBD simulation algorithm can also be used for scheduling at level of Master

need “zero-delay feedthrough” information (algebraic loops)

Caveat: naive

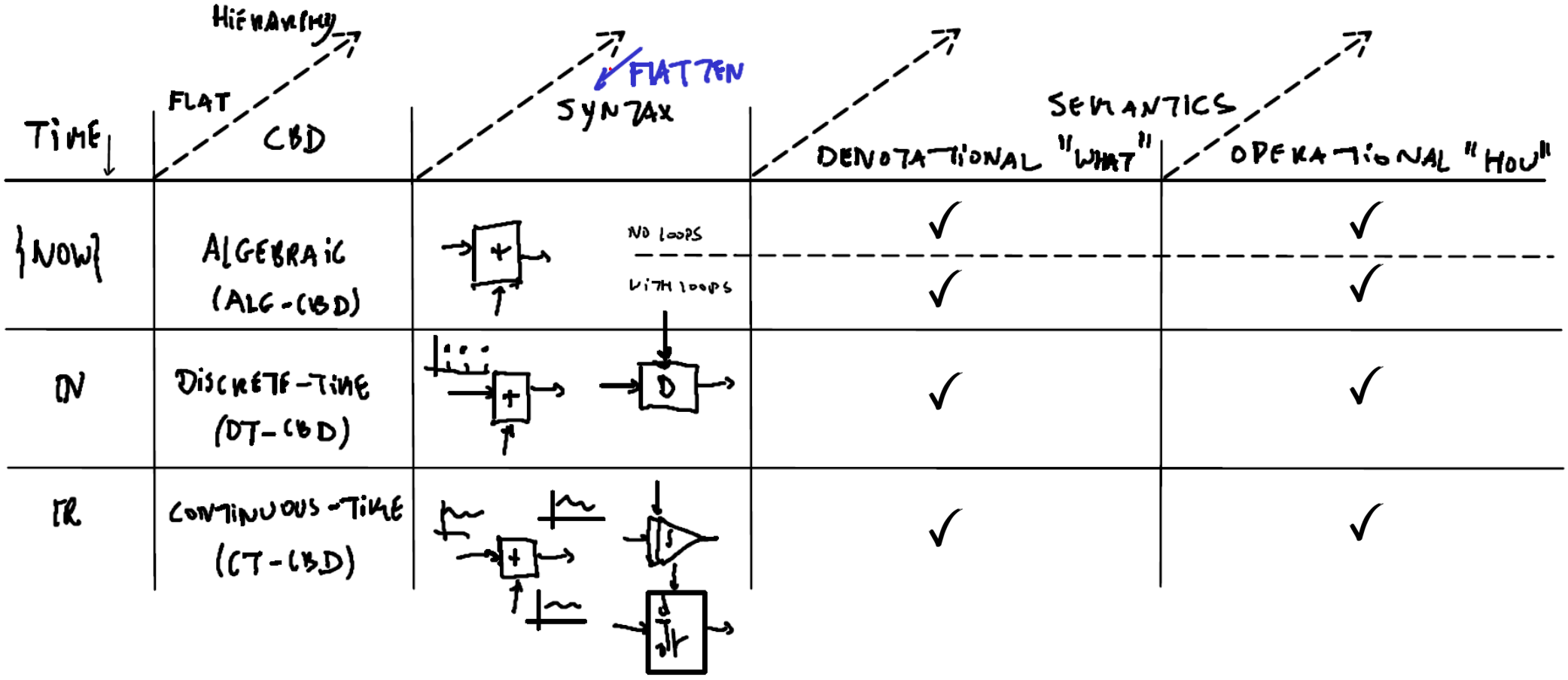


$logicalTime \leftarrow 0$
while not end_condition **do**
    $schedule \leftarrow LOOPDETECT(DEPGRAPH(cbd))$
    for gblock **in** schedule **do**
        $COMPUTE(gblock)$
    end for
    $logicalTime \leftarrow logicalTime + \Delta t$
end while

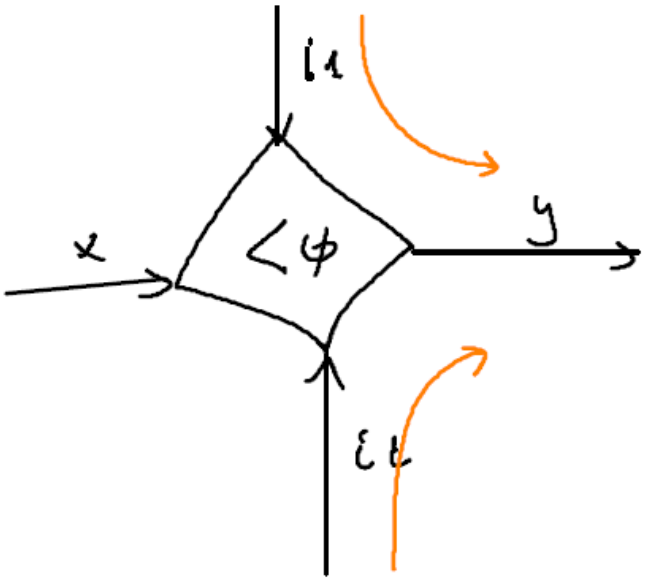
Physical Systems Modelling

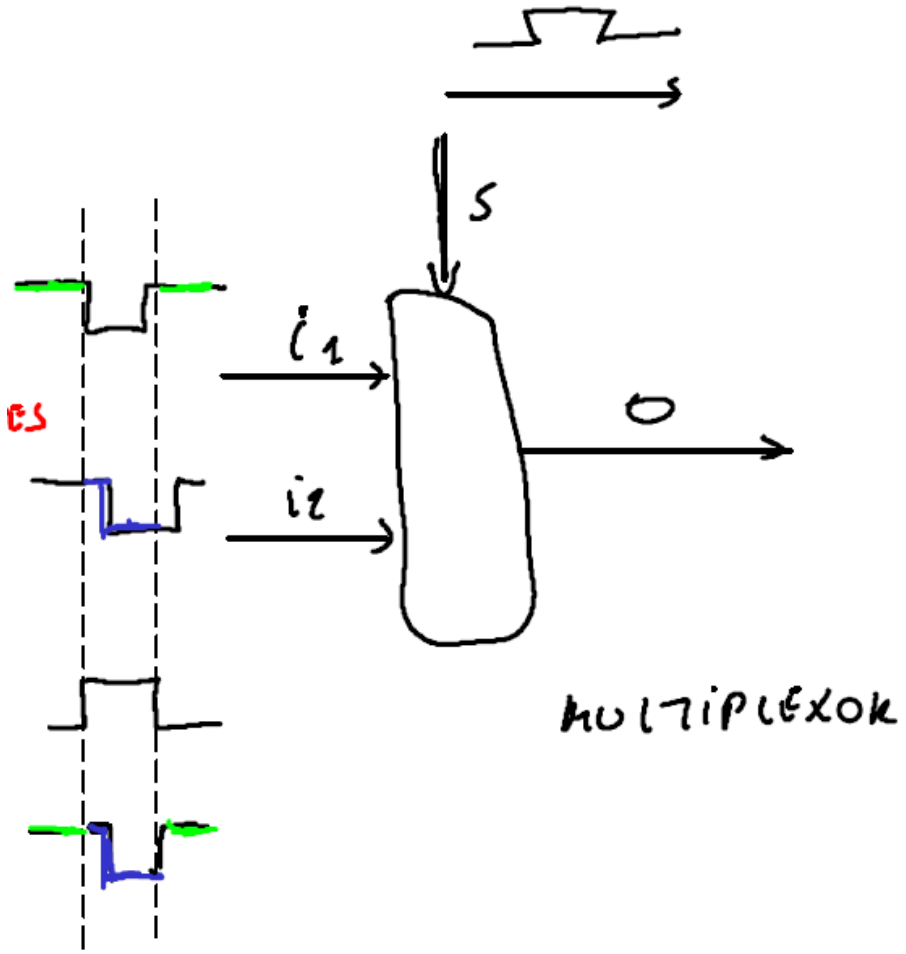
- Problem-Specific (technological)
- Domain-Specific (e.g., translational mechanical)
- (general) Laws of Physics
- Power Flow/Bond Graphs (physical: energy/power)
- Computationally a-causal
  (Mathematical and Object-Oriented) ← **Modelica**
- Causal Block Diagrams (data flow)
- Numerical (Discrete) Approximations
- Computer Algorithmic + Numerical
  (Floating Point vs. Fixed Point)
- As-Fast-As-Possible vs. Real-time (XiL)
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)
- Dynamic Structure

++ Dynamic Structure

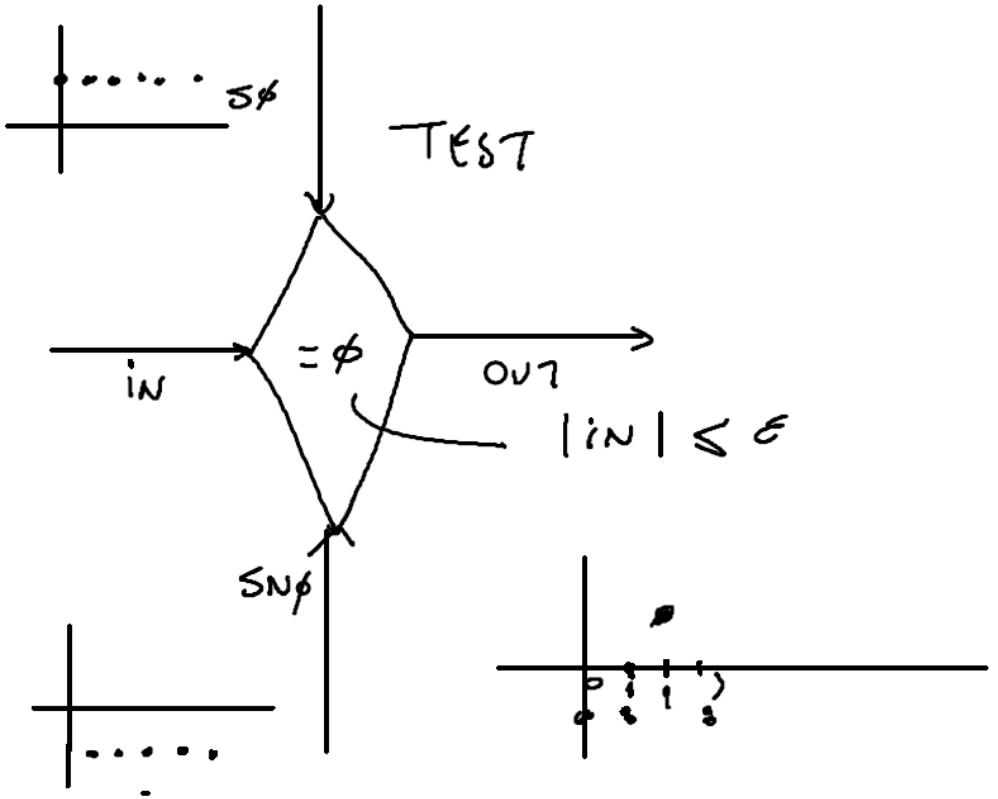


DYNAMIC STRUCTURE





multiplexok



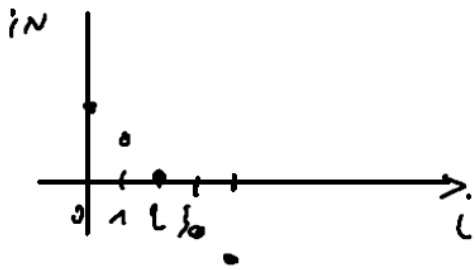
$\mathbb{N}, \mathbb{Z}$ $\quad i_1 == i_2$

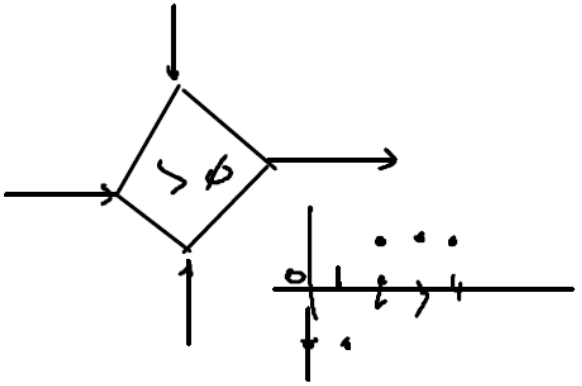
init

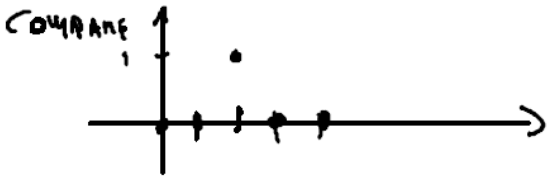
$\mathbb{R}$ $\quad x_1 == x_2$

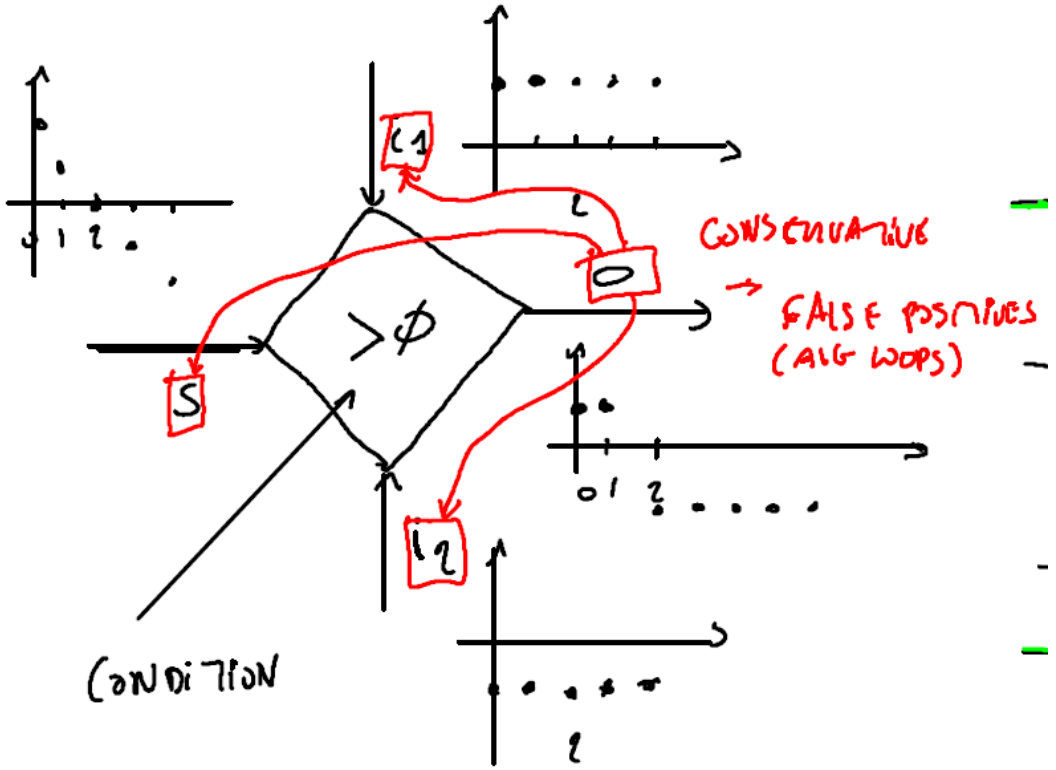
float, double

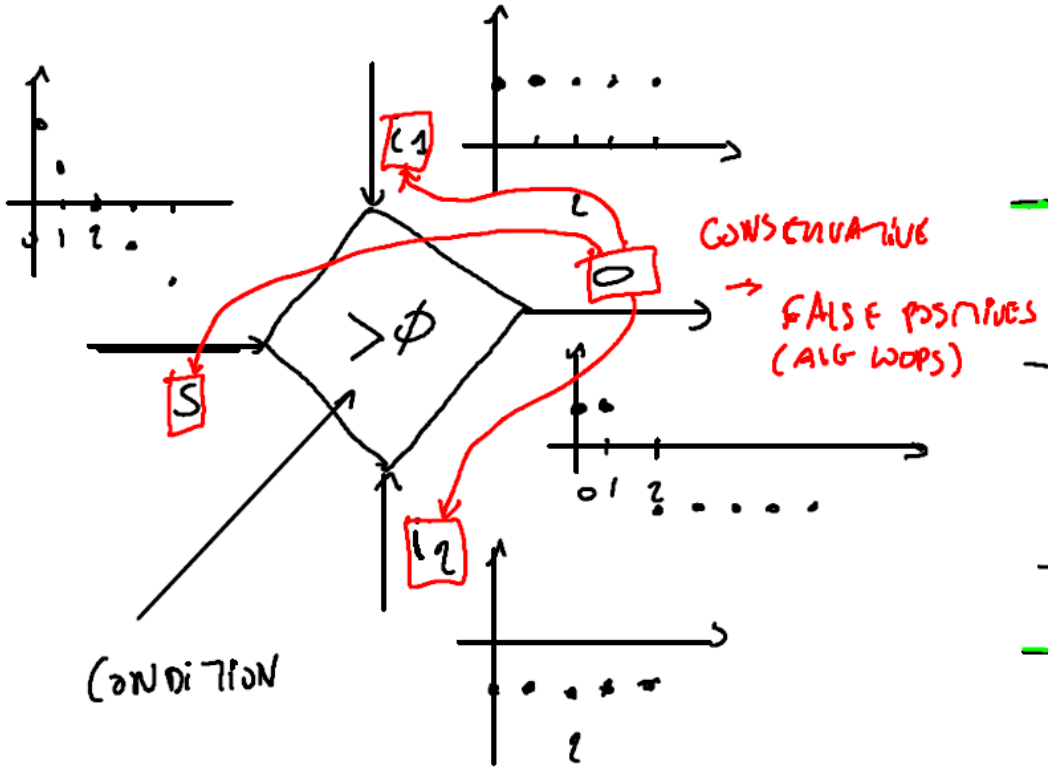
$|x_1 - x_2| \leq \varepsilon$

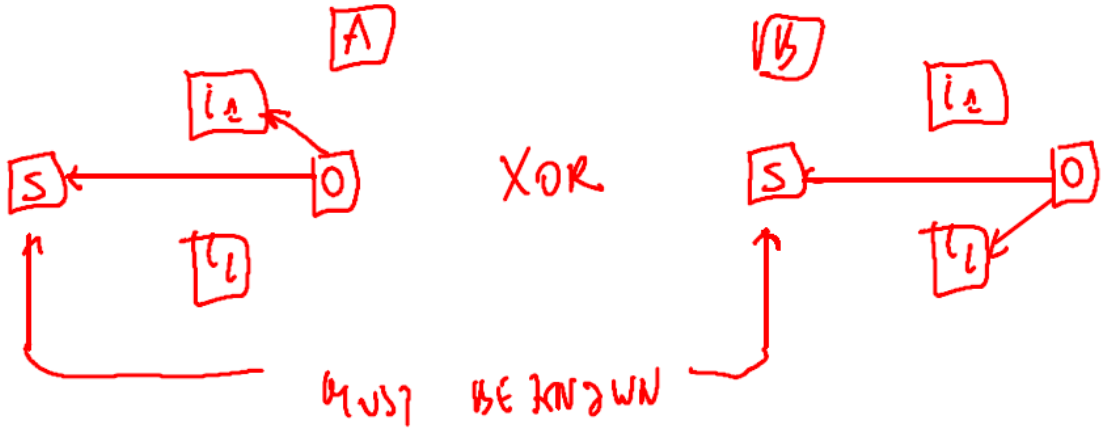







1
S
>Φ
0
1
CONSERVATIVE
→
FALSE POSITIVES
(ALG WOPS)
0 1 2
1 2
CONDITION
2

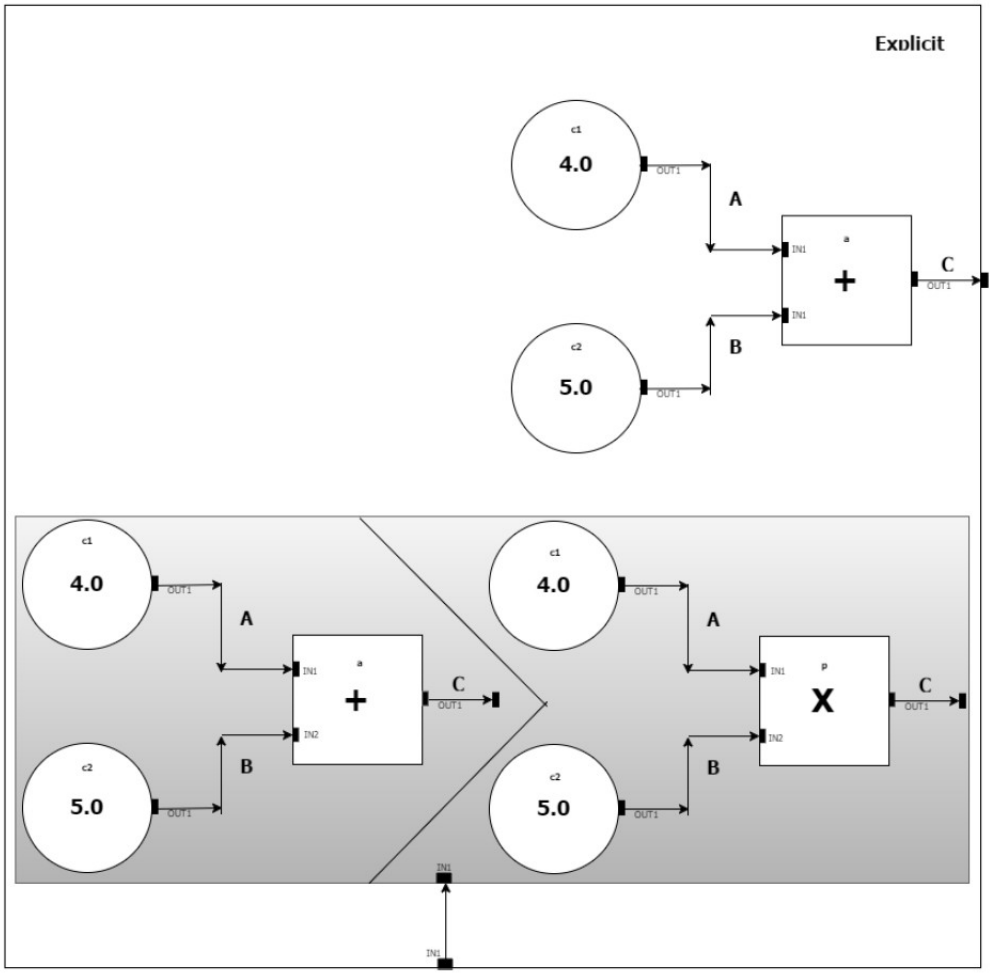


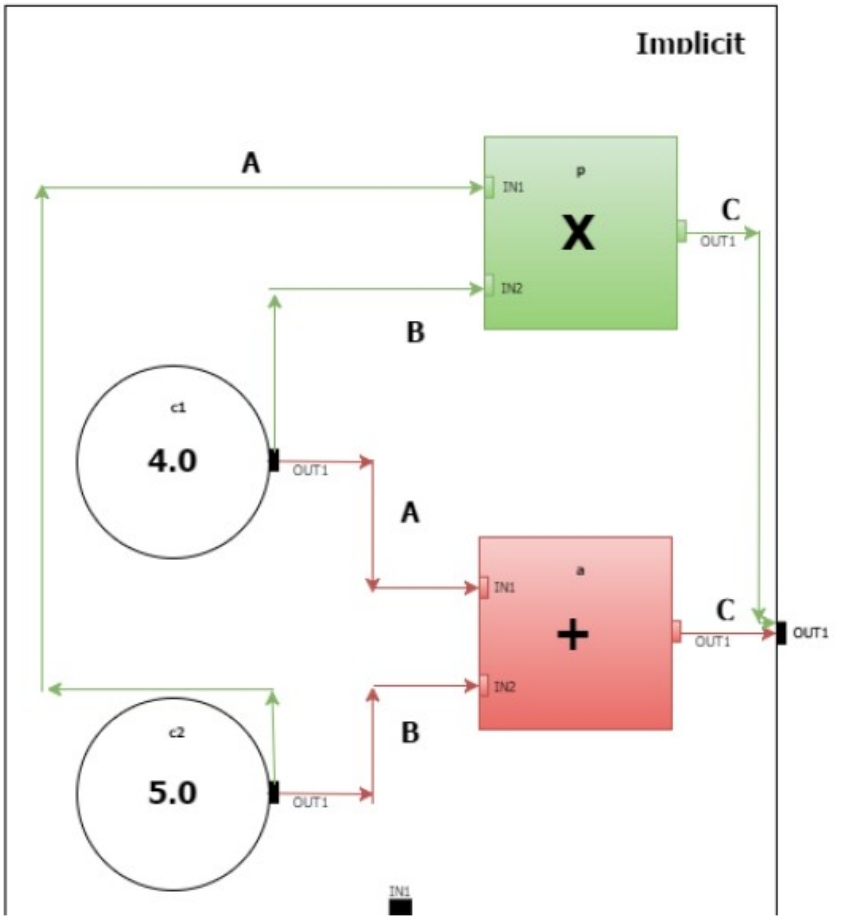


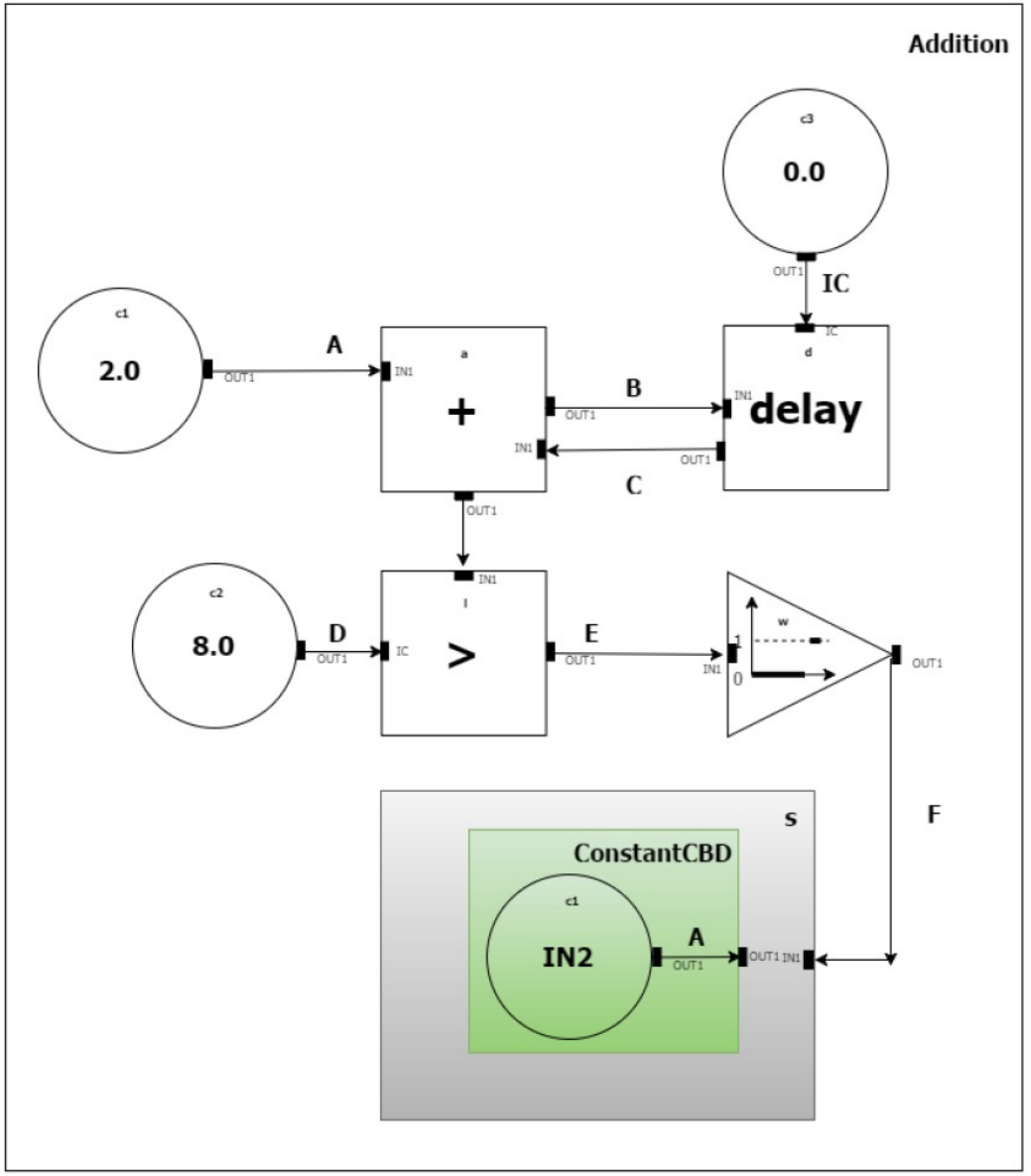
1. BUILD DEP. GRAPH UPTO [S]
2. PARTIAL SCAN/LOOP DET UPTO [S]
3. COMPLETE DEP GRAPH WITH [A] OR [B]
4. FULL SCAN/LOOP DETECT

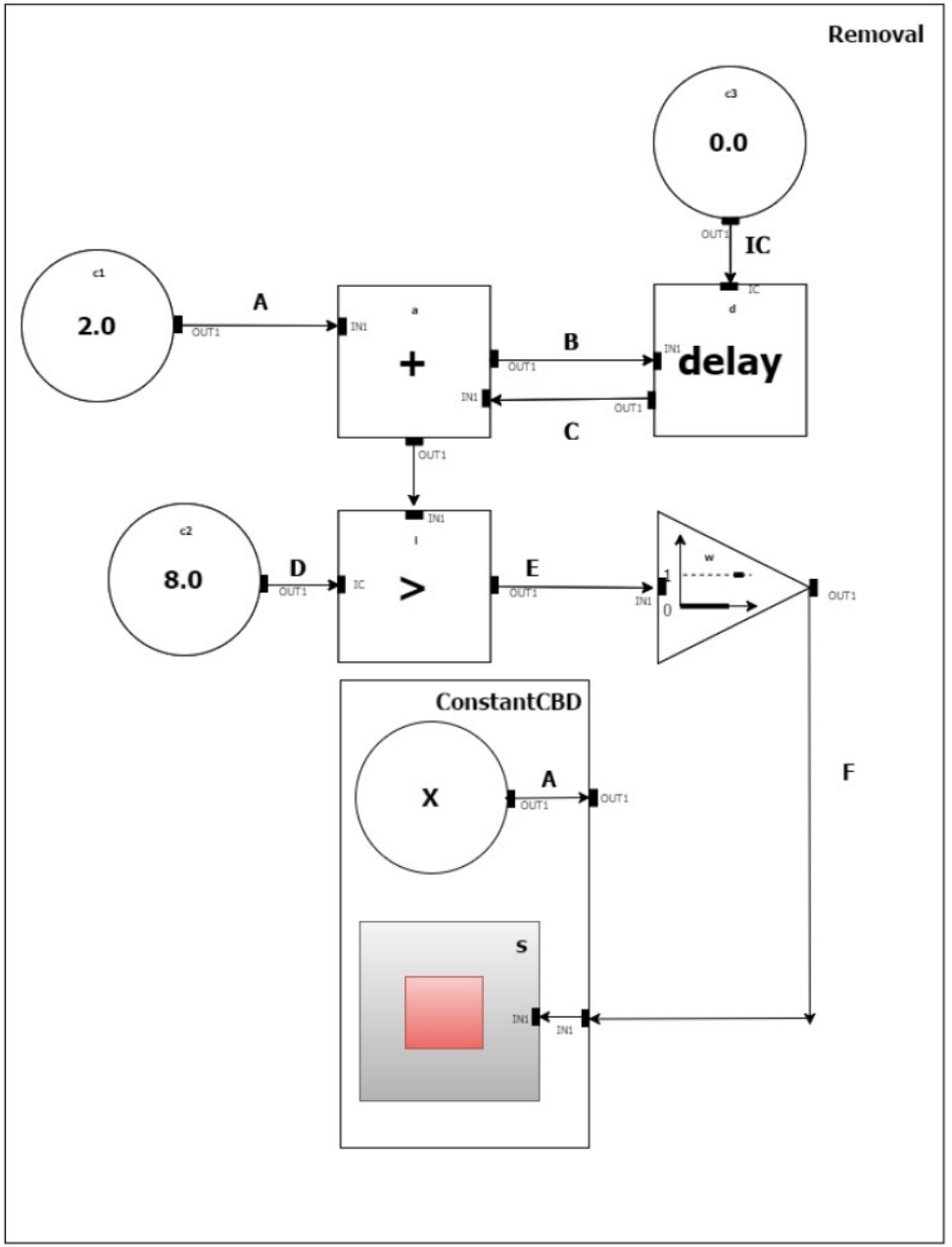
MULTIPLE TEST BLOCK

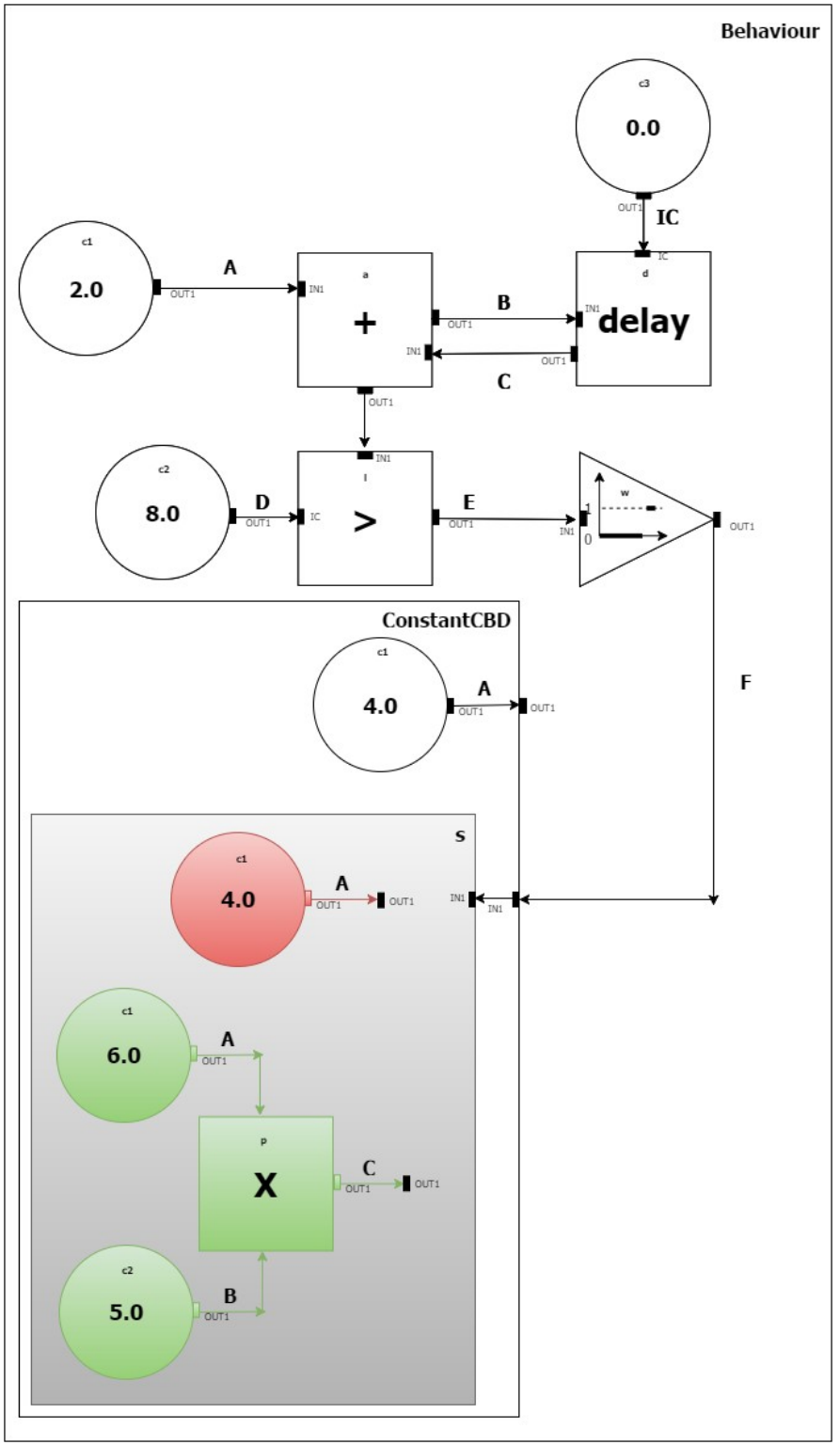
FIXED POINT

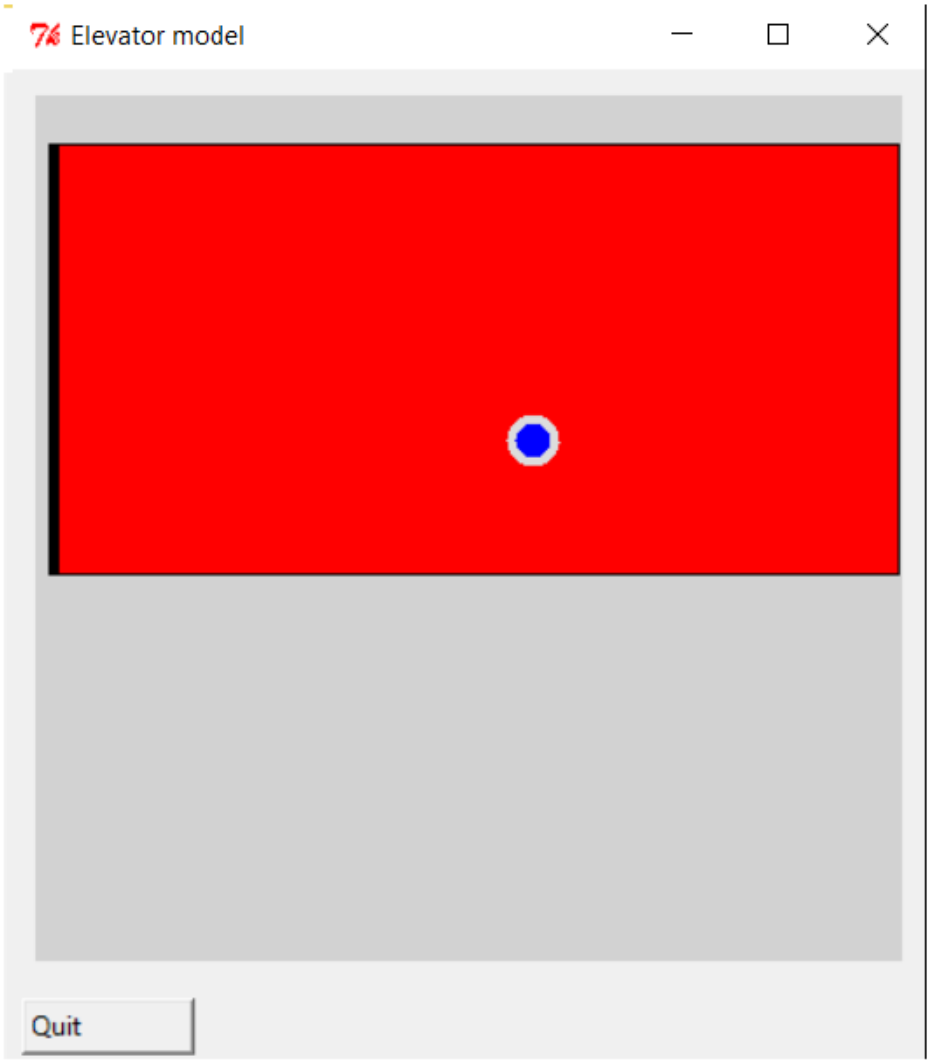

Explicit
c1
4.0
OUT1
A
a
IN1
+
C
OUT1
c2
5.0
B
IN1
IN2
P
X
IN1

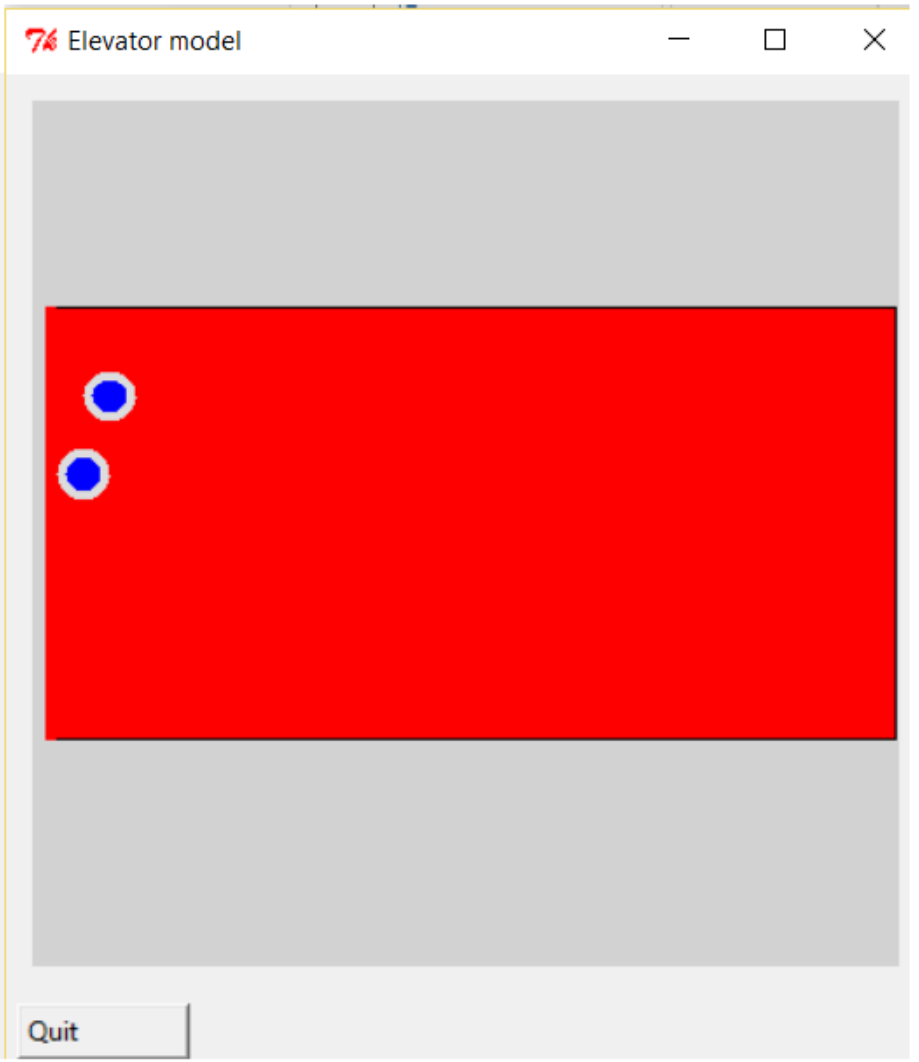

Implicit
A
P
IN1
X
C
OUT1
IN2
B
c1
4.0
OUT1
A
a
IN1
+
C
OUT1
OUT1
IN2
c2
5.0
B
OUT1
IN1

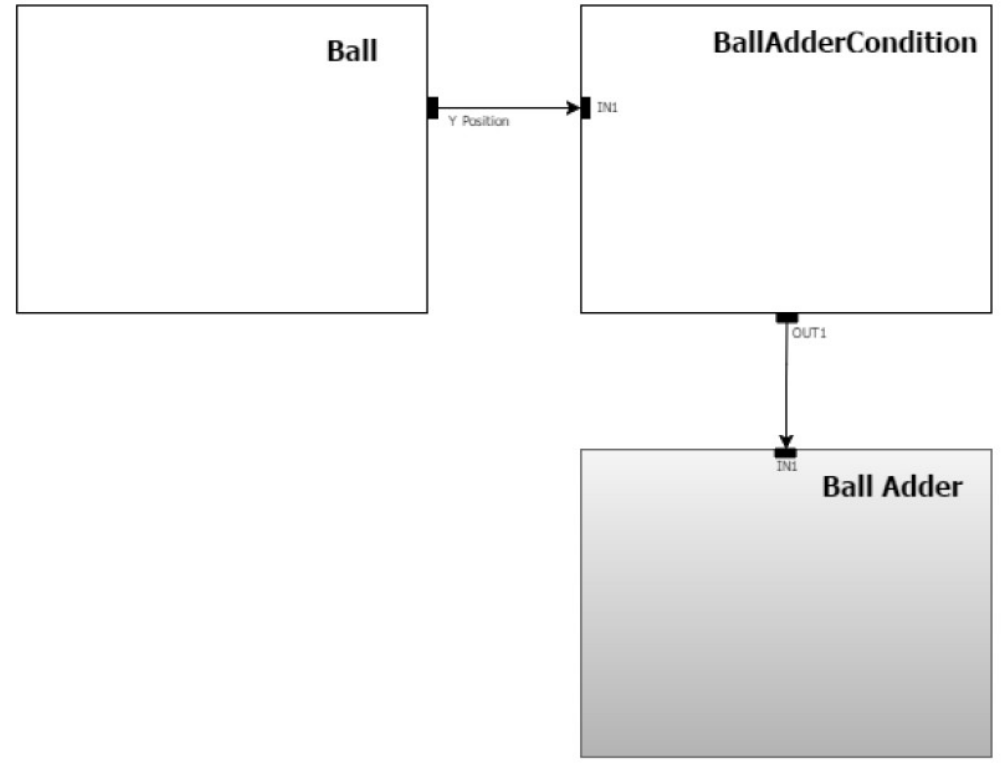

Addition
c3
0.0
OUT1
IC
c1
2.0
A
a
+
B
d
delay
C
c2
8.0
D
I
>
E
w
F
s
ConstantCBD
IN2

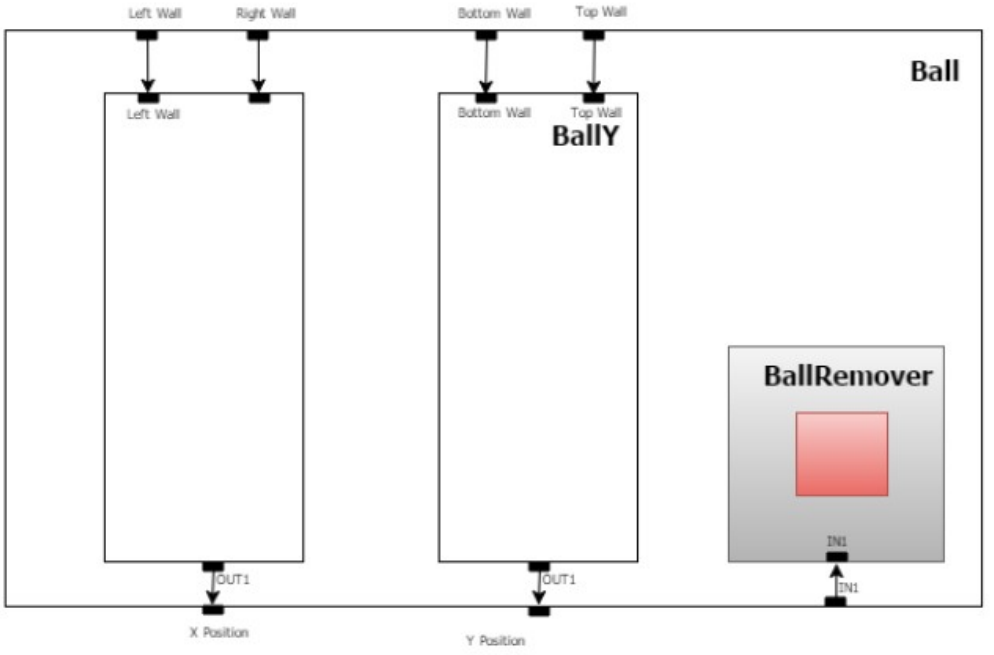

Removal
c3
0.0
IC
c1
2.0
A
a
+
B
d
delay
C
c2
8.0
D
I
>
E
w
F
ConstantCBD
X
A
s


Behaviour
c3
0.0
OUT1
IC
c1
2.0
A
OUT1
a
IN1
+
B
OUT1
IN1
d
IC
IN1
delay
OUT1
C
OUT1
c2
8.0
D
OUT1
IC
I
IN1
>
E
OUT1
w
1
0
IN1
OUT1
F
ConstantCBD
c1
4.0
A
OUT1
OUT1
S
c1
4.0
A
OUT1
OUT1
IN1
IN1
c1
6.0
A
OUT1
p
X
C
OUT1
OUT1
c2
5.0
B
OUT1


Elevator model
Quit


Elevator model
Quit


Ball
BallAdderCondition
IN1
Y Position
OUT1
IN1
Ball Adder


Left Wall
Right Wall
Bottom Wall
Top Wall
Ball
Left Wall
Bottom Wall
Top Wall
BallY
BallRemover
IN1
OUT1
OUT1
IN1
X Position
Y Position

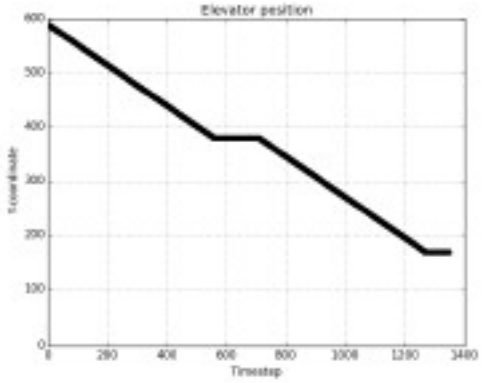


Figure 4.10: Y-position of the elevator

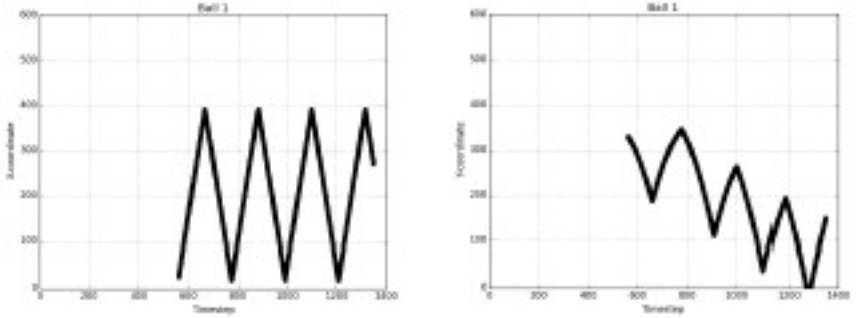


Figure 4.11: Position of ball 1

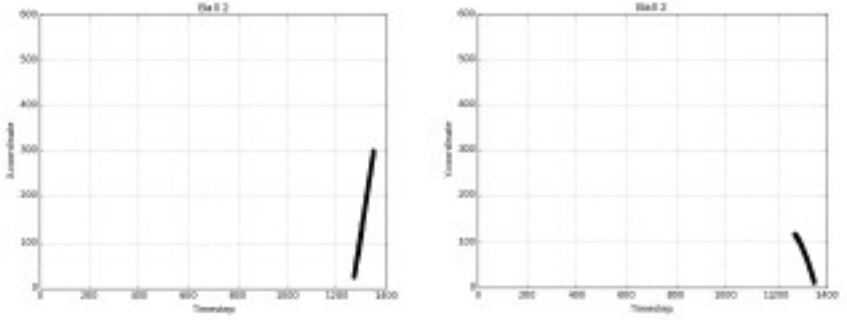


Figure 4.12: Position of ball 2