Universidade de Brasília

Algebraic Modeling of Continuous Time Systems[1]

José Edil G. de Medeiros

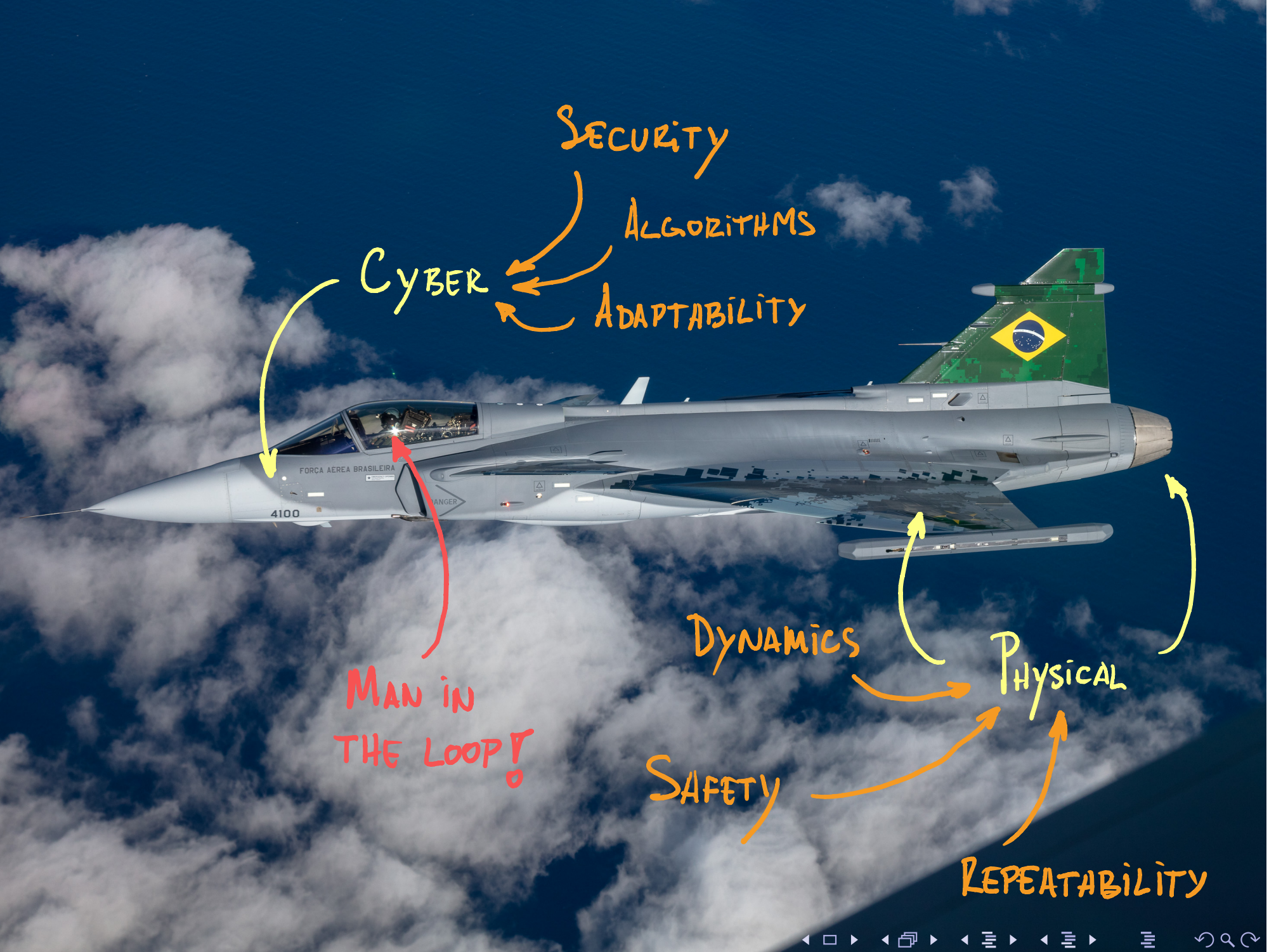
Electrical Engineering Department
University of Brasília, Brazil
j.edil@ene.unb.br

October, 2019

[1] Some slides kindly provided by Prof. Ingo Sander from KTH.


FORÇA AÉREA BRASILEIRA
4100
DANGER


Cyber
FORÇA AÉREA BRASILEIRA
4100
DANGER
Physical

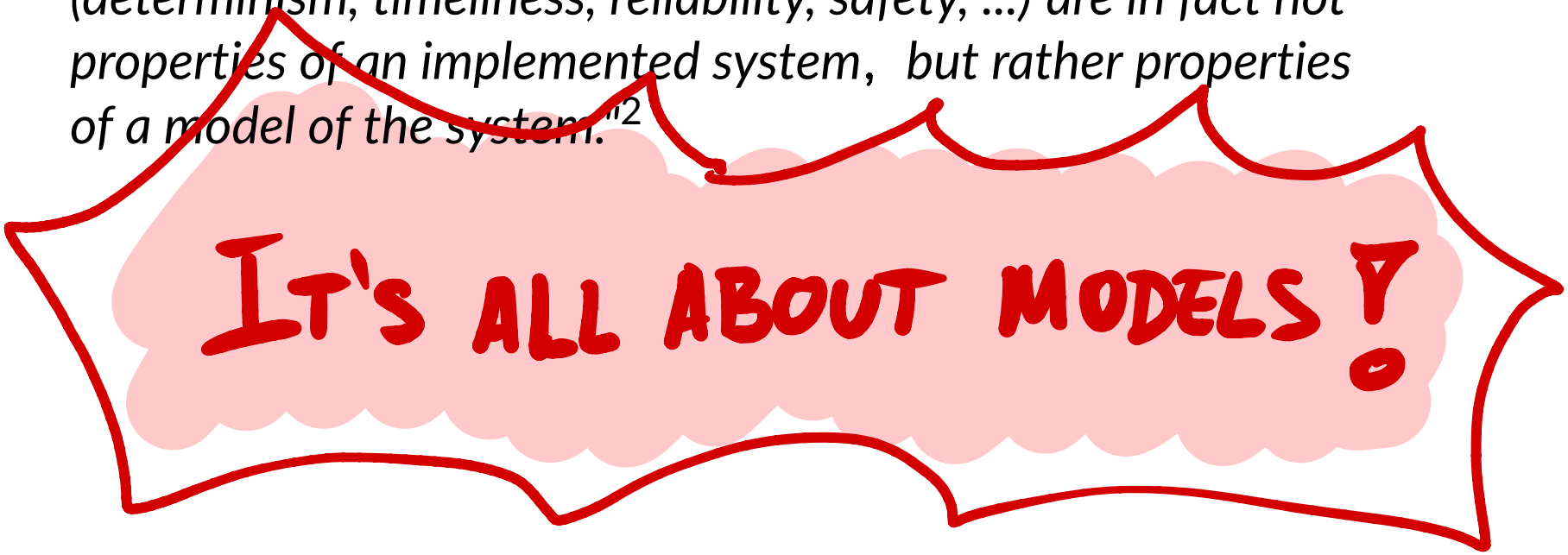

SECURITY
ALGORITHMS
CYBER
ADAPTABILITY
FORÇA AÉREA BRASILEIRA
4100
MAN IN THE LOOP!
DYNAMICS
PHYSICAL
SAFETY
REPEATABILITY


"You will never strike
oil by drilling through
the map"
Solomon Golomb

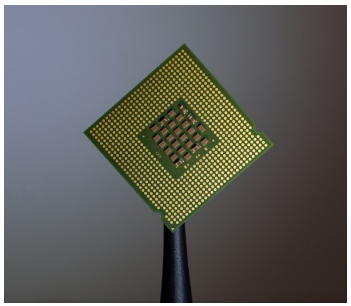
"Many predictive properties that we assert about systems (determinism, timeliness, reliability, safety, ...) are in fact not properties of an implemented system, but rather properties of a model of the system."[2]

[2] Prof. Hermann Kopetz apud Cyber-Physical Systems: A Rehash or a New Intellectual Challenge?; Edward Lee; Design Automation Conference (DAC); 2013.

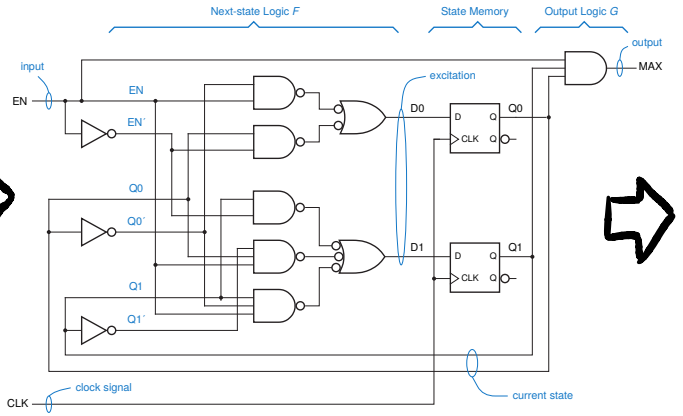
"Many predictive properties that we assert about systems (determinism, timeliness, reliability, safety, ...) are in fact not properties of an implemented system, but rather properties of a model of the system."[2]



[2] Prof. Hermann Kopetz apud Cyber-Physical Systems: A Rehash or a New Intellectual Challenge?; Edward Lee; Design Automation Conference (DAC); 2013.



ASIC

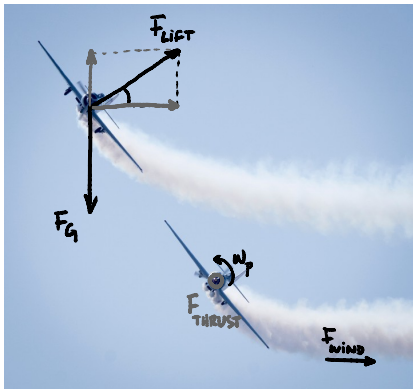


SYNCHRONOUS DIGITAL LOGIC

SINGLE-THREADED IMPERATIVE PROGRAMS

```c
#include <msp430.h>

int main(void) {
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    //Configure output ports
    P1DIR |= BIT0;              // P1.0 = LED

    //Configure clock system
    UCSCTL0 = 0x0000;           // Set lowest possible DCOx, MODx
                                // These are controlled by FLL
    UCSCTL1 = DCORSEL_5;        // Select DCO range to 16MHz
    UCSCTL2 |= 243;             // (N+1) * 32.768kHz = 16MHz
    UCSCTL3 = SELREF_2;         // Set DCO FLL reference = REFO
    UCSCTL4 = SELA_0 | SELS_4 | SELM_3 ;    // Set ACLK = XT1,
                                // SMCLK = DCOCLKDIV, MCLK = DCOCLK
    UCSCTL5 = DIVPA_0 | DIVA_0 | DIVS_3 | DIVM_0;   // Set SMCLK divider
    __delay_cycles(250000);

    //Configure timer A0
    TA0CTL = TASSEL_1 | MC_1 | TACLR;   // SMCLK = 32768Hz, Up Mode
    TA0CCR0 = 32767;                    // 1 second period

    while(1){
        P1OUT ^= BIT0;

        while((TA0CTL & TAIFG) == 0){
        }
        TA0CTL &= ~(TAIFG);
    }

    return 0;
}
```

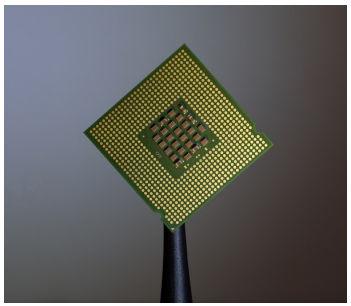
PHYSICAL SYSTEM



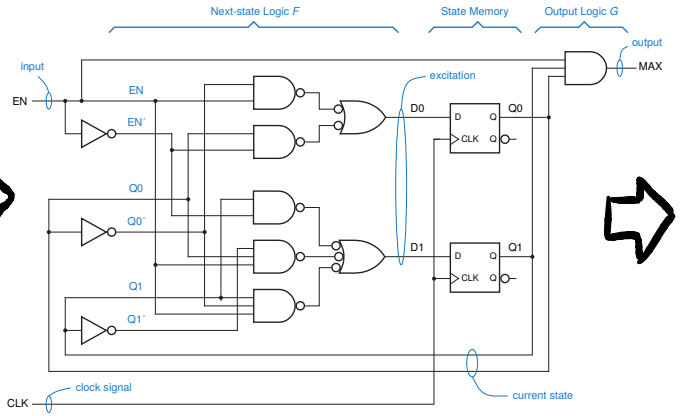
$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)\,dt + K_p T_d \frac{de(t)}{dt}$$

DIFFERENTIAL EQUATIONS

MODEL FOR CIRCUITS



ASIC



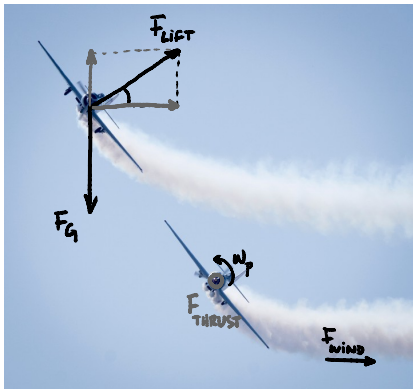
SYNCHRONOUS DIGITAL LOGIC

SINGLE-THREADED IMPERATIVE PROGRAMS

```c
#include <msp430.h>

int main(void) {
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    //Configure output ports
    P1DIR |= BIT0;              // P1.0 = LED

    //Configure clock system
    UCSCTL0 = 0x0000;           // Set lowest possible DCOx, MODx
                                // These are controlled by FLL
    UCSCTL1 = DCORSEL_5;        // Select DCO range to 16MHz
    UCSCTL2 |= 243;             // (N+1) * 32.768kHz = 16MHz
    UCSCTL3 = SELREF_2;         // Set DCO FLL reference = REFO
    UCSCTL4 = SELA_0 | SELS_4 | SELM_3 ;    // Set ACLK = XT1,
                                // SMCLK = DCOCLKDIV, MCLK = DCOCLK
    UCSCTL5 = DIVPA_0 | DIVA_0 | DIVS_3 | DIVM_0;   // Set SMCLK divider
    __delay_cycles(250000);

    //Configure timer A0
    TA0CTL = TASSEL_1 | MC_1 | TACLR;   // SMCLK = 32768Hz, Up Mode
    TA0CCR0 = 32767;                    // 1 second period

    while(1){
        P1OUT ^= BIT0;

        while((TA0CTL & TAIFG) == 0){
        }
        TA0CTL &= ~(TAIFG);
    }

    return 0;
}
```

MODEL FOR ALGORITHMS/PROCESSES

PHYSICAL SYSTEM



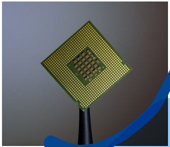
$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)\,dt + K_p T_d \frac{de(t)}{dt}$$

DIFFERENTIAL EQUATIONS

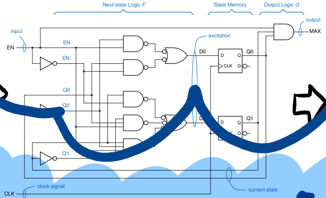
MODEL FOR DYNAMICS

MODEL FOR CIRCUITS

SINGLE-THREADED IMPERATIVE PROGRAMS



ASIC

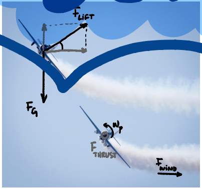


SYNCHRONOUS DIGITAL

```
#include <msp430.h>

int main(void) {
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    //Configure output ports
    P1DIR |= BIT0;              //       = LED

    //Configure clock system
    UCSCTL0 = 0x0000;           // Set lo  t possible DCOx, MODx
                                // These a  ontrolled by FLL
    U   L1 = DCO     ;          // Select DCO   ge to 16MHz
    UCSCTL2 =    245;           // (N+1) * 32.7      = 16MHz
    UCSCTL3 = SELREF_2;         // Set DCO FLL refer
    UCSCTL4 = SELA_0 | SELS_4 | SELM_3 ;   // Set ACLK = XT1;
                                // SMCLK = DCOCLKDIV, MCLK = DCOCLK
    UCSCTL5 = DIVPA_0 | DIVA_0 | DIVS_3 | DIVM_0;   // Set SMCLK   ider
    __delay_cycles(250000);

    //Configure timer A0
    TA0CTL = TASSEL_1 | MC_1 | TA    // SMC   32768Hz, Up M  e
    TA0CCR0 = 32767                  // 1   d period

      le(
        P1   ^= BI

        wh   (TA0C     ) ==
        }
        TA0CTL &= ~(TAIFG);
    }

    return 0;
}
```

DETERMINISM

PHYSICAL SYSTEM

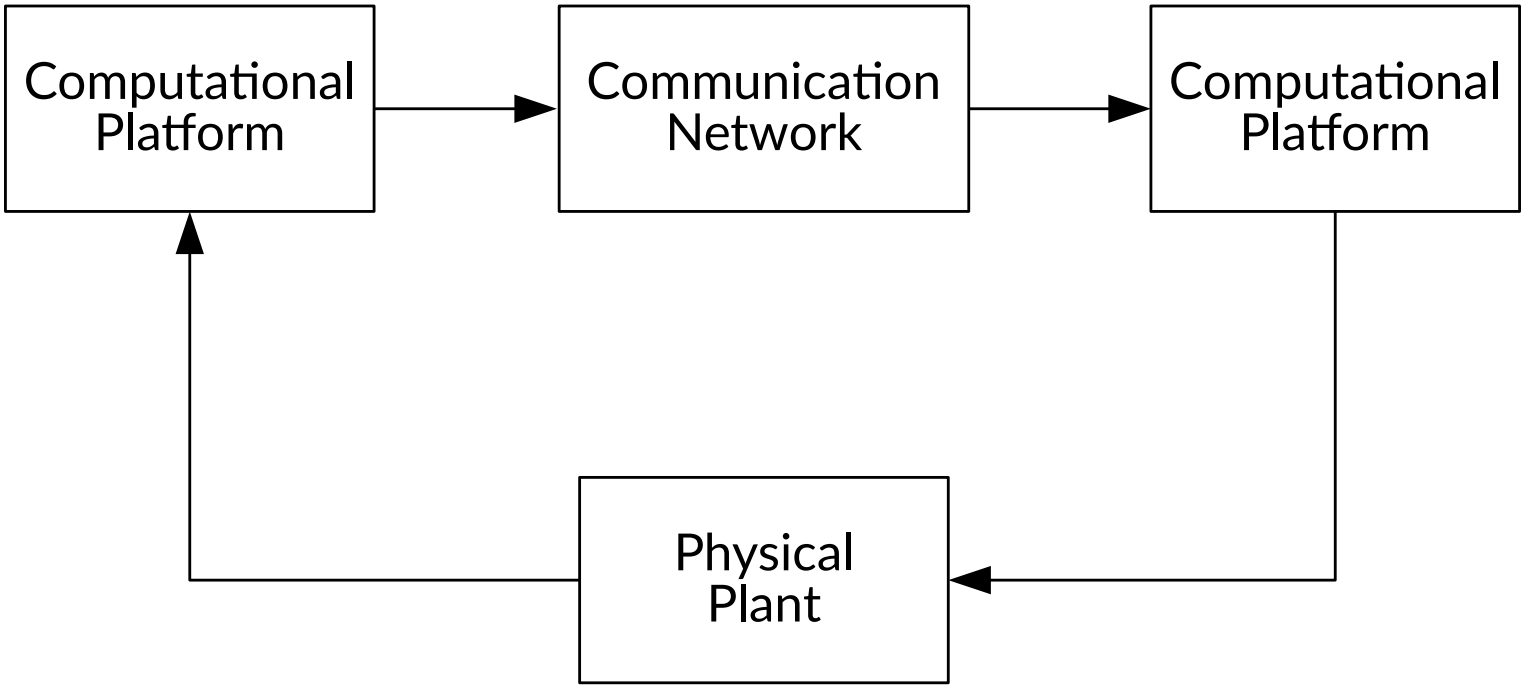


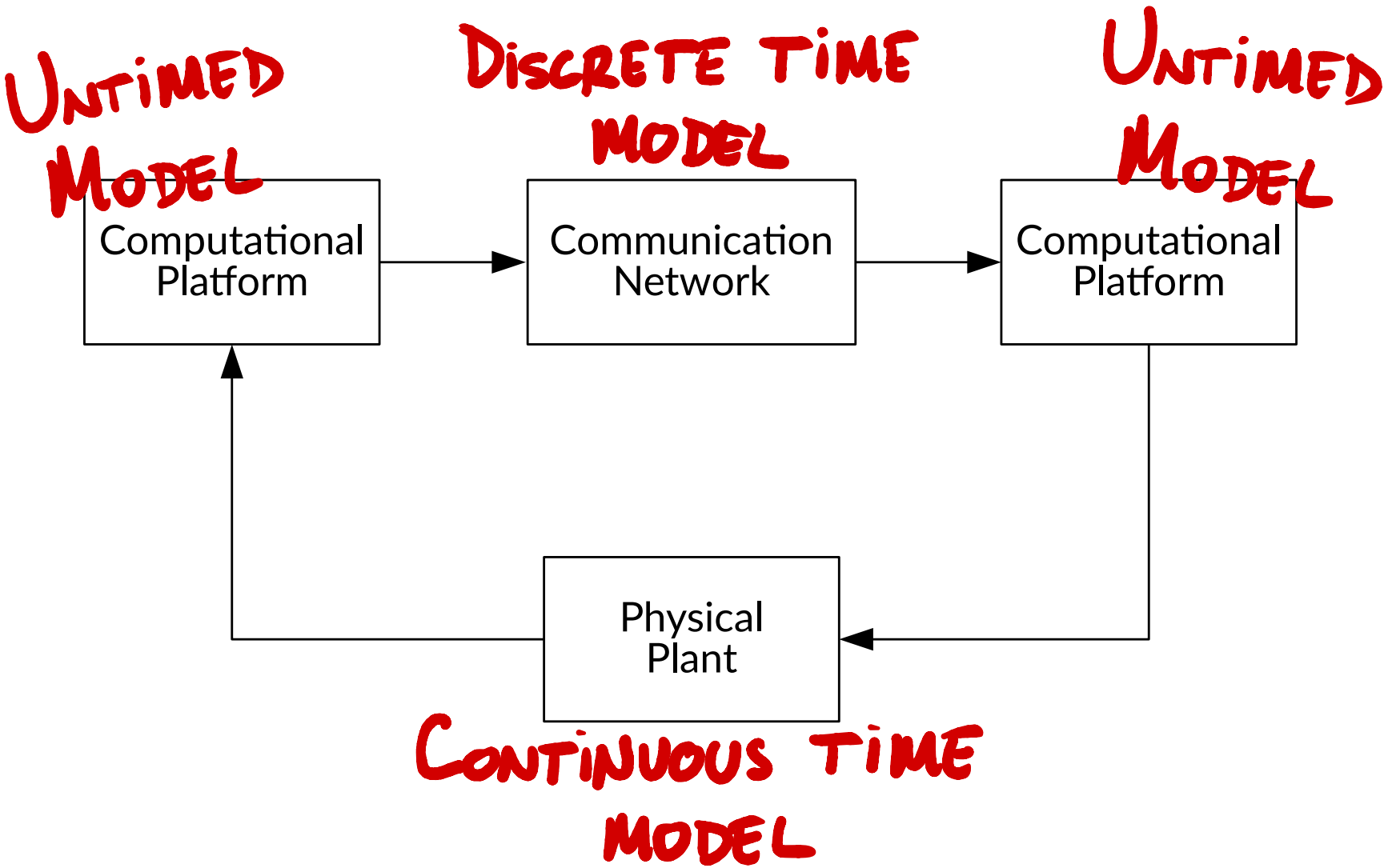
MODEL FOR ALGORITHMS / PROCESSES

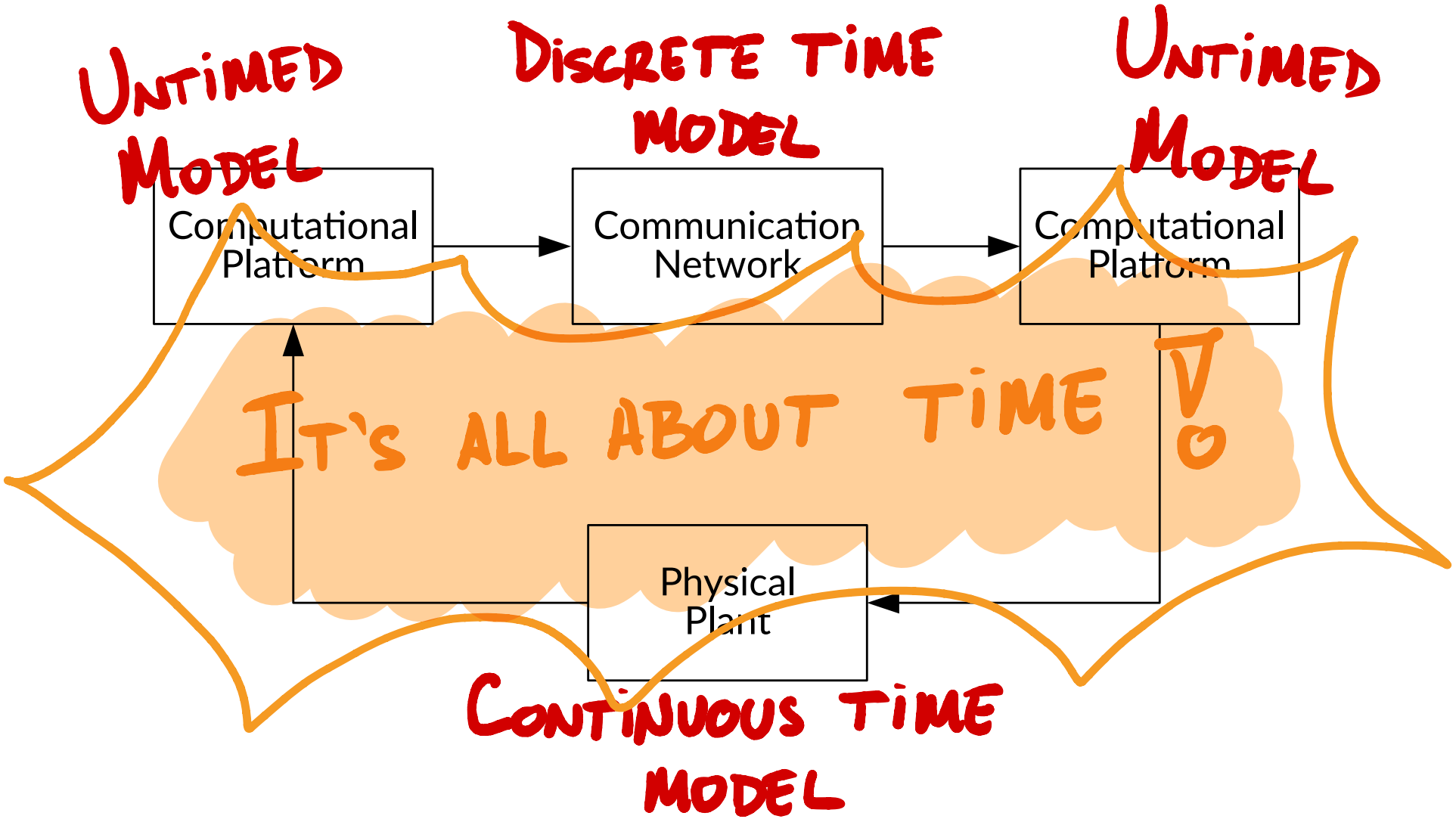
$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)\,dt + K_p T_d \frac{de(t)}{dt}$$

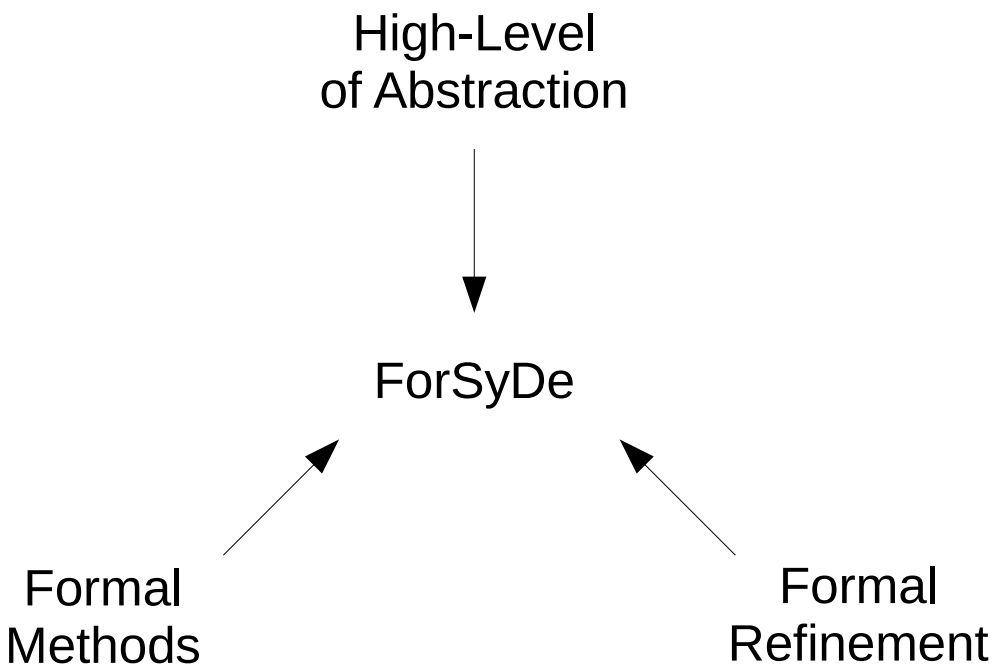
DIFFERENTIAL EQUATIONS

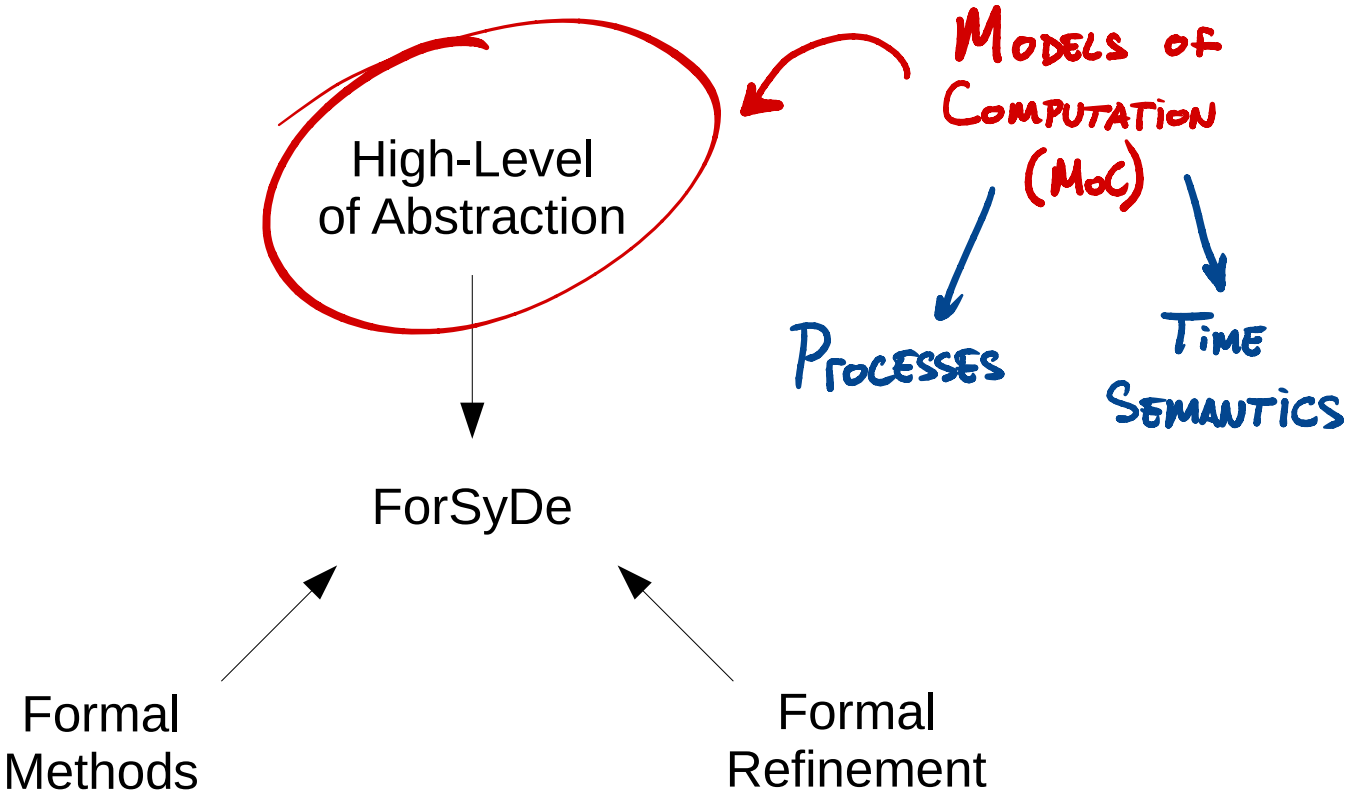
MODEL FOR DYNAMICS

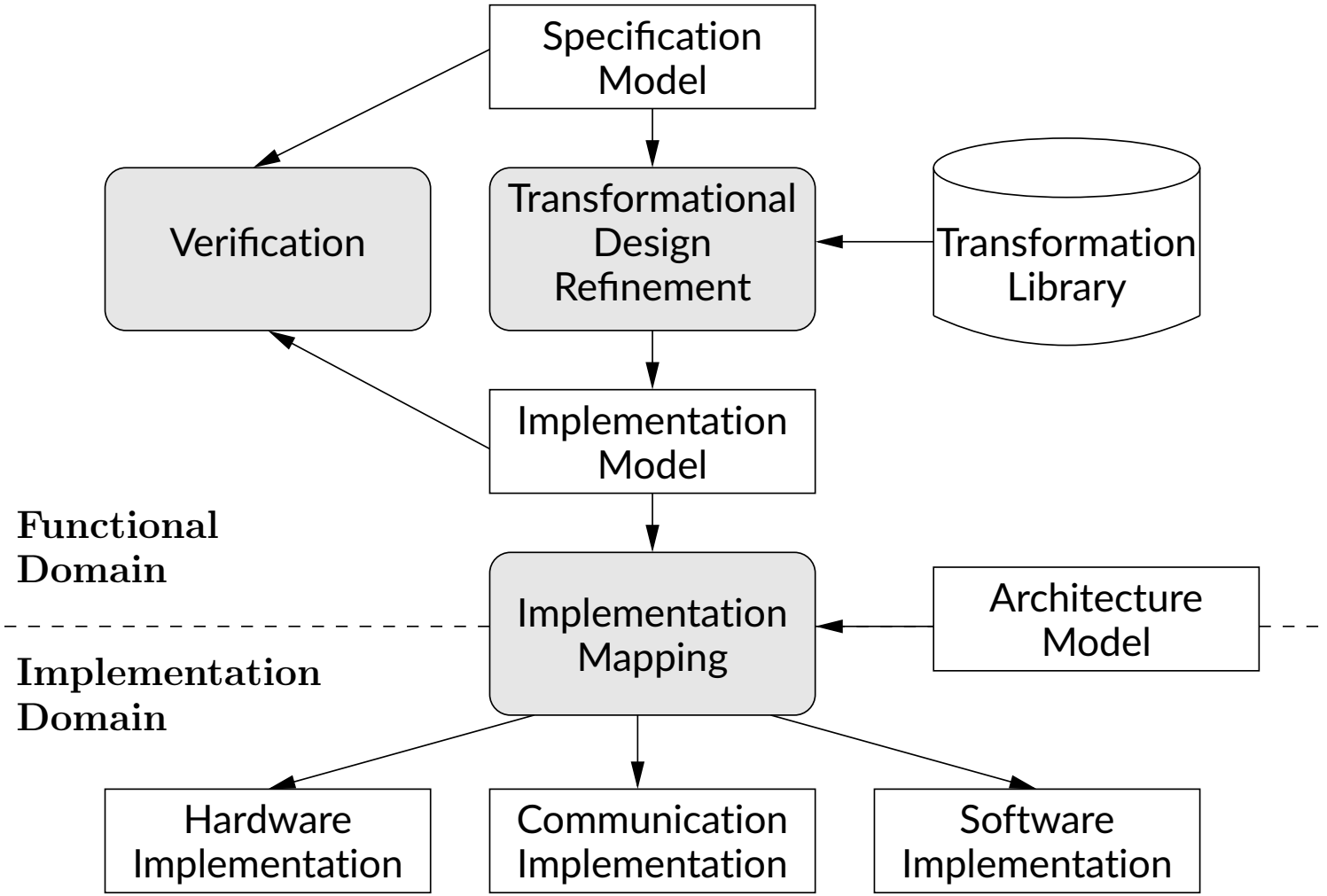

Computational Platform
Communication Network
Computational Platform
Physical Plant

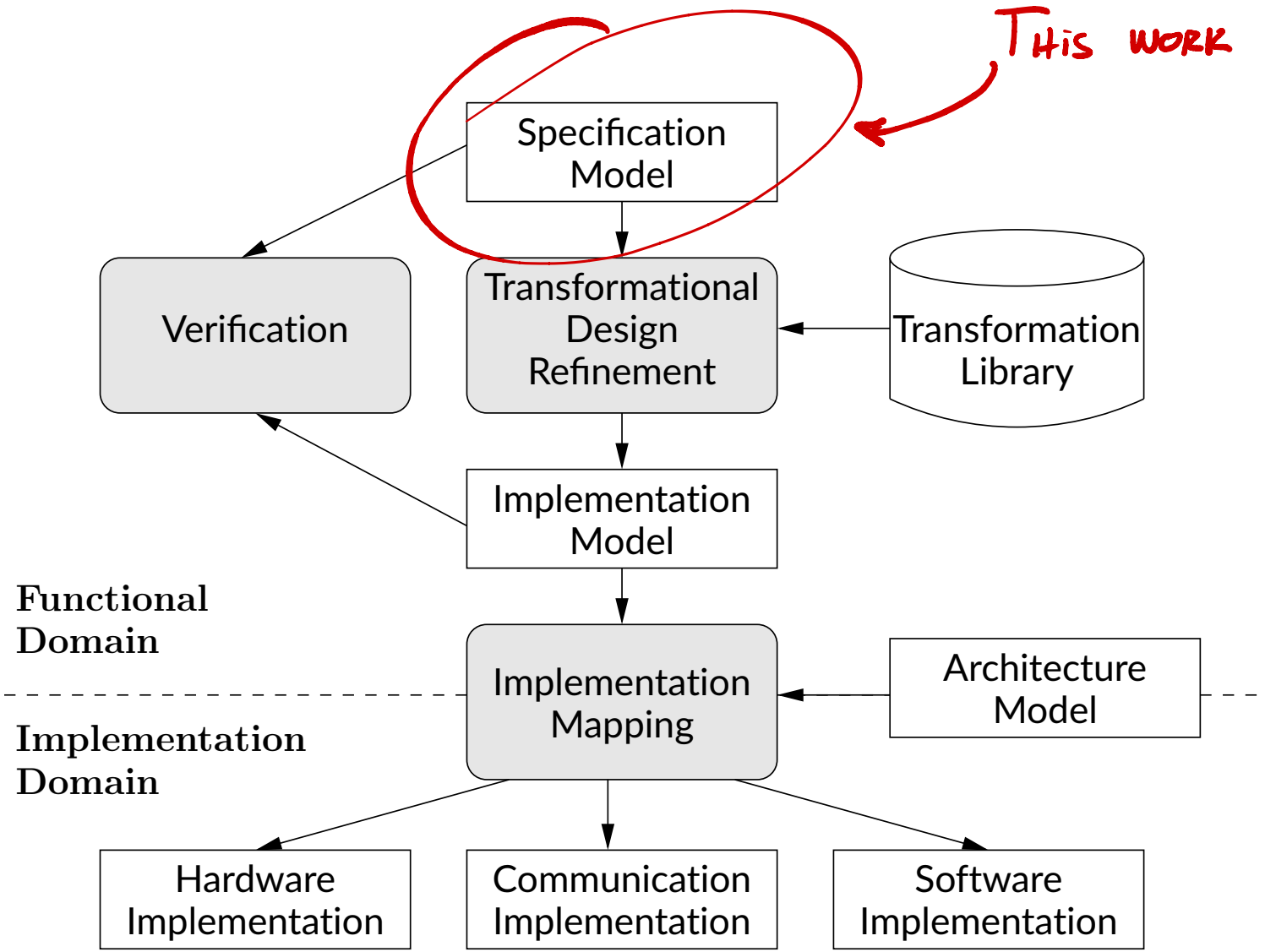

Untimed Model
Discrete Time Model
Untimed Model
Computational Platform
Communication Network
Computational Platform
Physical Plant
Continuous Time Model

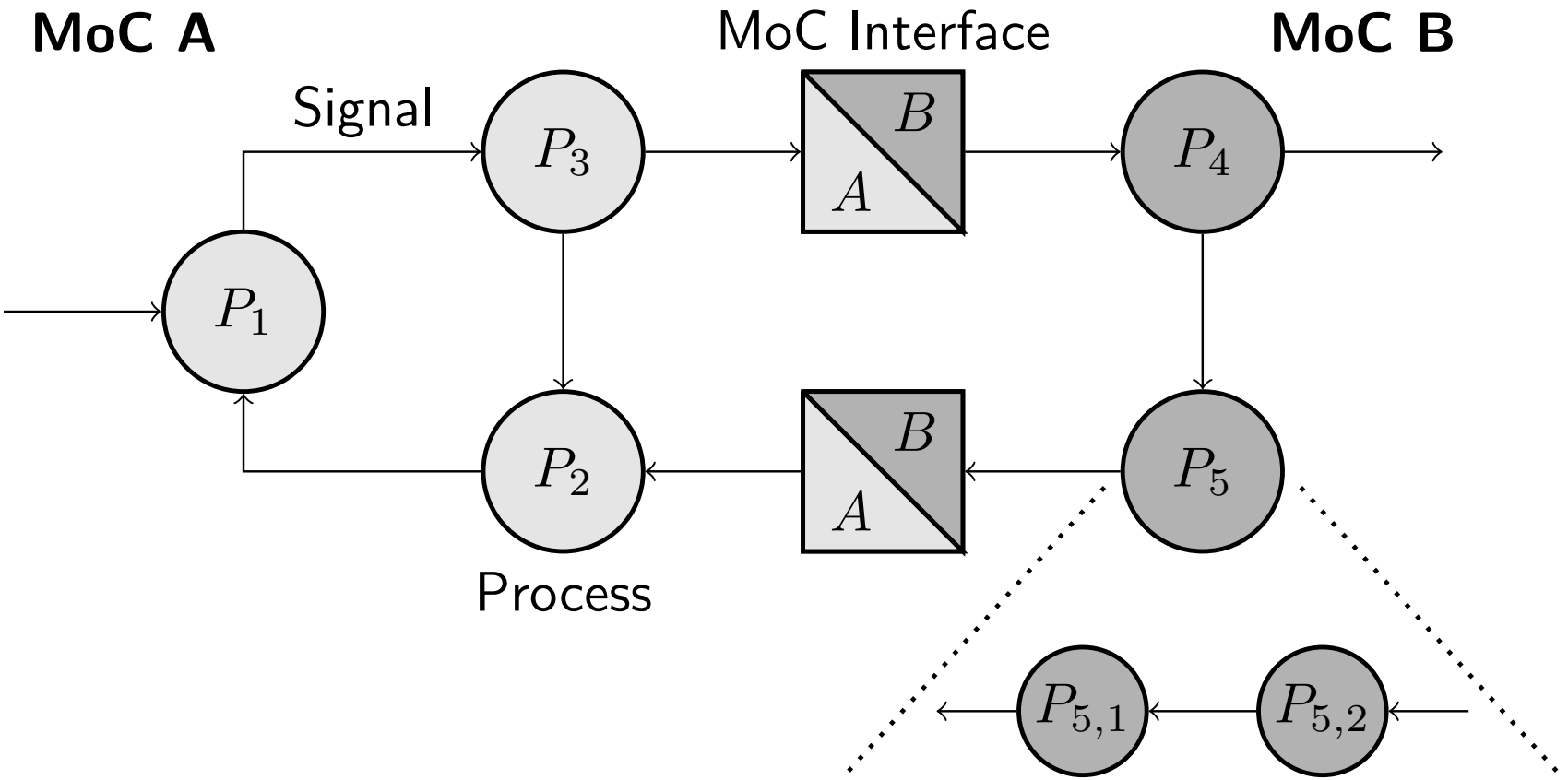

UNTIMED MODEL
DISCRETE TIME MODEL
UNTIMED MODEL
Computational Platform
Communication Network
Computational Platform
IT'S ALL ABOUT TIME !!
Physical Plant
CONTINUOUS TIME MODEL

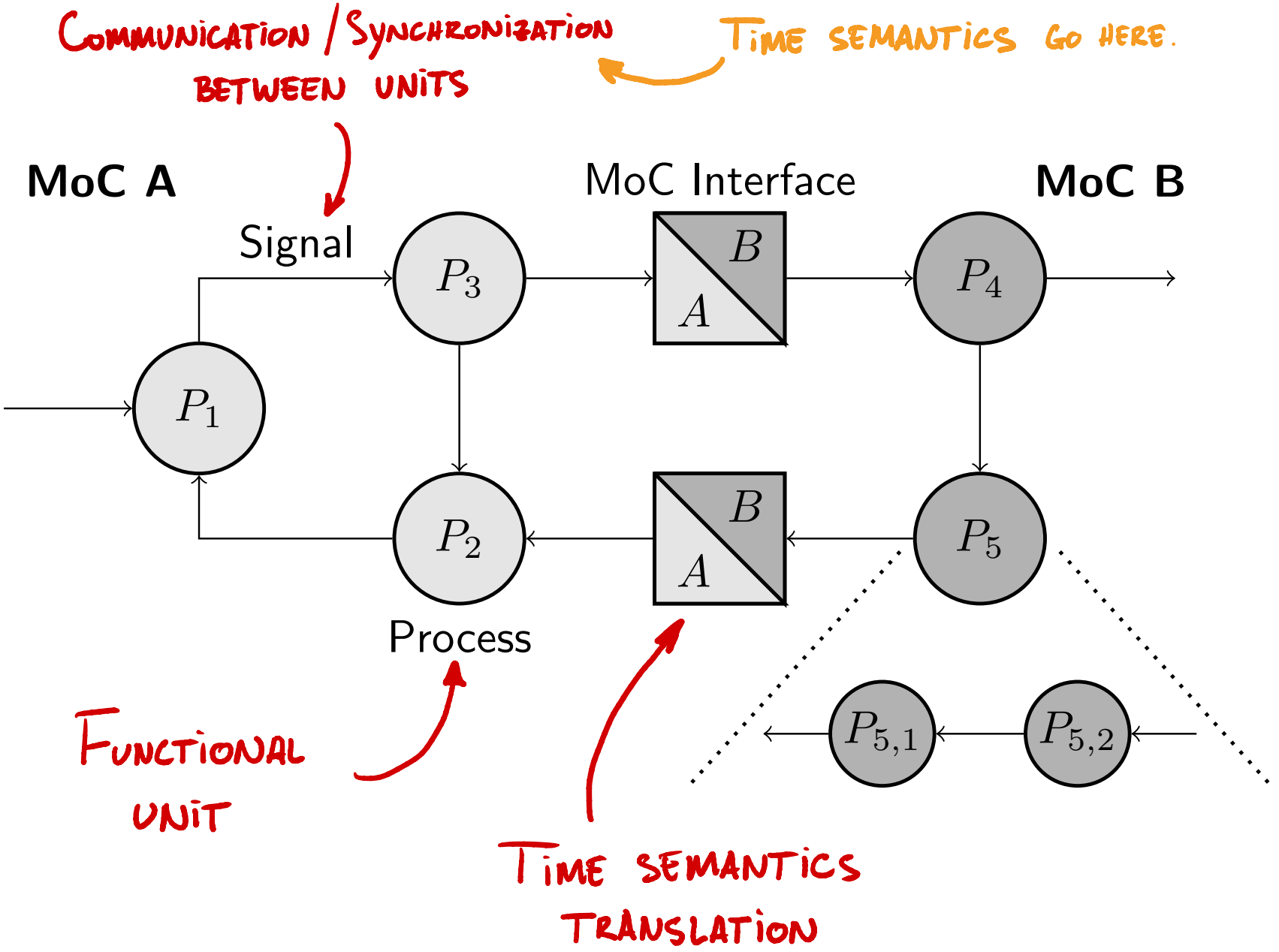

High-Level
of Abstraction
ForSyDe
Formal
Methods
Formal
Refinement

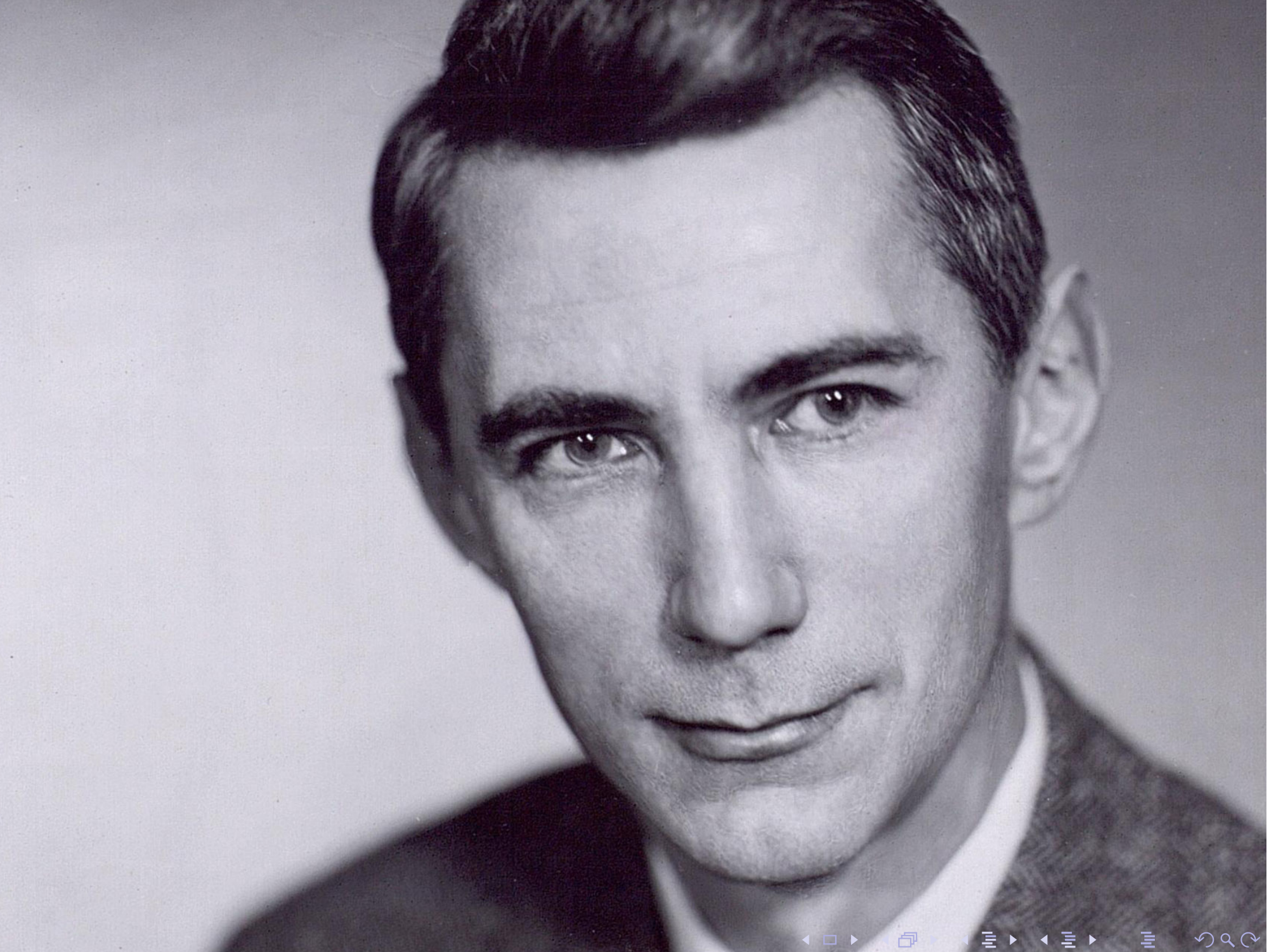

High-Level
of Abstraction
Models of
Computation
(MoC)
Processes
Time
Semantics
ForSyDe
Formal
Methods
Formal
Refinement

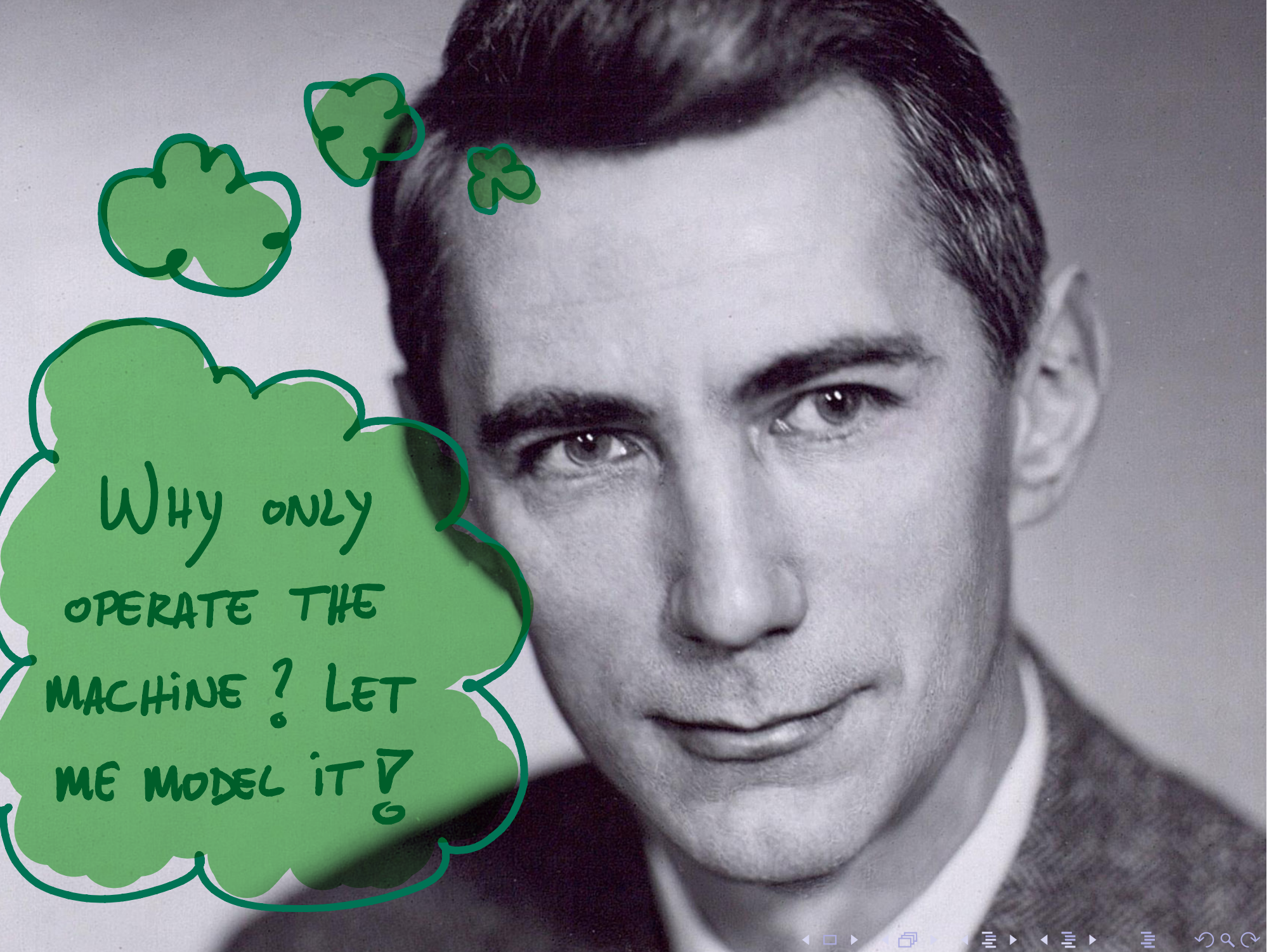

Specification Model
Verification
Transformational Design Refinement
Transformation Library
Implementation Model
Functional Domain
Implementation Mapping
Architecture Model
Implementation Domain
Hardware Implementation
Communication Implementation
Software Implementation

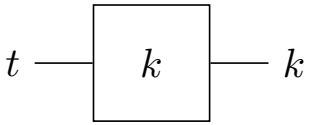

This work
Specification Model
Verification
Transformational Design Refinement
Transformation Library
Implementation Model
Functional Domain
Implementation Mapping
Architecture Model
Implementation Domain
Hardware Implementation
Communication Implementation
Software Implementation


MoC A
MoC Interface
MoC B
Signal
P_1
P_2
P_3
P_4
P_5
$P_{5,1}$
$P_{5,2}$
B
A
B
A
Process

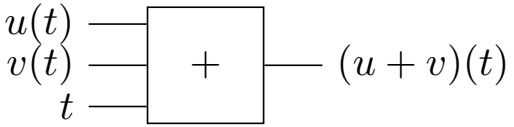

COMMUNICATION / SYNCHRONIZATION BETWEEN UNITS
TIME SEMANTICS GO HERE.
MoC A
MoC Interface
MoC B
Signal
P_1
P_2
P_3
P_4
P_5
$P_{5,1}$
$P_{5,2}$
B
A
B
A
Process
FUNCTIONAL UNIT
TIME SEMANTICS TRANSLATION



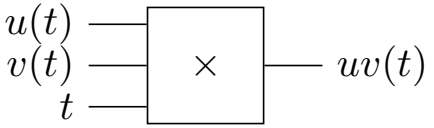

WHY ONLY OPERATE THE MACHINE? LET ME MODEL IT!



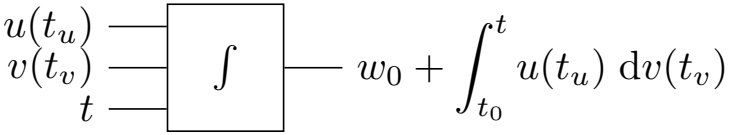
A constant unit



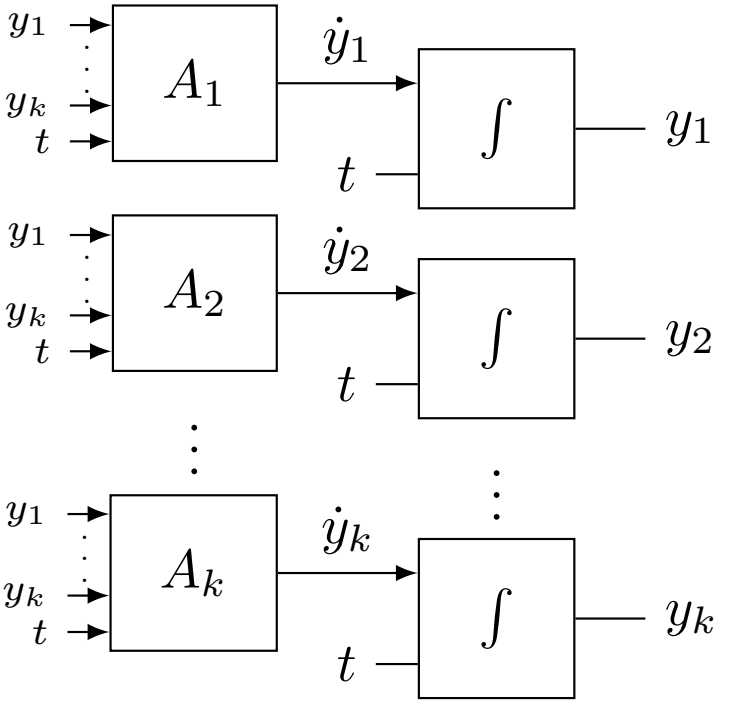
An adder unit



A multiplier unit



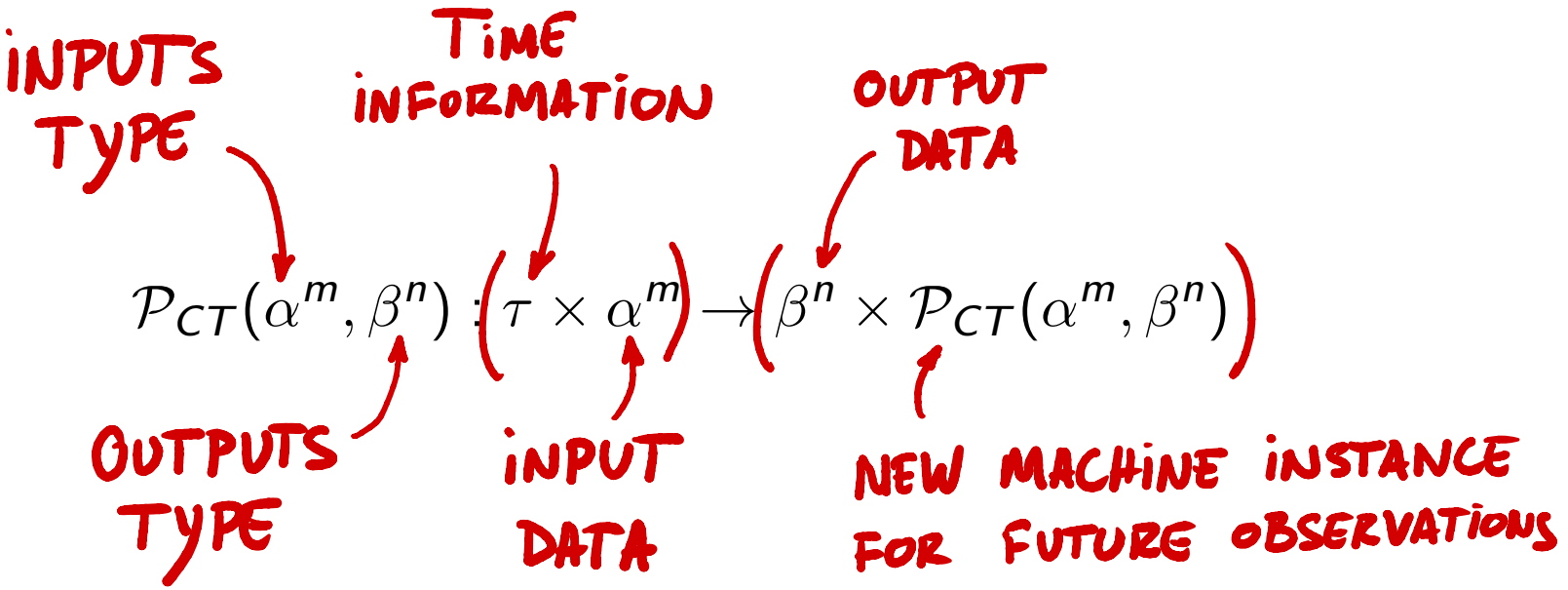
An integrator unit



COMPOSITION RULES

BASIC UNITS (AXIOMS) → GENERAL PURPOSE ANALOG COMPUTER (GPAC)

$$\mathcal{P}_{CT}(\alpha^m, \beta^n) : \tau \times \alpha^m \rightarrow \beta^n \times \mathcal{P}_{CT}(\alpha^m, \beta^n)$$


INPUTS TYPE
TIME INFORMATION
OUTPUT DATA
$\mathcal{P}_{CT}(\alpha^m, \beta^n) : (\tau \times \alpha^m) \rightarrow (\beta^n \times \mathcal{P}_{CT}(\alpha^m, \beta^n))$
OUTPUTS TYPE
INPUT DATA
NEW MACHINE INSTANCE FOR FUTURE OBSERVATIONS

"NEXT STATE" FUNCTION

$$\mathcal{P}_{CT}(\alpha^m, \beta^n) : \tau \times (\alpha^m \to \beta^n) \times \mathcal{P}_{CT}(\alpha^m, \beta^n)$$

$$const_{CT} : \mathcal{P}_{CT}(\alpha, \beta)$$
$$const_{CT}^{k} = (t, a) \mapsto (k, const_{CT}^{k})$$

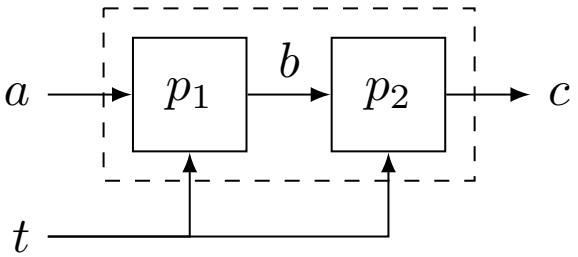
$$add_{CT} : \mathcal{P}_{CT}(\alpha \times \alpha, \alpha)$$
$$add_{CT} = (t, a, b) \mapsto (a + b, add_{CT})$$

$$mult_{CT} : \mathcal{P}_{CT}(\alpha \times \alpha, \alpha)$$
$$mult_{CT} = (t, a, b) \mapsto (ab, mult_{CT})$$

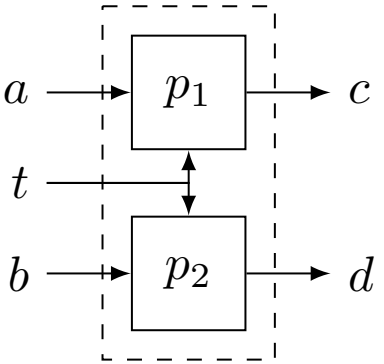
$$\triangleright : \mathcal{P}_{CT}(\alpha, \beta) \times \mathcal{P}_{CT}(\beta, \gamma) \rightarrow \mathcal{P}_{CT}(\alpha, \gamma)$$
$$p_1 \triangleright p_2 = (t, a) \mapsto (c, p_1' \triangleright p_2')$$

$$\|: \mathcal{P}_{CT}(\alpha, \beta) \times \mathcal{P}_{CT}(\gamma, \delta) \rightarrow \mathcal{P}_{CT}(\alpha \times \beta, \gamma \times \delta)$$
$$p_1 \parallel p_2 = (t, a) \mapsto (c, d, p_1' \parallel p_2')$$

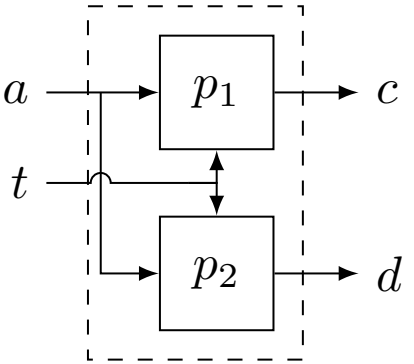
$$\star : \mathcal{P}_{CT}(\alpha, \beta) \times \mathcal{P}_{CT}(\alpha, \gamma) \rightarrow \mathcal{P}_{CT}(\alpha, \beta \times \gamma)$$
$$p_1 \star p_2 = (t, a) \mapsto (b, c, p_1' \star p_2')$$



$p_1 \triangleright p_2$



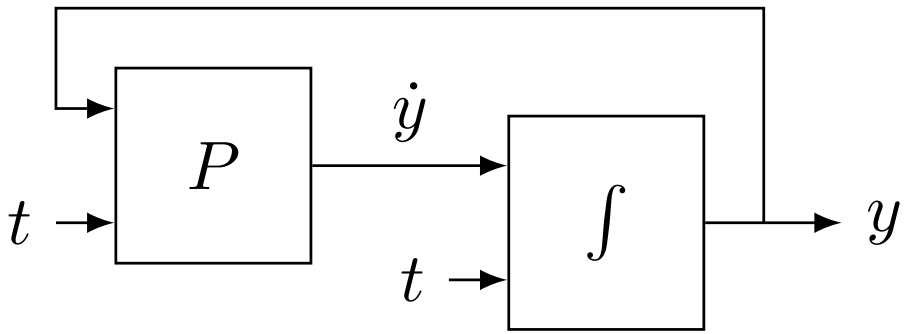
$p_1 \parallel p_2$

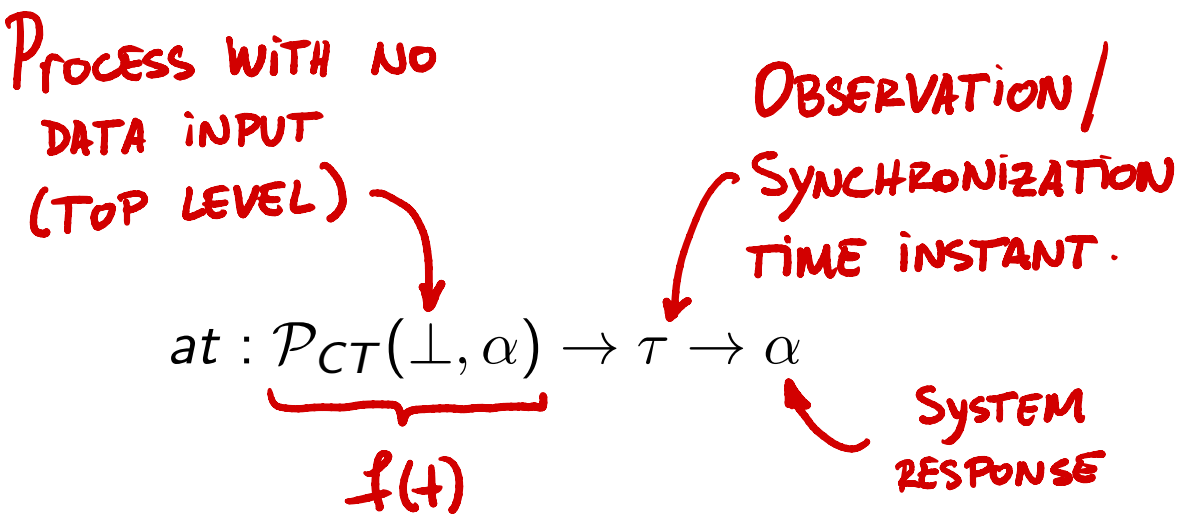


$p_1 \star p_2$

$$\int : \mathcal{P}_{CT}(\alpha \times \beta, \gamma) \rightarrow \mathcal{P}_{CT}(\alpha, \gamma)$$

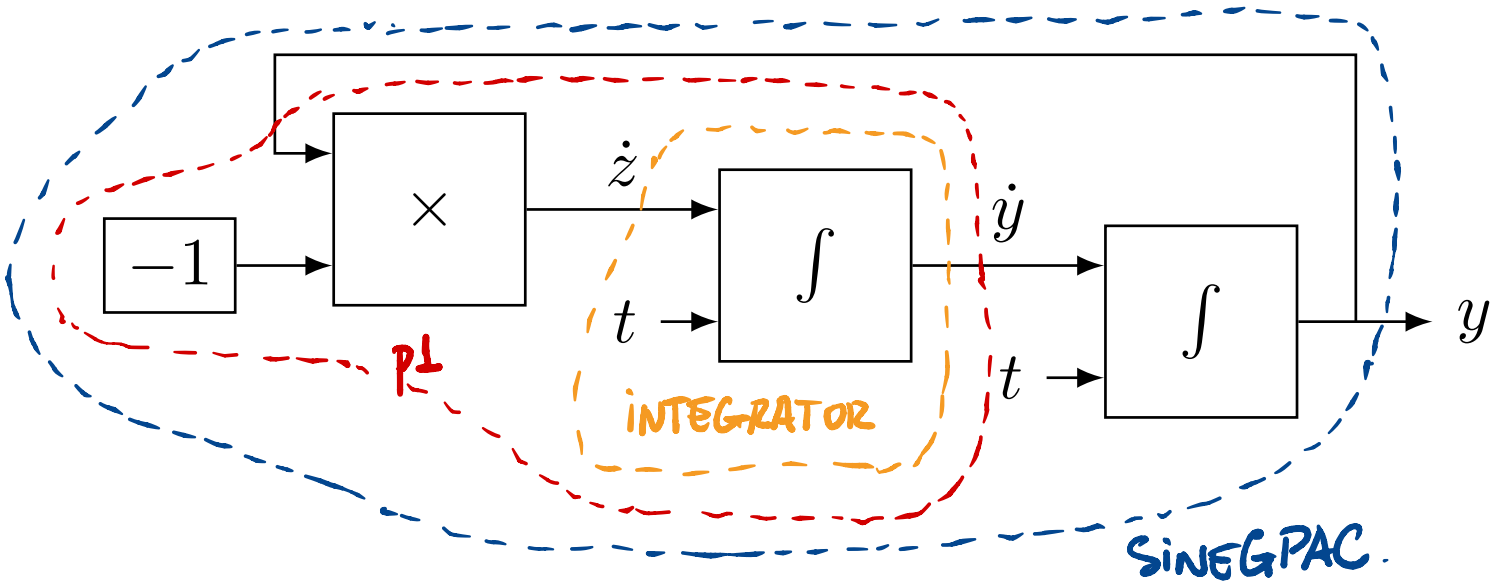
$$\int_{t_0, y_0} p_1 = (t, a) \mapsto (c, \int_{t,c} p_1')$$






```
at :: PCT () a -> Time -> a
p `at` t = pT
  where (pT, _) = prCT p t ()
```

"ADVANCE PROCESS p UNTIL TIME t".

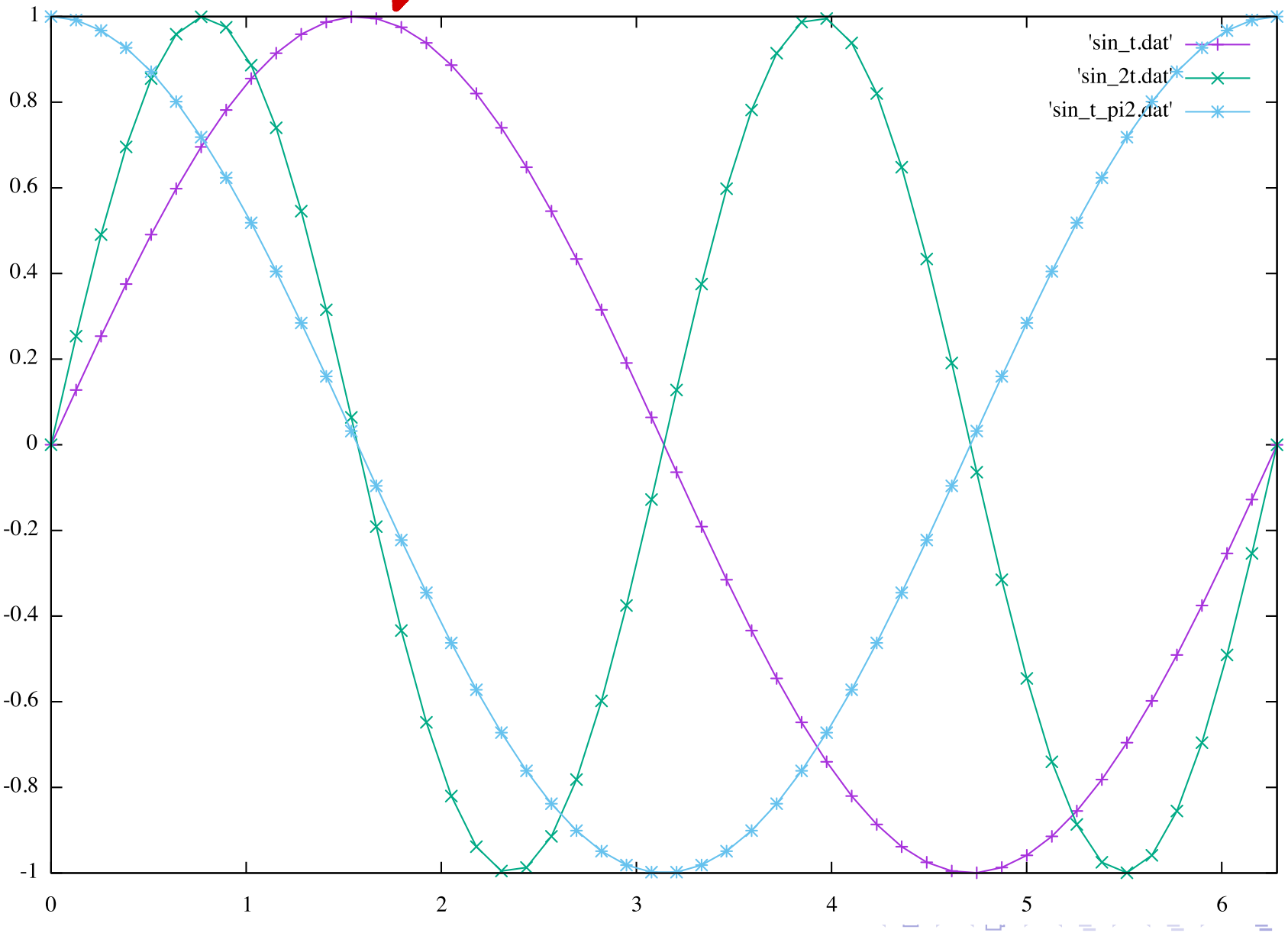



```
sineGPAC = intCT rk4 0 0 p1
  where
    p1 = (constCT (-1) *** idCT) >>> multCT >>> integrator
    integrator  = intCT rk4 0 1 loopBreaker
    loopBreaker = (idCT *** constCT 0) >>> adderCT
tScale k = (idCT &&& constCT k) >>> multCT
tShift k = (idCT &&& constCT k) >>> adderCT

p1 = time >>> sineGPAC
p2 = time >>> tScale 2 >>> sineGPAC
p3 = time >>> tShift (pi/2) >>> sineGPAC
```

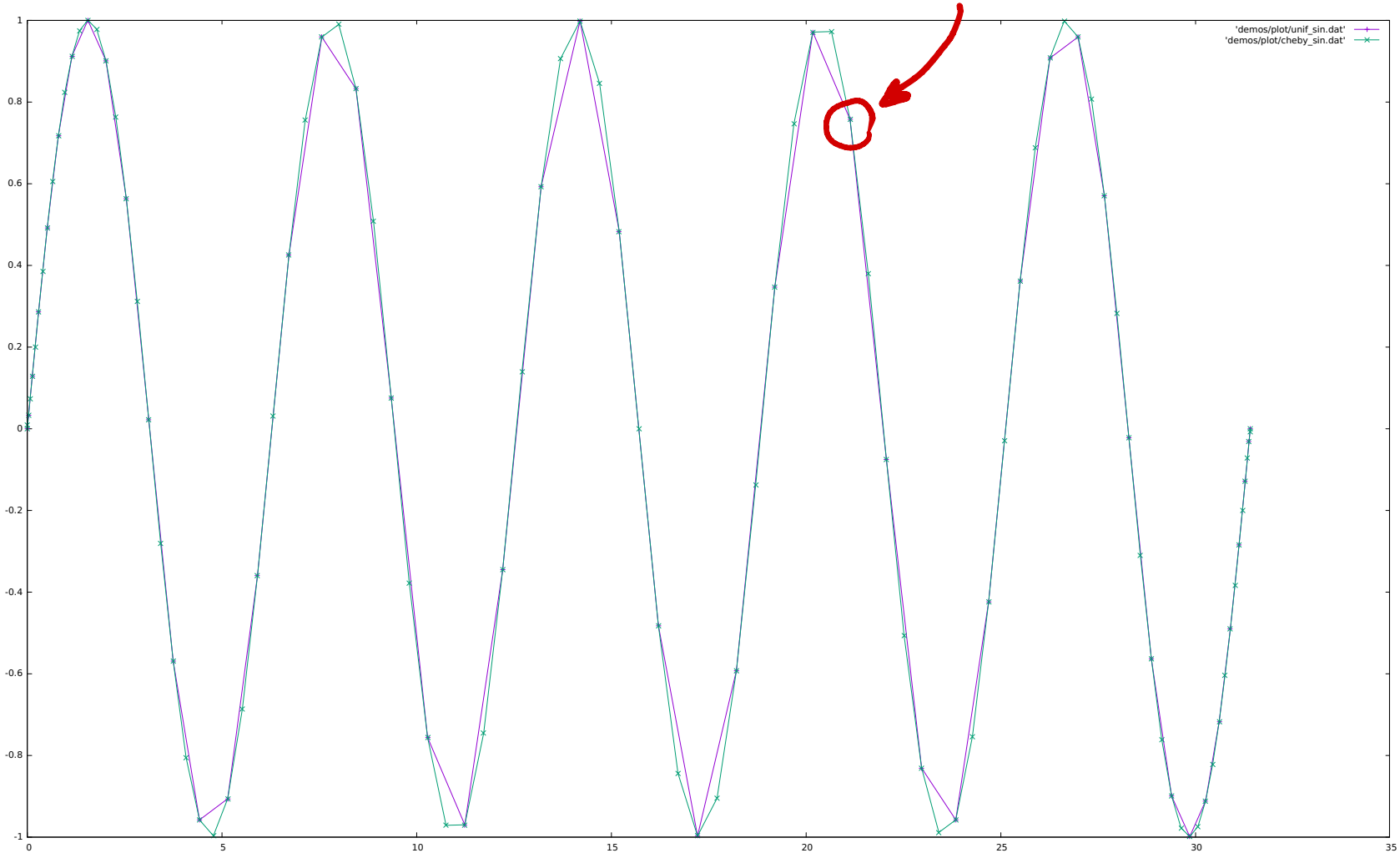
SAMPLER

TIME SCALING/SHIFTING IS NOT PART OF THE SYSTEM MODEL SEMANTICS.

SAME SYSTEM = DIFFERENT RESPONSES → TIMING SPECIFICATION SHALL BE EXPLICIT!



System response shall be independent of sampling.



a) Interaction with discrete-time MoCs.
b) Semantic preserving transformations.
c) Implementation in HW constrained platforms.


José E. G. de Medeiros, George Ungureanu, and Ingo Sander.
An algebra for modeling continuous time systems.
In *Design, Automation and Test in Europe (DATE)*. Dresden, 2018.


Ingo Sander, Axel Jantsch, and Seyed-Hosein Attarzadeh-Niaki.
ForSyDe: System design using a functional language and models of computation.
In *Handbook of Hardware/Software Codesign*. Springer, Dordrecht, 2017.

More information on ForSyDe

`https://forsyde.github.io/`