Opportunity in Embracing Imperfection: Is simulation the real thing?

Pieter J. Mosterman

Senior Research Scientist
Design Automation Department
MathWorks

Adjunct Professor
School of Computer Science
McGill

In your opinion, what lasting legacy has YACC brought to language development?

YACC made it possible for many people who were not language experts to make little languages (also called domain-specific languages) to improve their productivity. Also, the design style of YACC - base the program on solid theory, implement the theory well, and leave lots of escape hatches for the things you want to do that don't fit the theory - was something many Unix utilities embodied. It was part of the atmosphere in those days, and this design style has persisted in most of my work since then.

Interview with Stephen C. Johnson in “The A-Z of programming languages: YACC,” *Computerworld*, 09.07.2008
http://news.idg.no/cw/art.cfm?id=094E3B6E-17A4-0F78-311509693E8E95C1

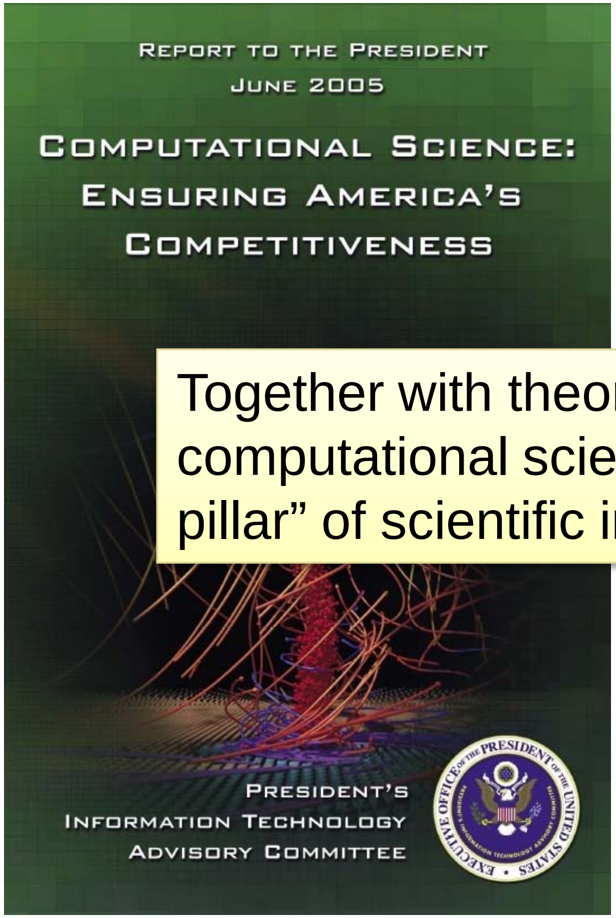
Opportunity in Embracing Imperfection: Is simulation the real thing?

Pieter J. Mosterman

Senior Research Scientist
Design Automation Department
MathWorks

Adjunct Professor
School of Computer Science
McGill

The importance of computation



Together with theory and experimentation, computational science now constitutes the “third pillar” of scientific inquiry,

The importance of computation

H. Res. 487

In the House of Representatives, U. S.,
July 16, 2007.

Whereas the United States of America is a great and prosperous Nation, and modeling and simulation contribute

uted to the United States by—

(1) expanding the understanding of nuclear chain reactions during the Manhattan Project through some of the earliest simulations replicating the reaction process, which ultimately contributed to the end of World War II;

(2) serving as a foundational element of the Stockpile Stewardship Program, which enabled the President of the United States to certify the safety, security, and reliability of the nuclear stockpile for more than ten years

Resolved, That the House of Representatives—

3) encourages the expansion of modeling and simulation as a tool and subject within higher education;
4) recognizes modeling and simulation as a National Critical Technology;

Agenda

- Outline
- Model-Based Design
- Problem statement
- A solution approach
- Outlook

The science in engineering a system

experimentation

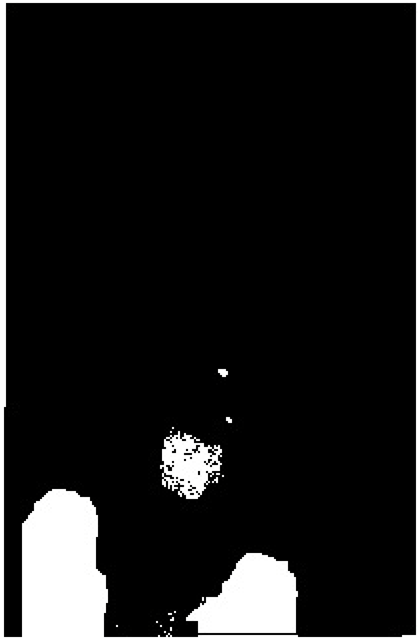
theory

Discard detail but keep pertinent behavior

The science in engineering a system

experimentation

theory



Where is the heat?

But often no analytical solution

The science in engineering a system

experimentation

theory

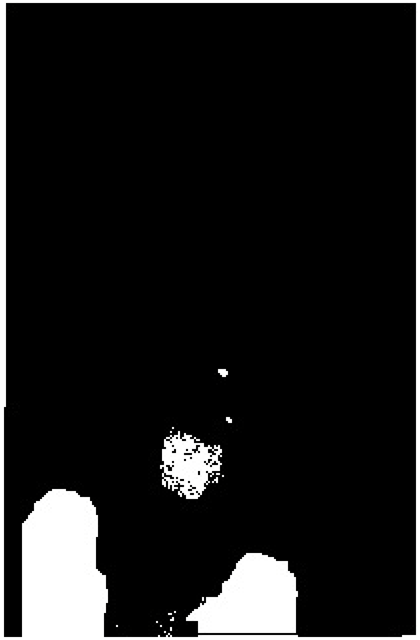
computation

A computational solution

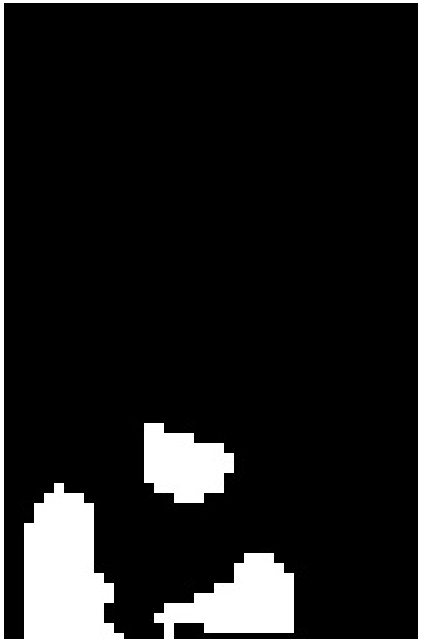
The science in engineering a system

experimentation

theory



computation



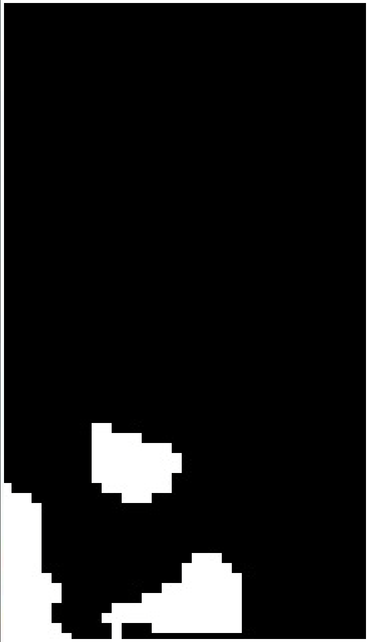
A computational solution

Not much heat at the nozzles … let’s change the material …

The science in engineering a system

experimentation

computation



Not much he

putational solution

e the material …

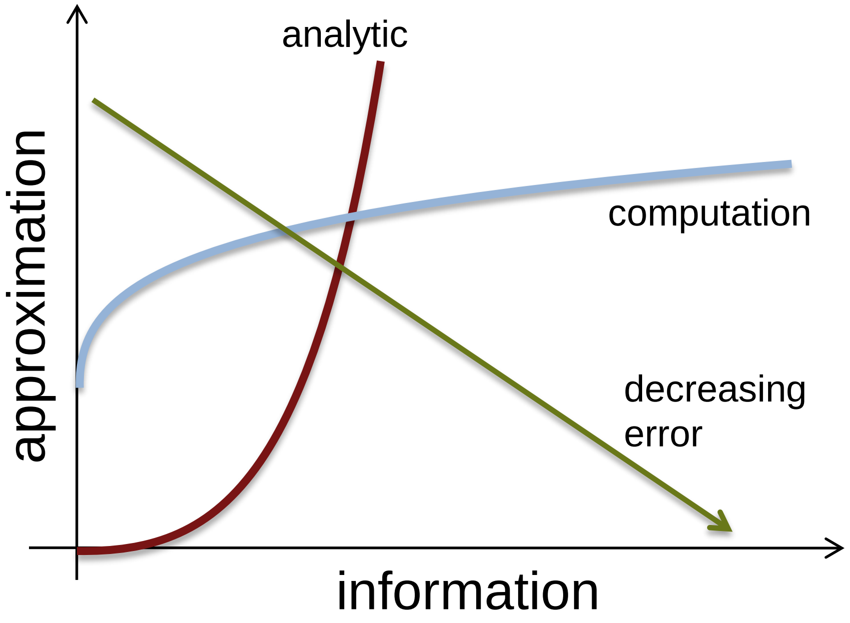
Computational methods to add more detail

Same computational approximation
Information beyond what is in a first principles model

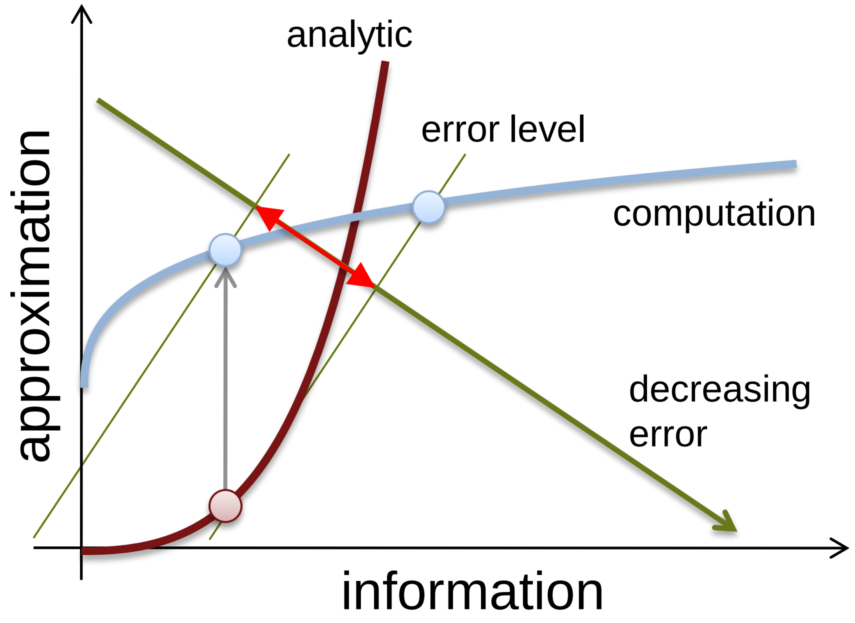
Computational methods to add more detail

Same computational approximation

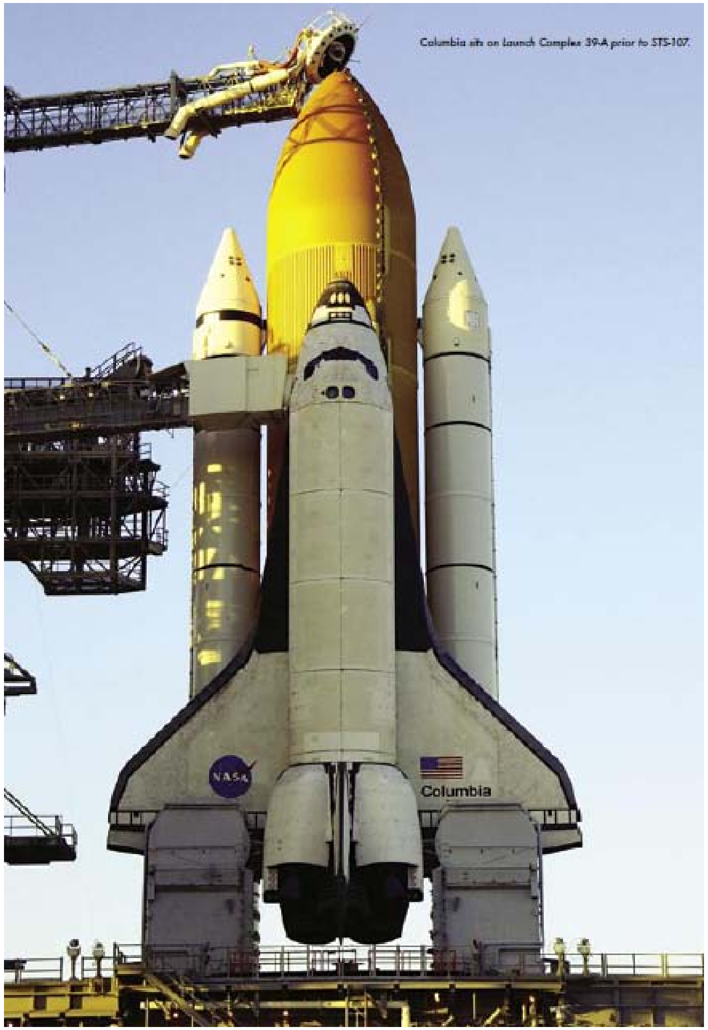
Model quality



Model quality



Computational methods are not that mature



Computational methods are not that mature

" … engineers used Crater during STS-107 to analyze a piece of debris that was at maximum 640 times larger in volume than the pieces of debris used to calibrate and validate the Crater model."

H. W. Gehman, Jr. *et al.*,"Report of Columbia Accident Investigation Board, Volume I," *National Aeronautics and Space Administration*, August, 2003

Technology maturation: a comparison

Conservative | Cautious improvement | … till failure

1849 → 1883 → 1940

Széchenyi Chain Bridge

Tacoma Narrows Bridge

Brooklyn Bridge

"Computational Science Demands a New Paradigm," Douglass E. Post and Lawrence G. Votta, *Physics Today*, 2005

Technology maturation: a comparison

Conservative | Cautious improvement | ***<u>… till failure</u>*** | Firm methodology

1849 → **1883** → **1940** → **1998**

Széchenyi Chain Bridge

Tacoma Narrows Bridge

Brooklyn Bridge

Akashi-Kaikyō Bridge

"Computational Science Demands a New Paradigm," Douglass E. Post and Lawrence G. Votta, *Physics Today*, 2005

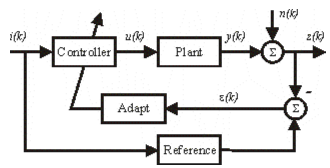
Premise

- Approximation is <u>not</u> the culprit
  - Model-Based Design is successfully exploiting computation
  - But still very *ad hoc*; lots of testing required
- Embrace the imperfection!
  - Create better models <u>without</u> reducing the approximation
  - Use computational methods to:
    - Enhance model information
    - Enhance domain information
    - Analyze and design complex execution engines
  - Requires precise definition of the execution semantics
    - Differential equations, difference equations, discrete event, etc.
    - Approximations
- Treat computation as <u>equal</u> to experiment and theory

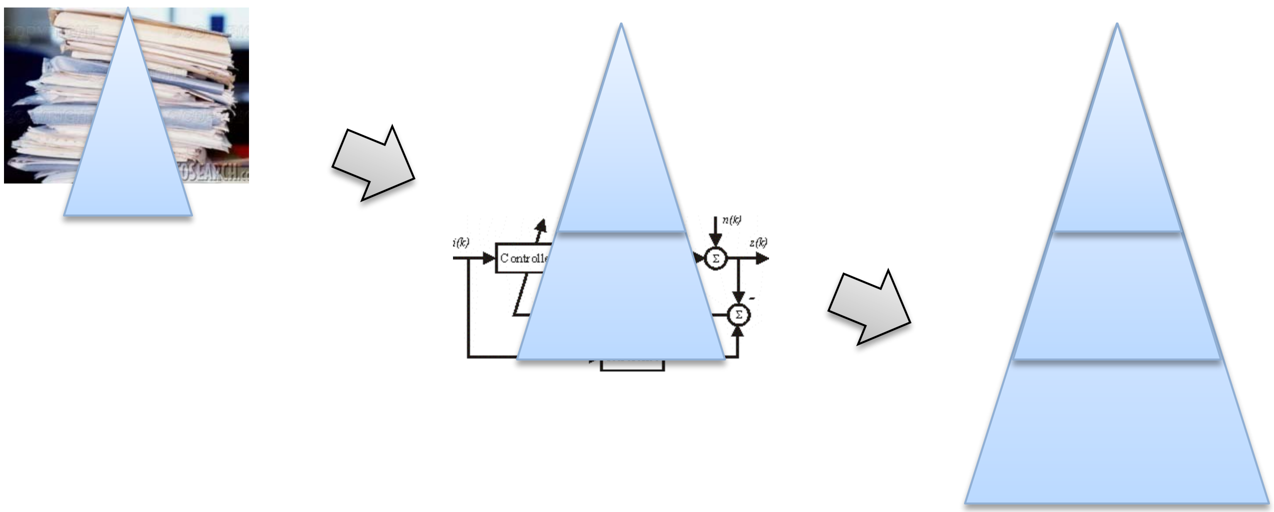
Agenda

- Outline
- Model-Based Design
- Problem statement
- A solution approach
- Outlook

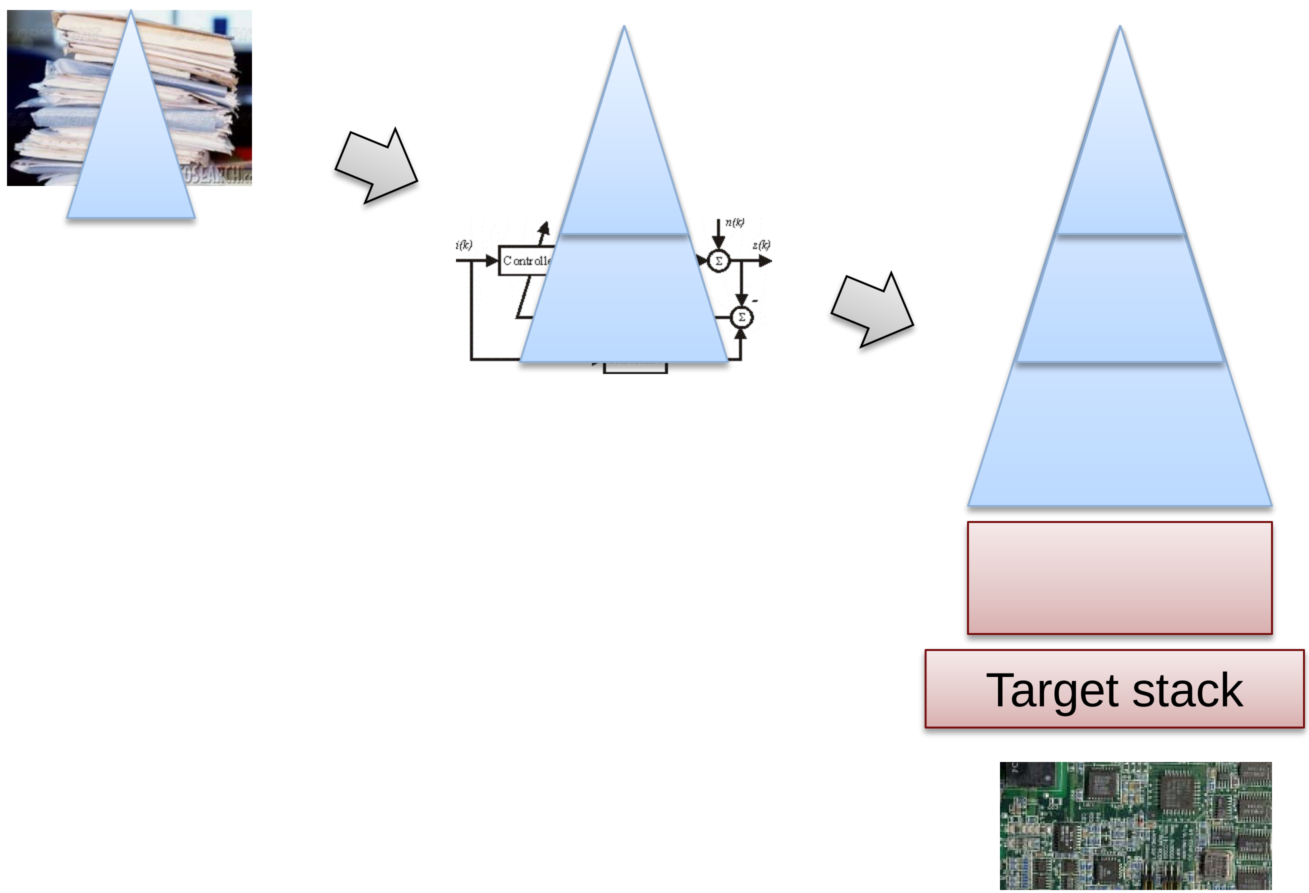
Design of an engineered system



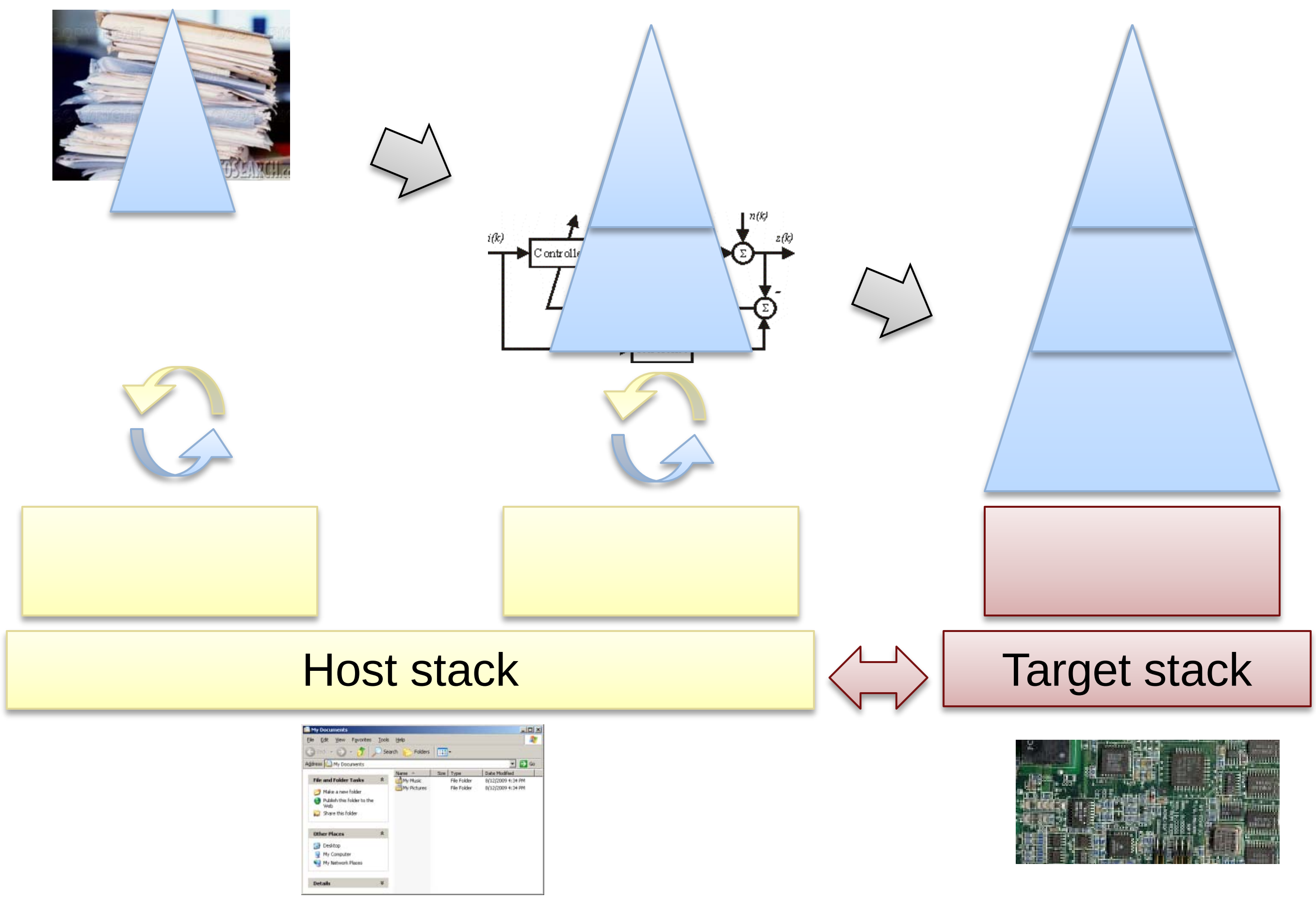
Increasingly more detail



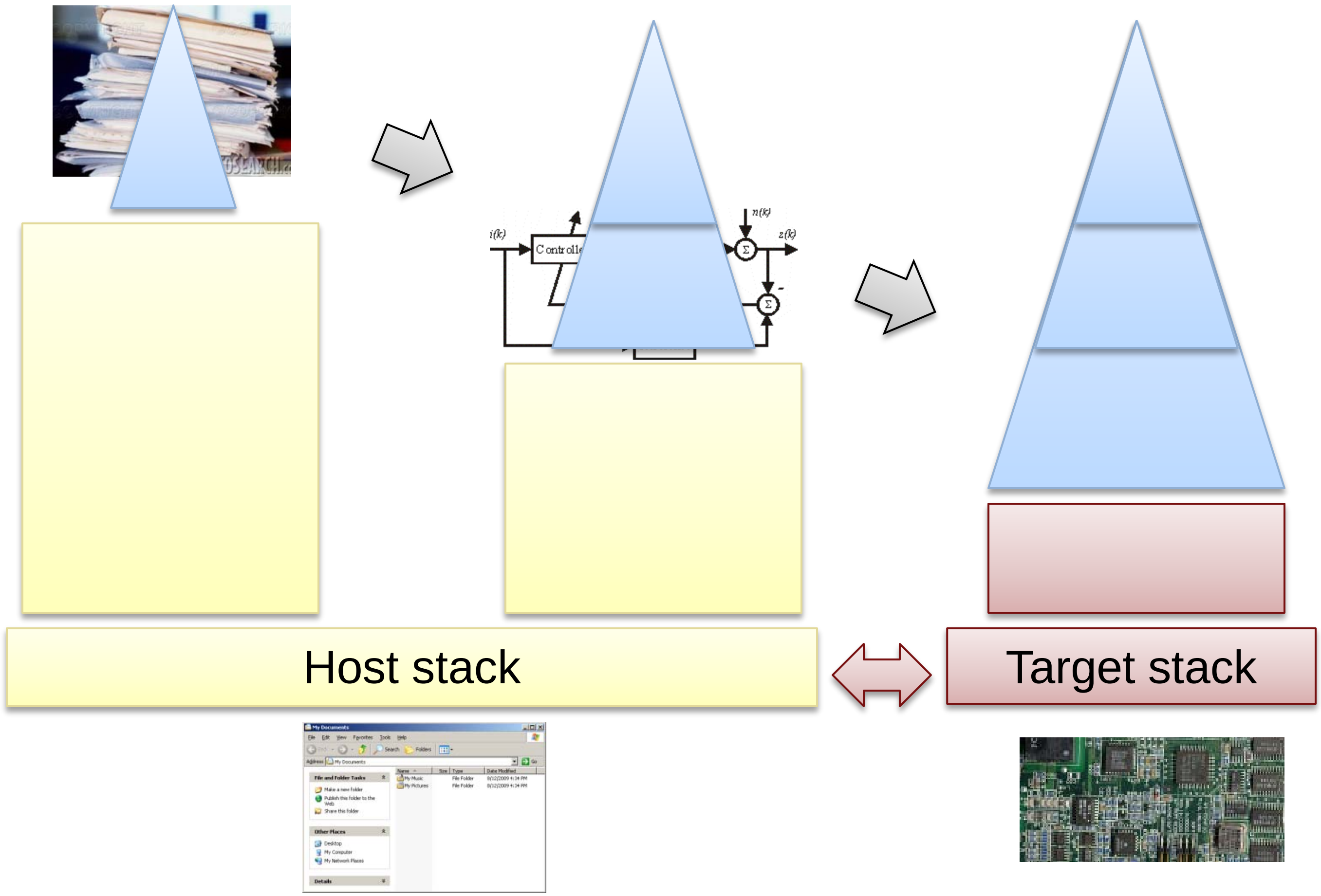
System behavior



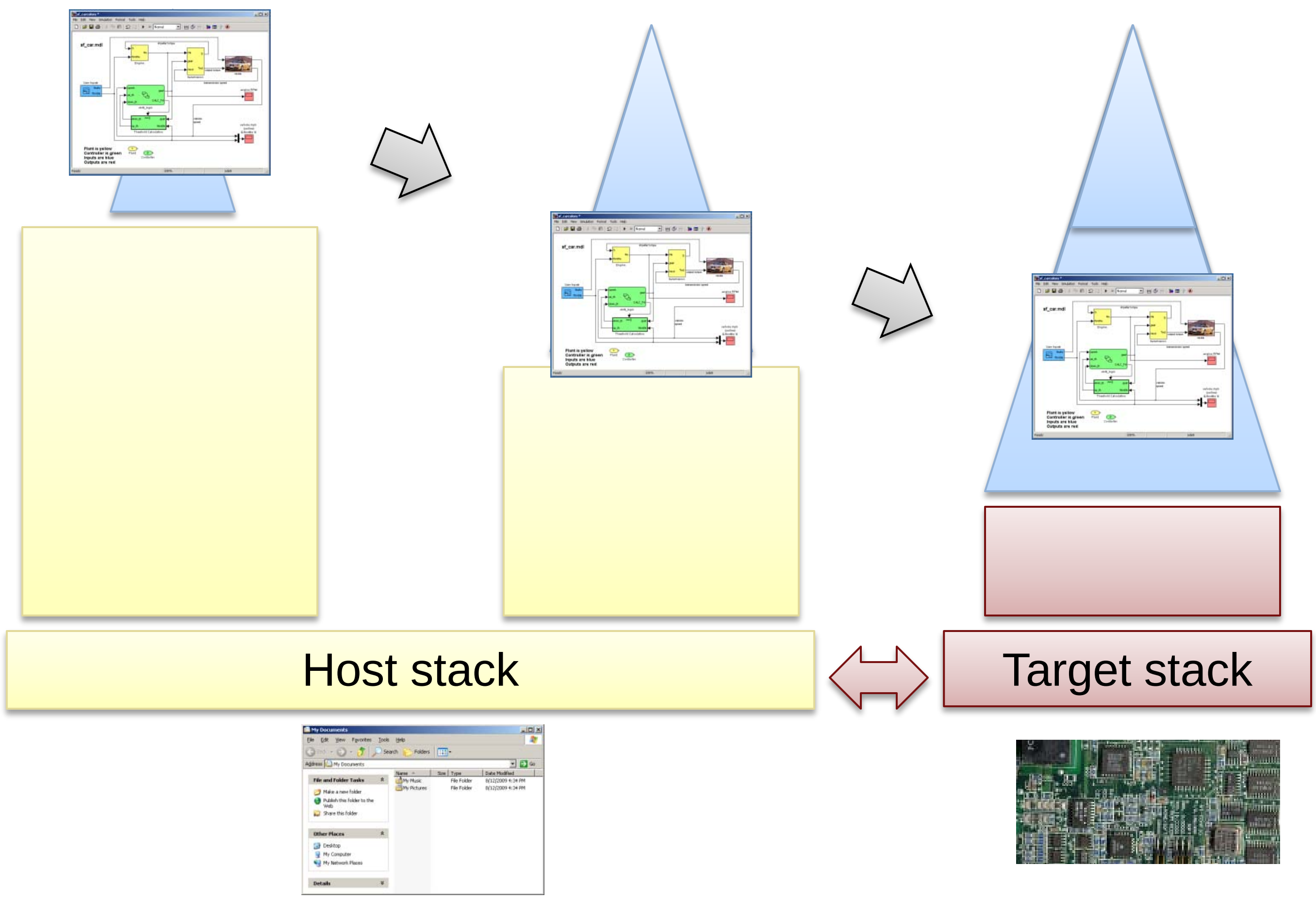
Simulation studies



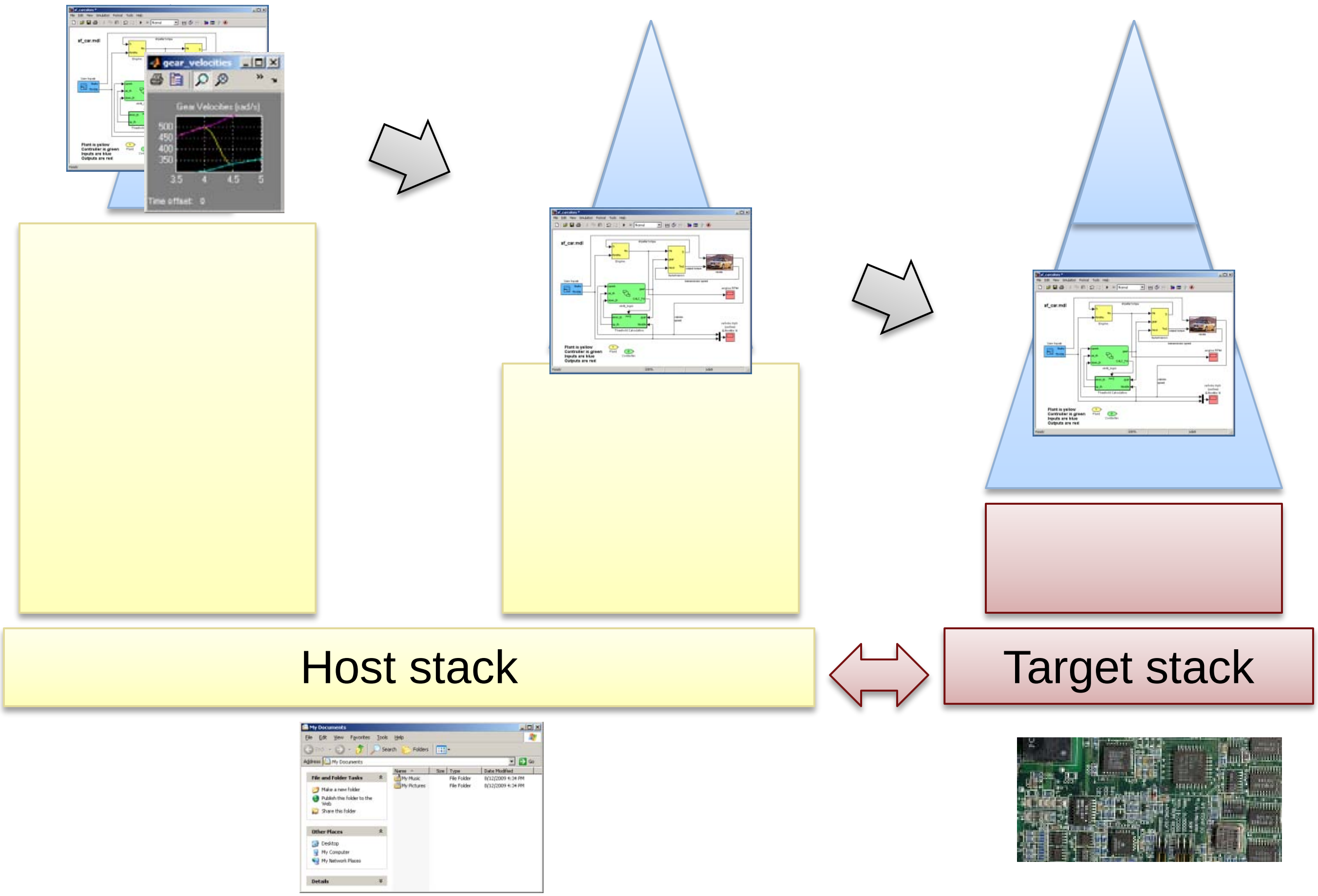
Model-Based Design



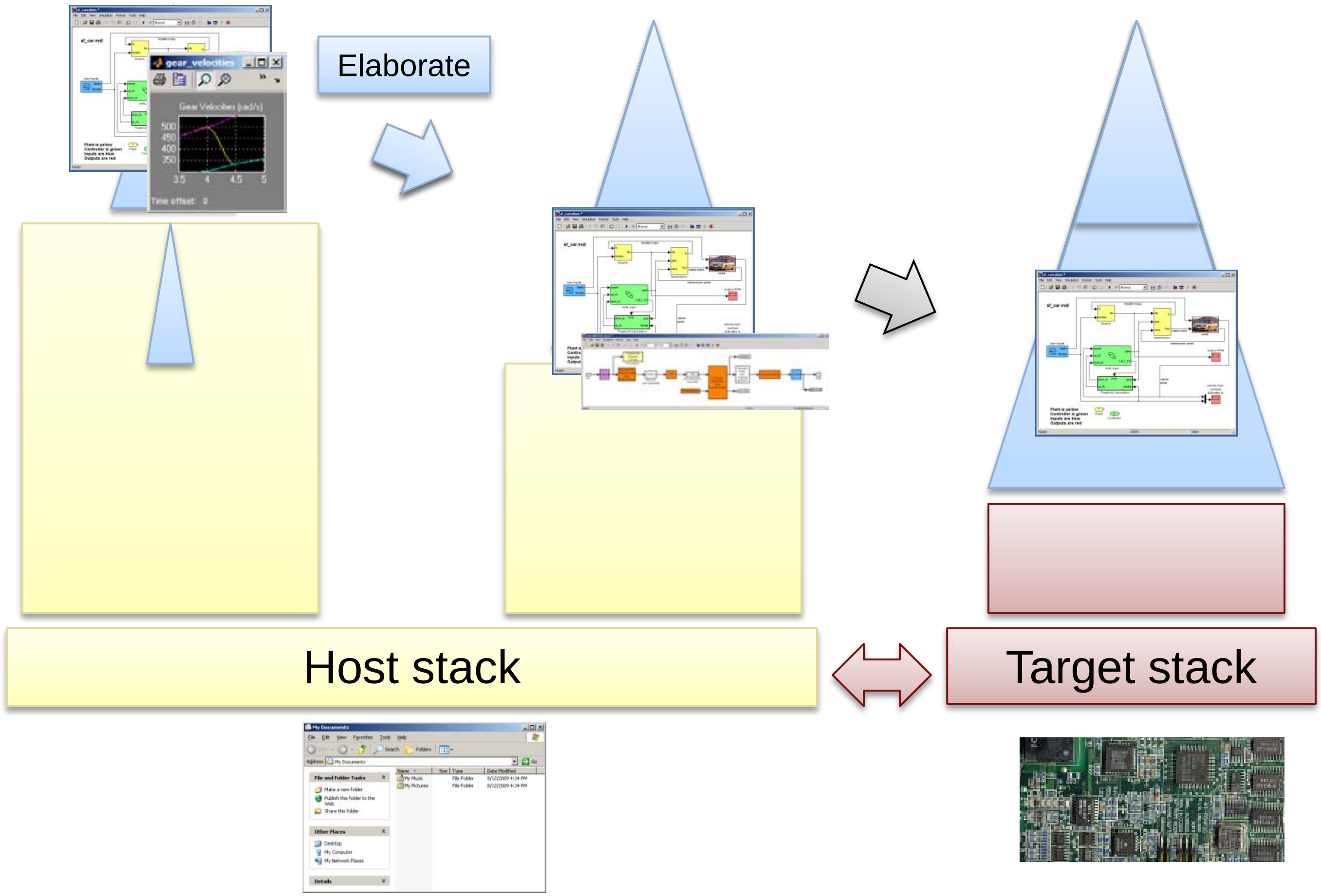
Model-Based Design



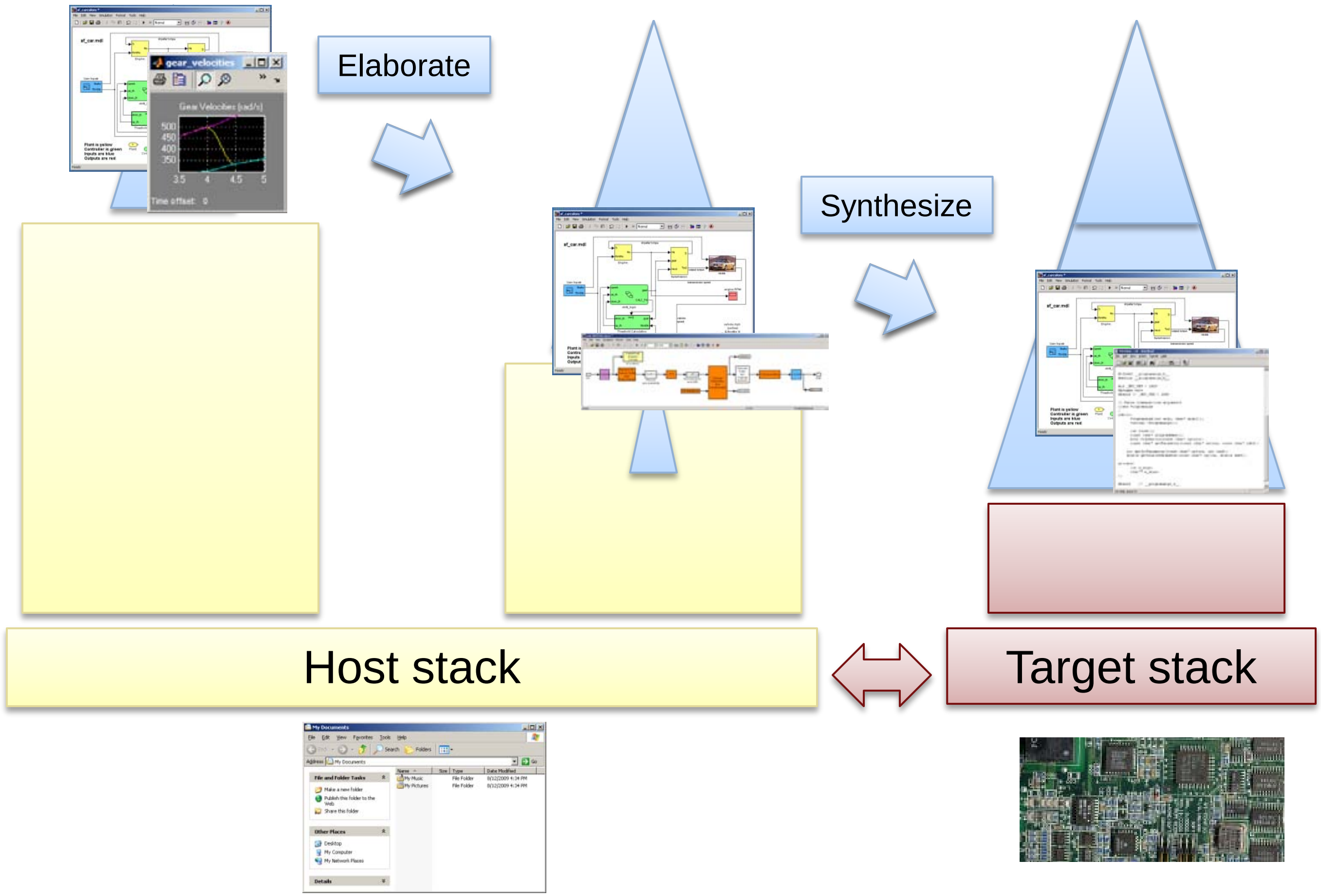
Executable specifications



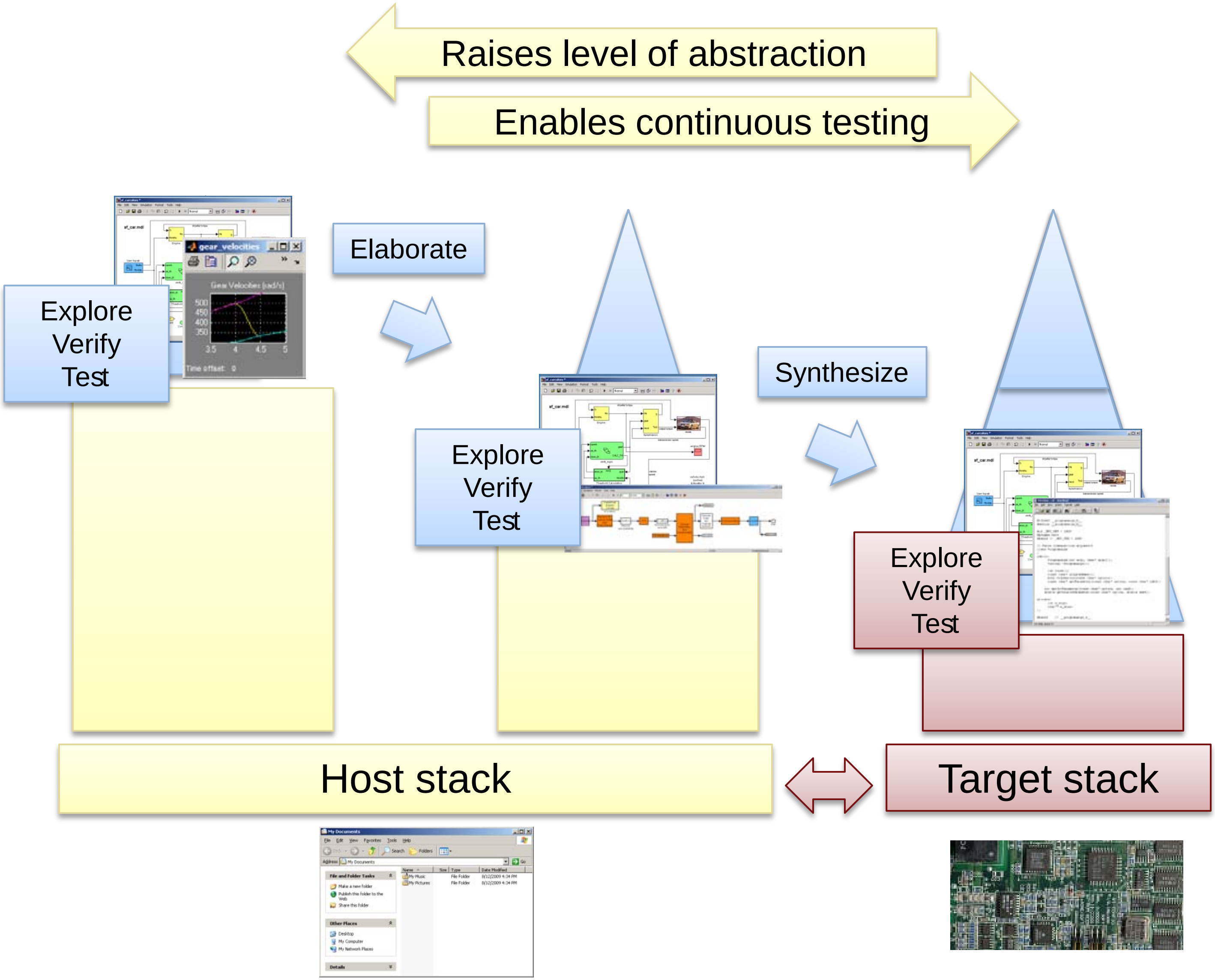
Model elaboration



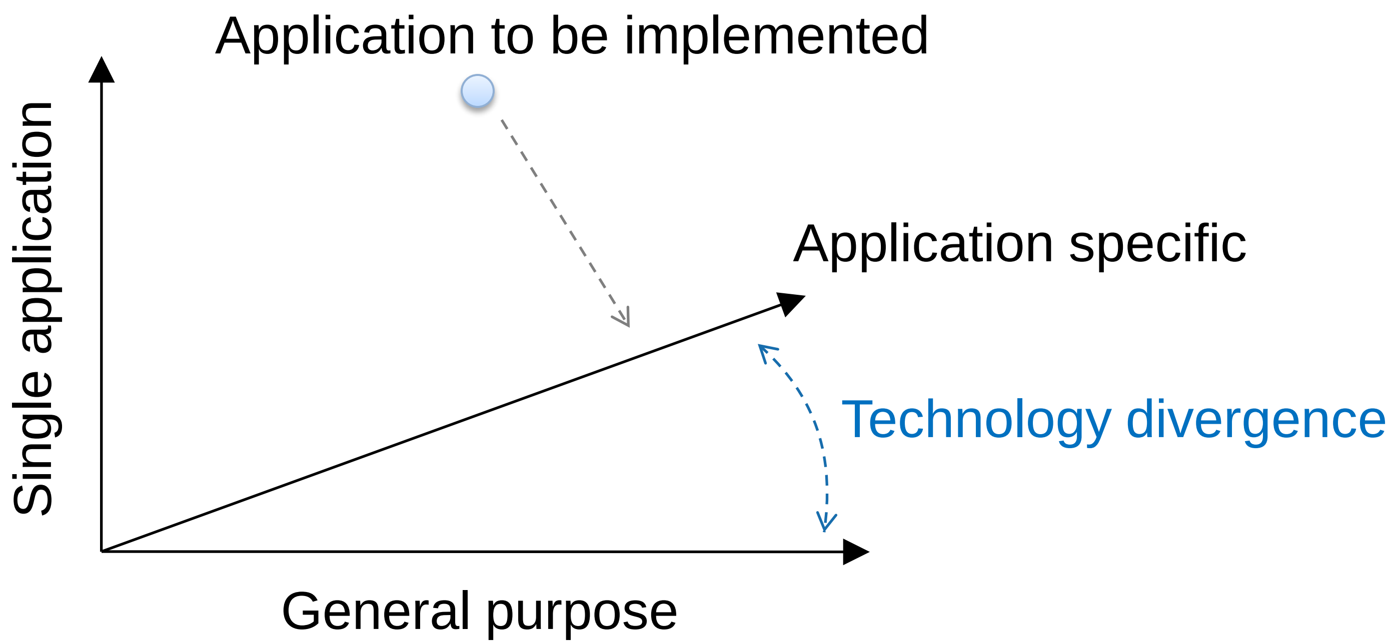
Automatic code generation



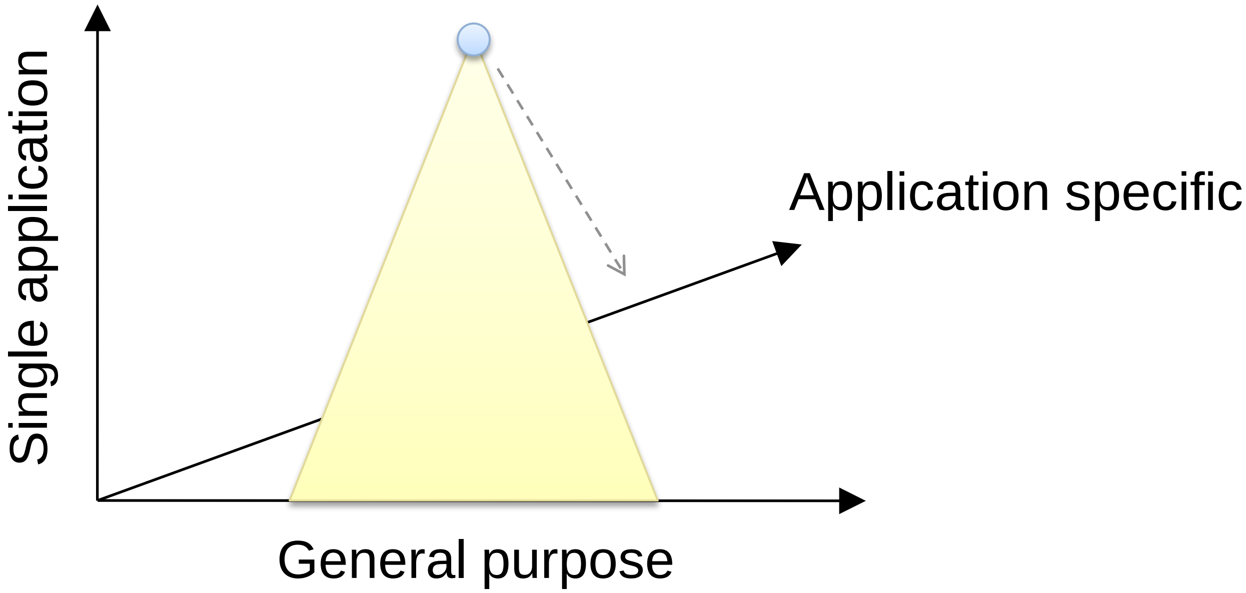
Model-Based Design



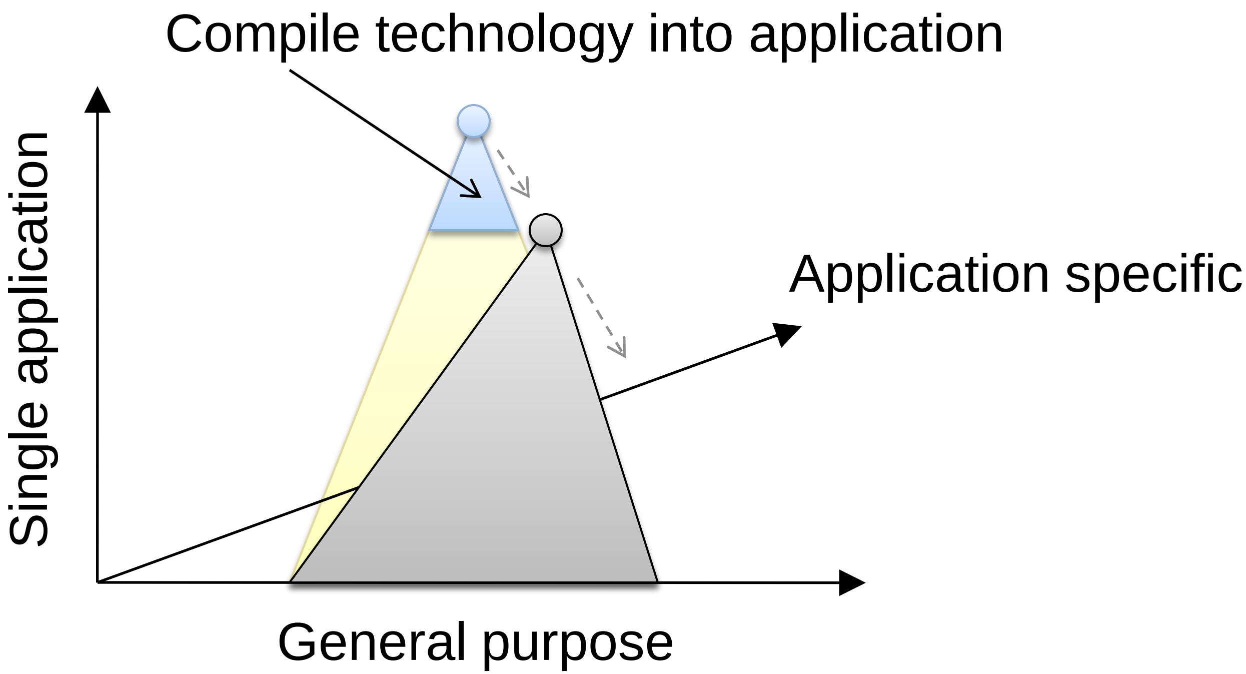
Mapping an application



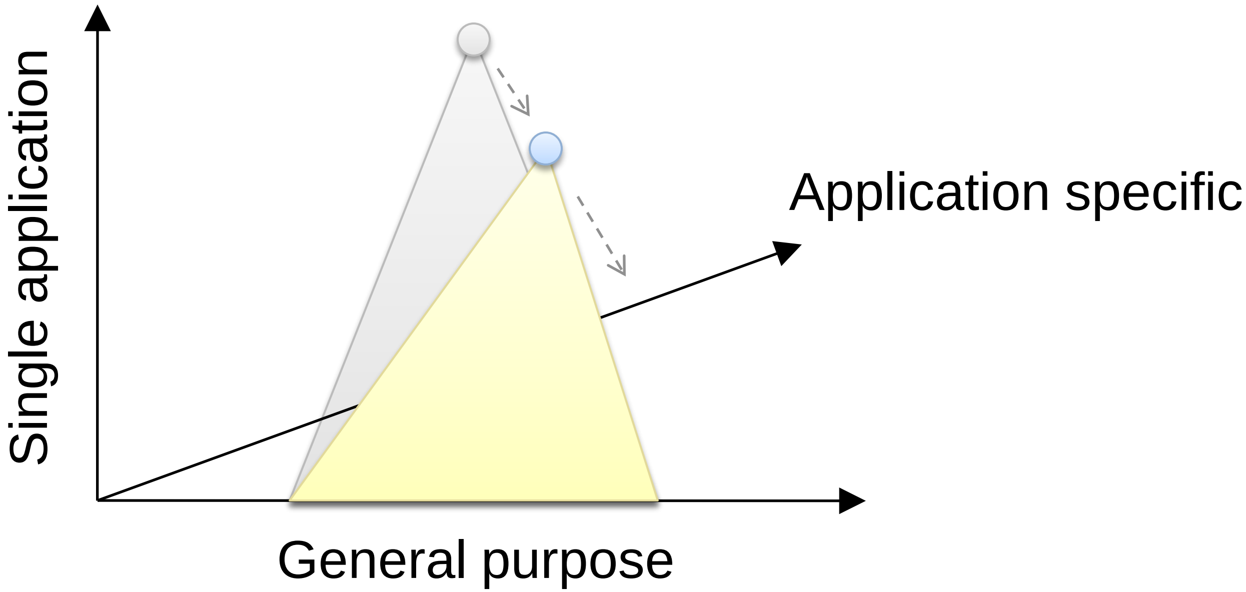
Mapping an application



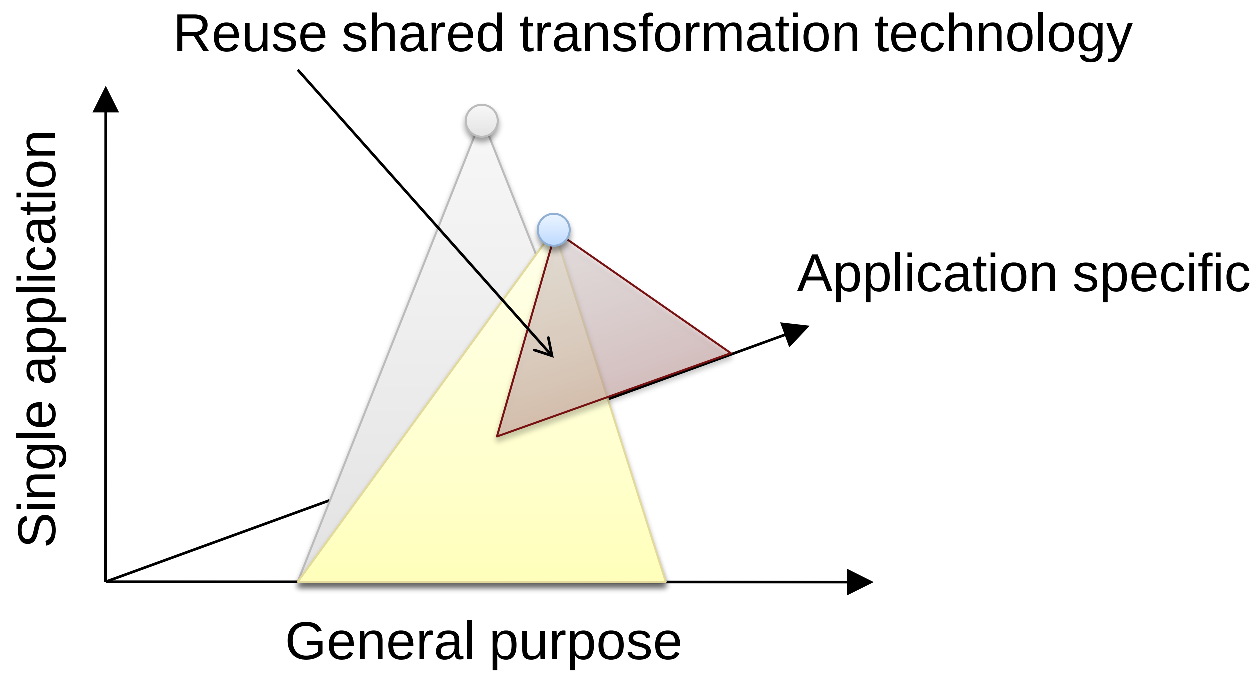
Mapping an application



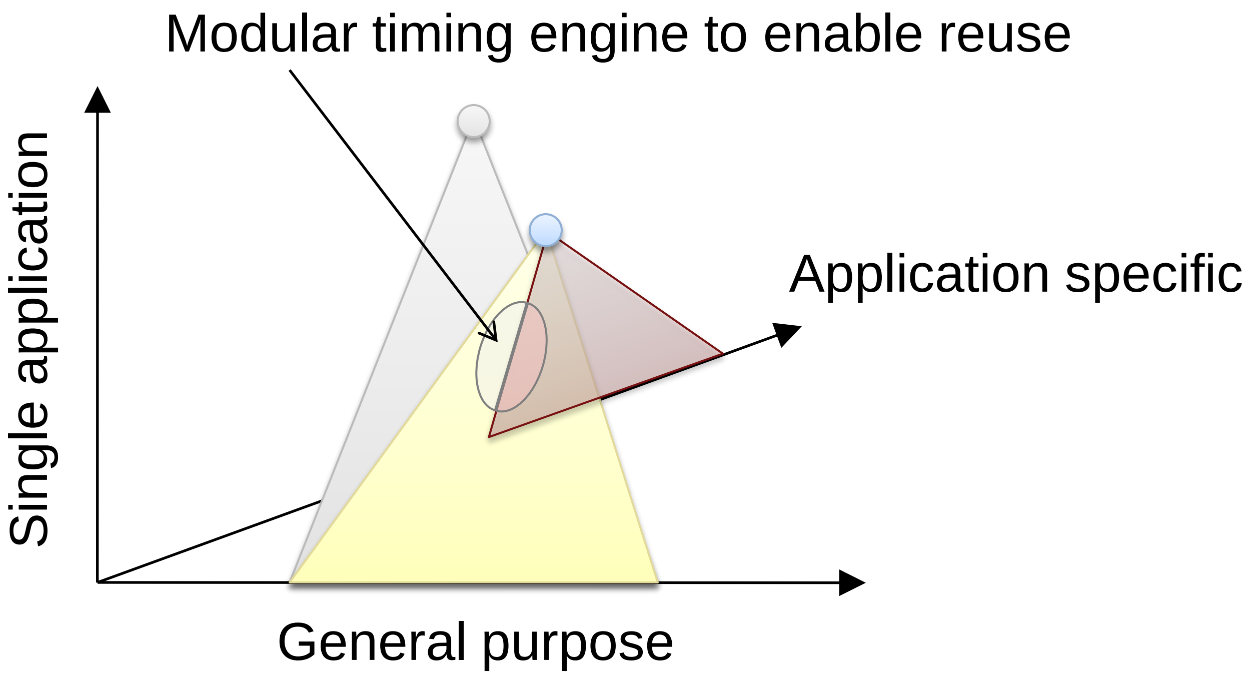
Mapping an application



Mapping an application



Mapping an application

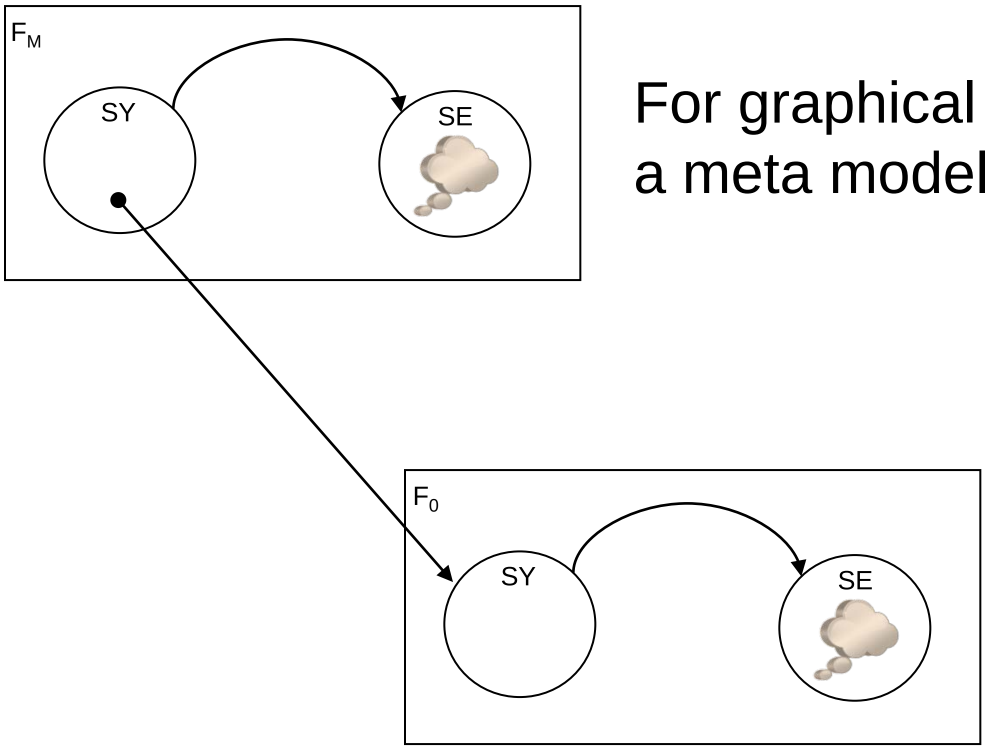


Agenda

- Outline
- Model-Based Design
- Problem statement
- A solution approach
- Outlook

Computer Automated Multiparadigm Modeling for technology reuse

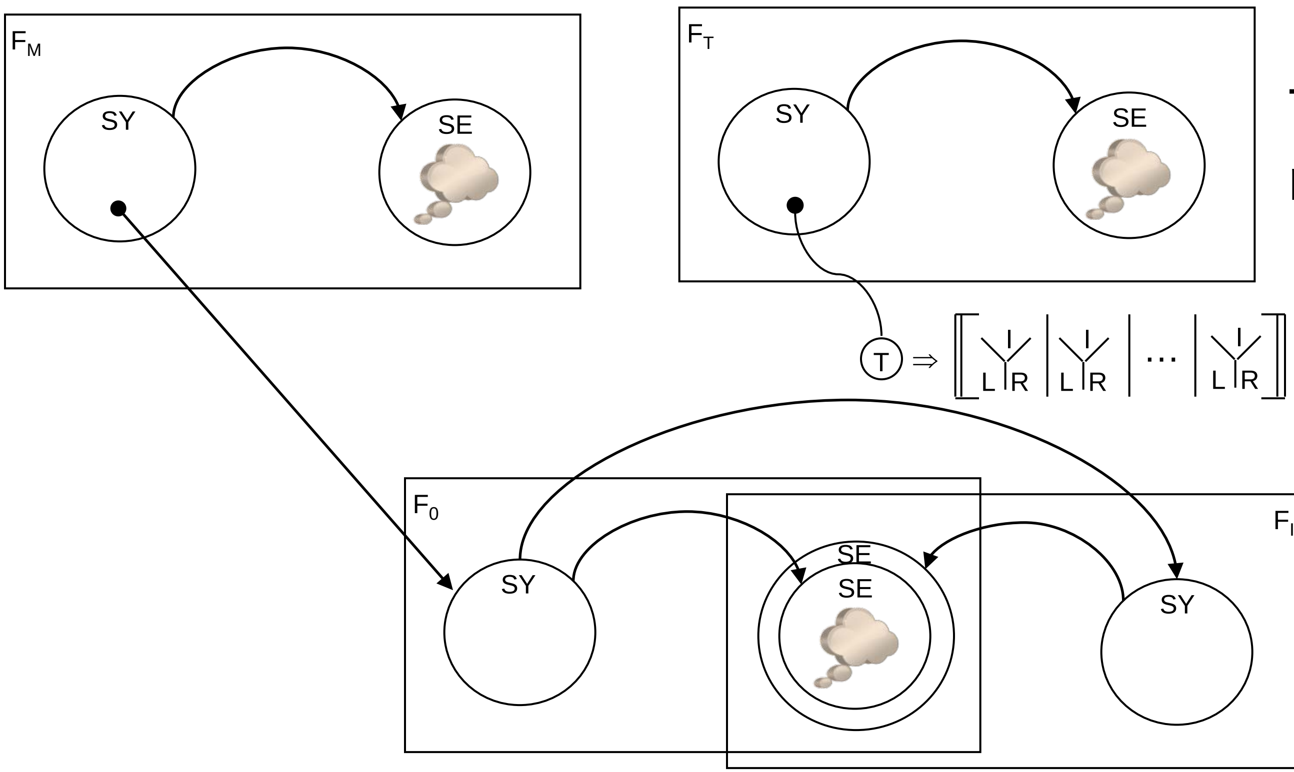
- A syntax, a semantic domain, and a mapping



For graphical syntax, often a meta model is used

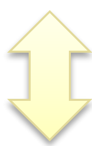
Define semantics as a syntactic transformation—semantic anchoring

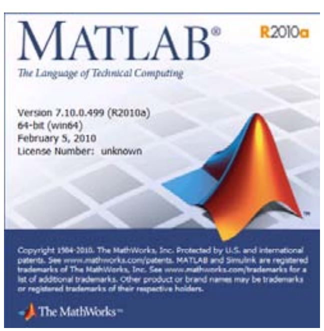
- Target semantic domain must be subsumed

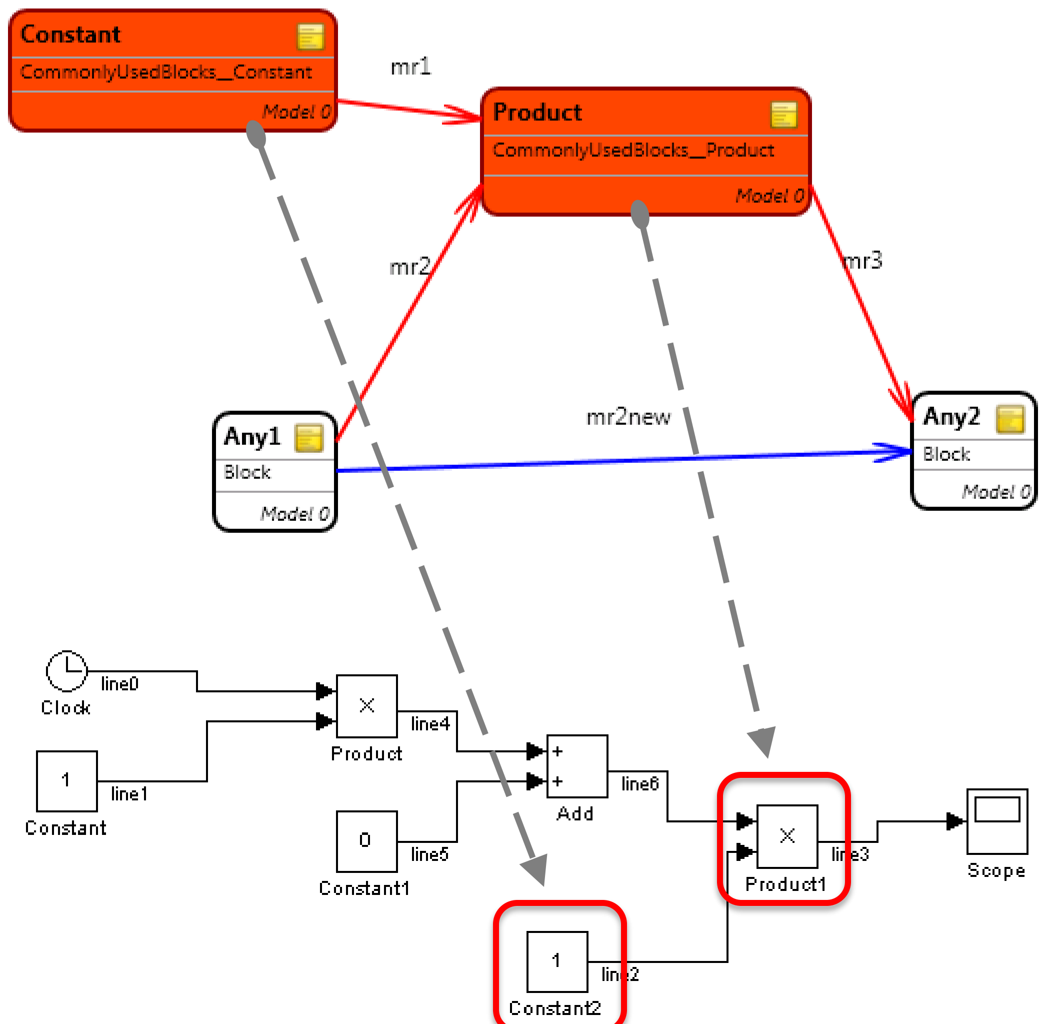


Transformation model

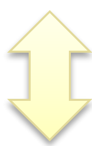
Modeling a model transformation

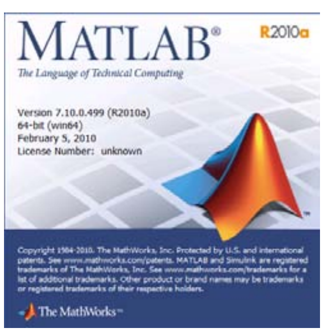


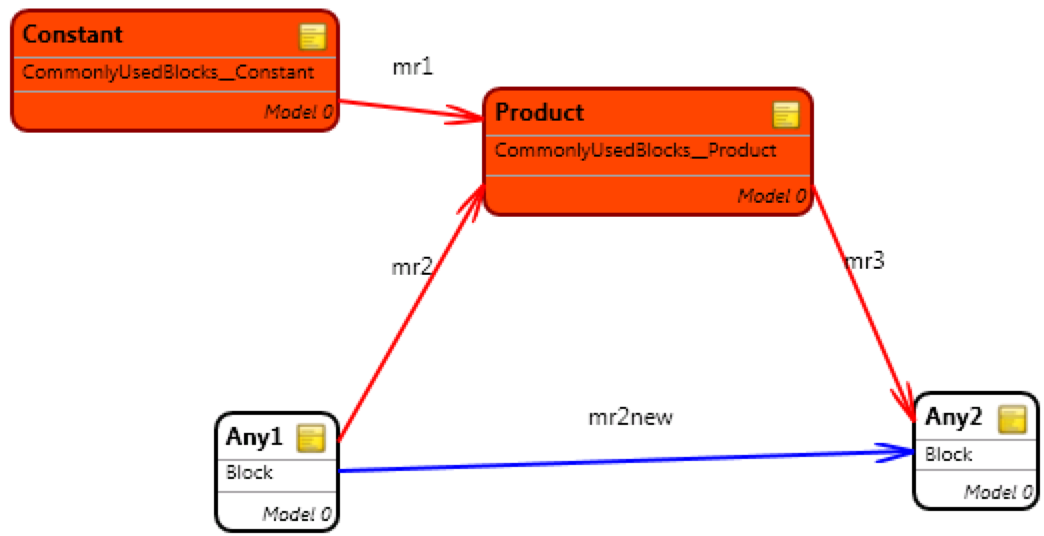


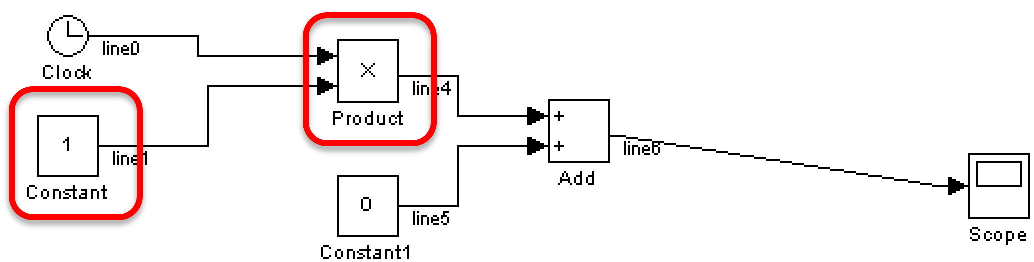


Modeling a model transformation

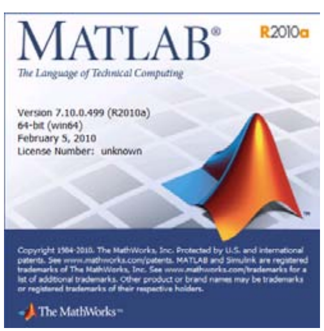


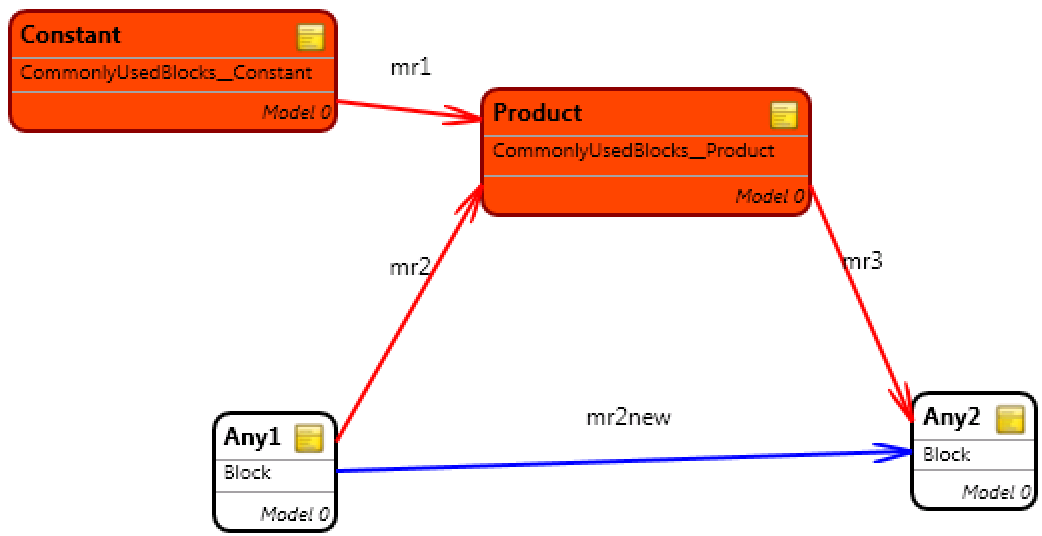


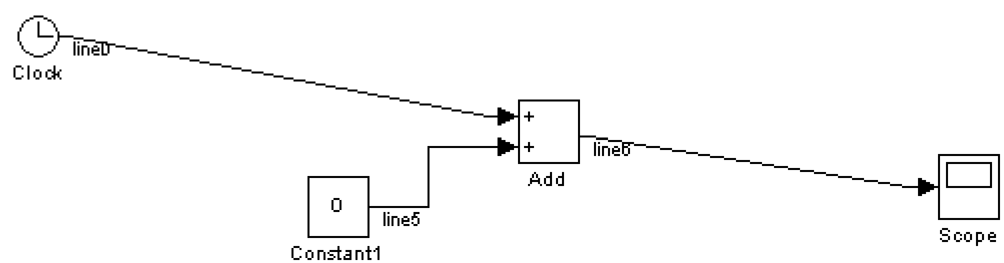




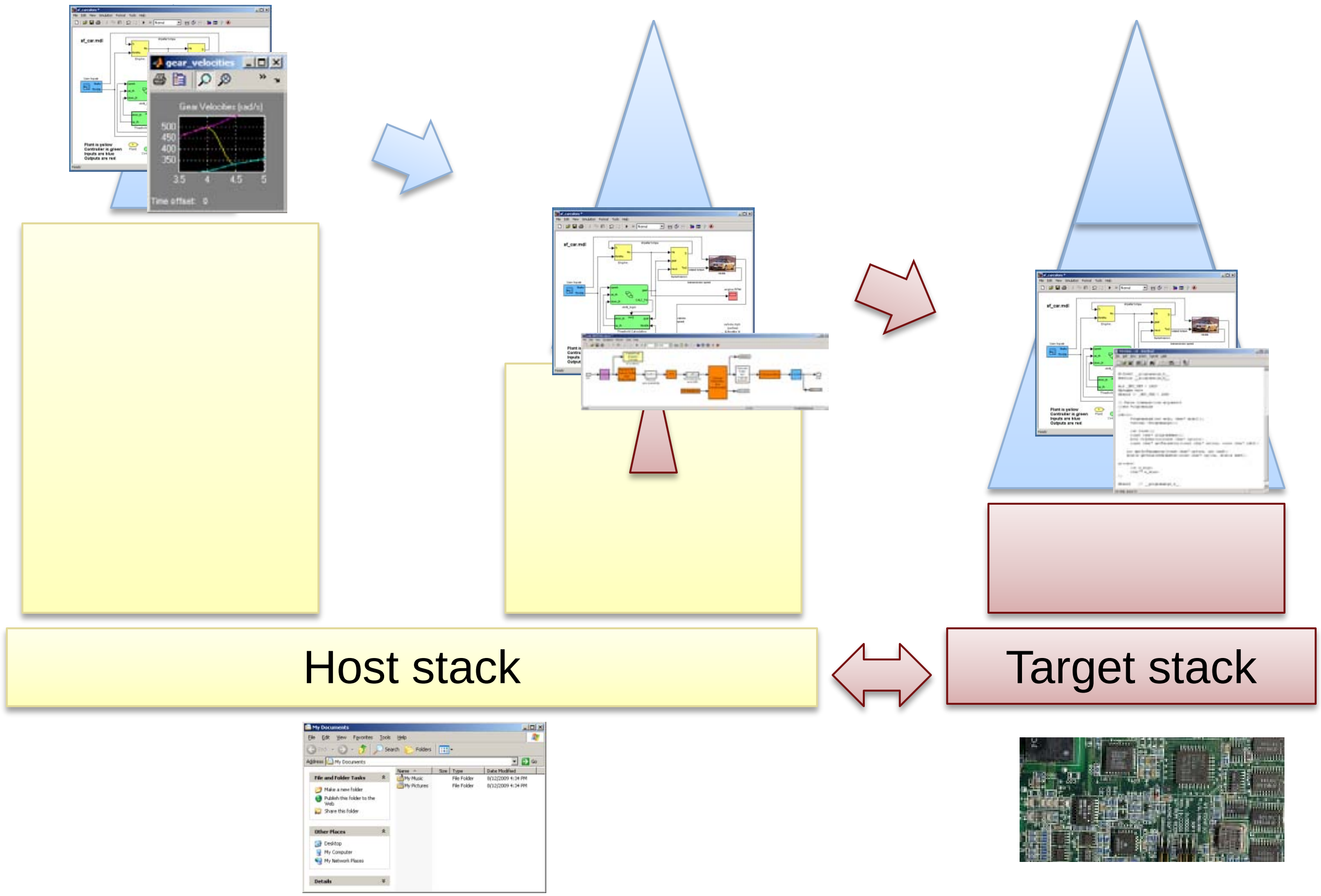
Modeling a model transformation



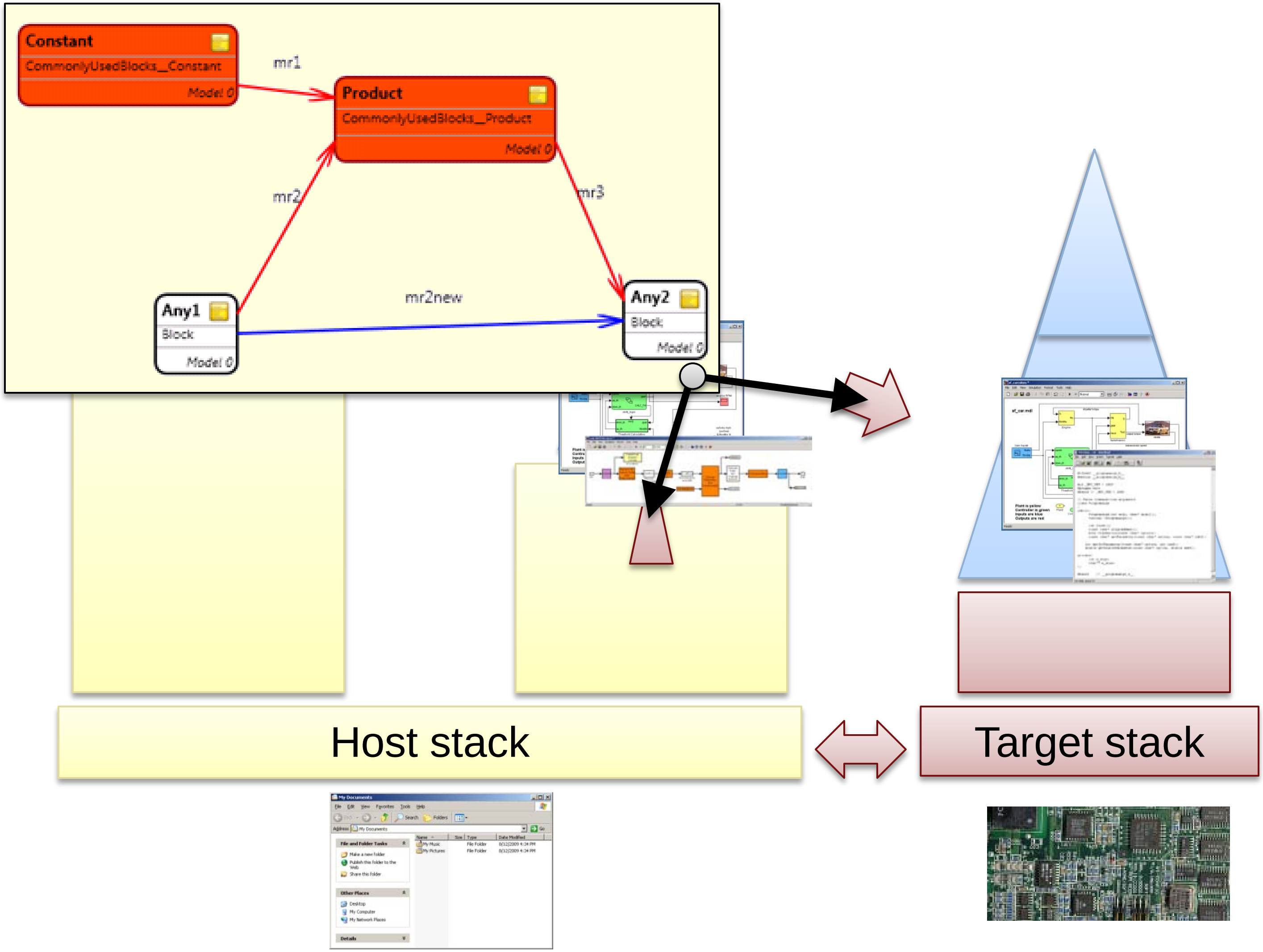




Model-Based Design

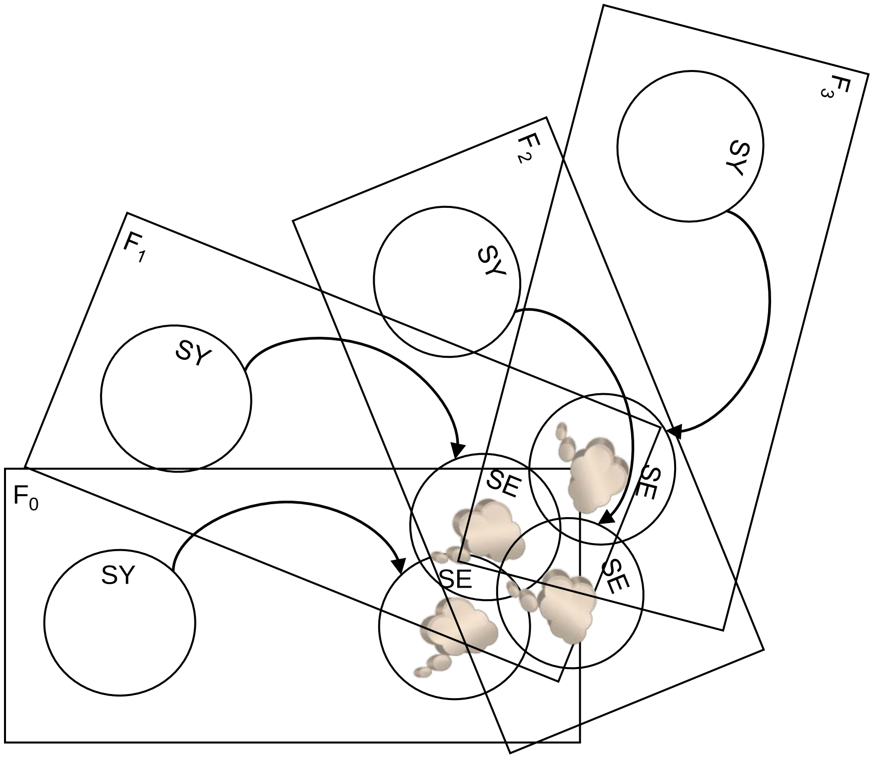


Model-Based Design



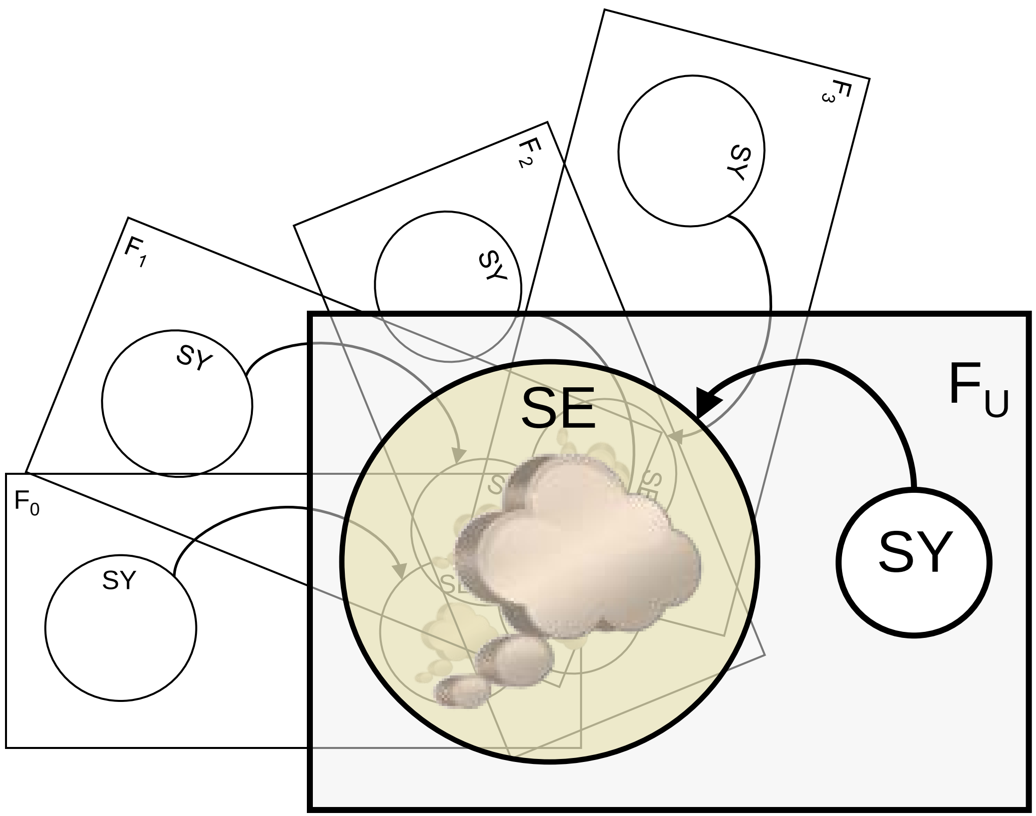
Multi-domain models comprise many formalisms …

- Can we develop a unifying semantic domain?



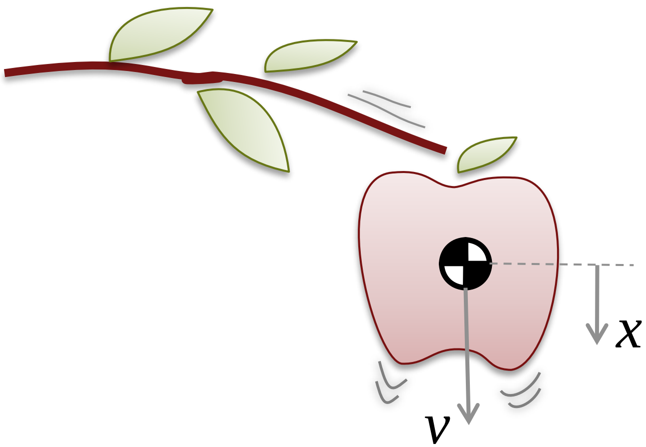
Multi-domain models comprise many formalisms …

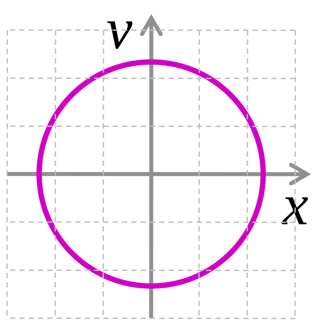
- Can we develop a unifying semantic domain?



Modeling a physical system

From first principles …





Hooke's Law: $$F = -\frac{x - x_0}{C}$$

Newton's Second: $$F = ma$$

A bit of calculus: $$a(t) = \frac{dv(t)}{dt}$$

$$v(t) = \frac{dx(t)}{dt}$$

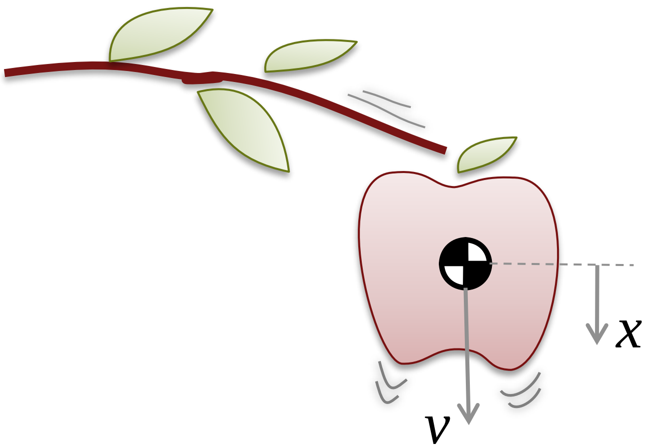
An ideal oscillator:

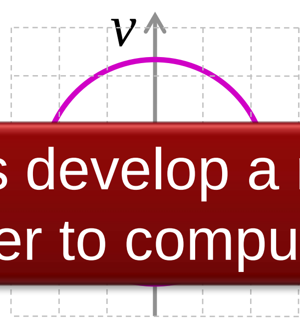
$$v(t) = \frac{dx(t)}{dt}$$

$$m\frac{dv(t)}{dt} = -\frac{x(t) - x_0}{C}$$

Modeling a physical system

From first principles …





Let's develop a numerical solver to compute a solution …

Hooke's Law: $$F = -\frac{x - x_0}{C}$$

Newton's Second: $$F = ma$$

A bit of calculus: $$a(t) = \frac{dv(t)}{dt}$$

$$v(t) = \frac{dx(t)}{dt}$$

An ideal oscillator:

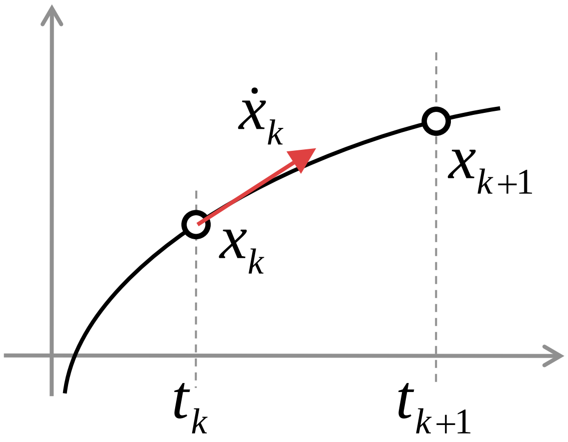
$$v(t) = \frac{dx(t)}{dt}$$

$$m\frac{dv(t)}{dt} = -\frac{x(t) - x_0}{C}$$

Numerical integration

Euler: step h in time along $\dot{x} = f(x,t)$

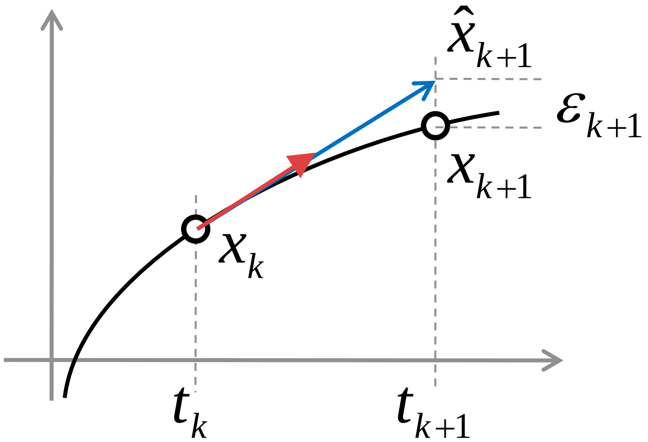
$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$



Numerical integration

Euler: step h in time along $\dot{x} = f(x,t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$



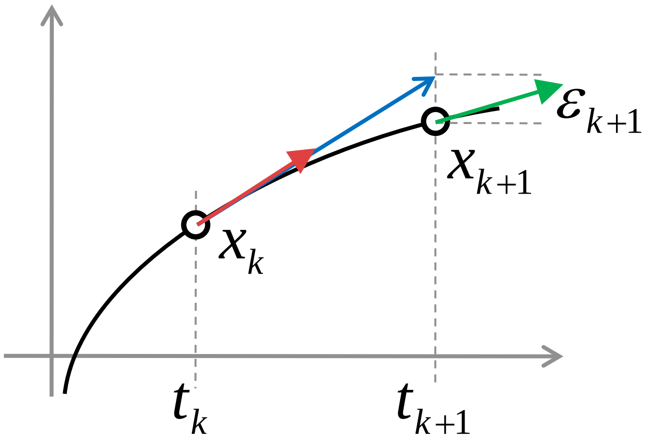
Numerical integration

Euler: step h in time along $\dot{x} = f(x,t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$

Trapezoidal: average the end points

$$\hat{x}_t(t_{k+1}) = x(t_k) + \frac{\boxed{\dot{x}(t_{k+1})} + \boxed{\dot{x}(t_k)}}{2}h_k$$



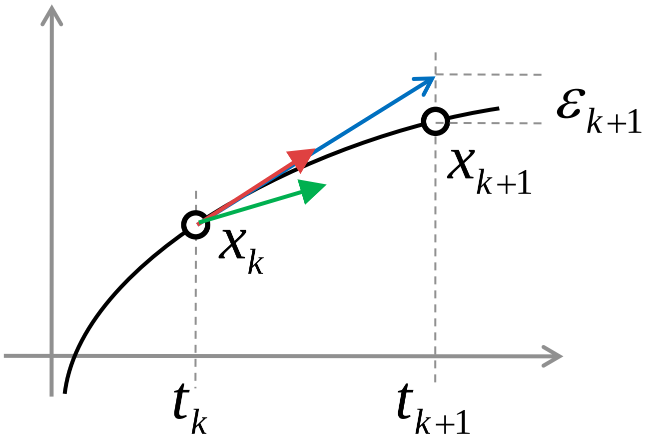
Numerical integration

Euler: step h in time along $\dot{x} = f(x,t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$

Trapezoidal: average the end points

$$\hat{x}_t(t_{k+1}) = x(t_k) + \frac{\boxed{\dot{x}(t_{k+1})} + \boxed{\dot{x}(t_k)}}{2}h_k$$



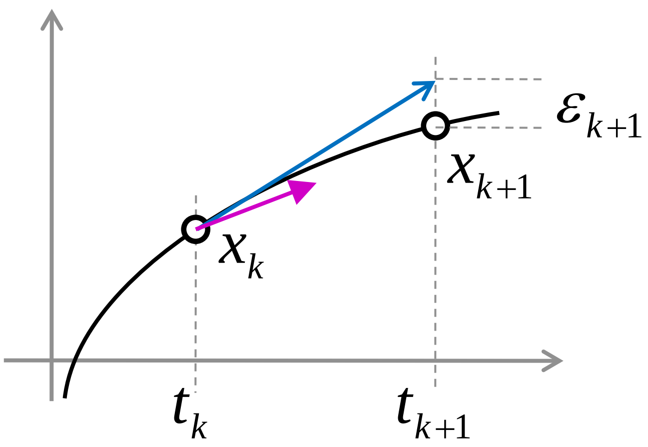
Numerical integration

Euler: step h in time along $\dot{x} = f(x,t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$

Trapezoidal: average the end points

$$\hat{x}_t(t_{k+1}) = x(t_k) + \boxed{\frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2}}h_k$$



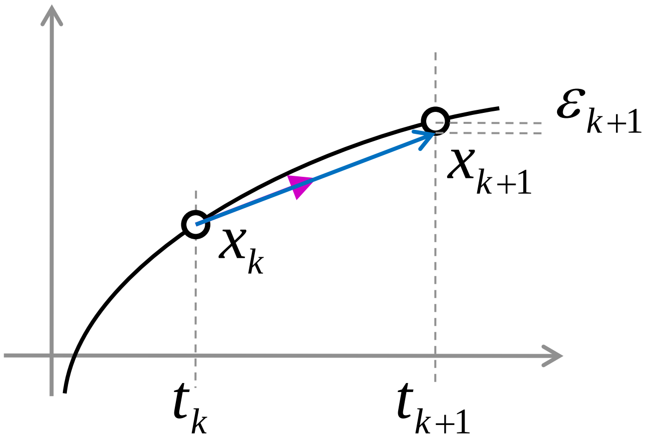
Numerical integration

Euler: step h in time along $\dot{x} = f(x,t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$

Trapezoidal: average the end points

$$\hat{x}_t(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2} h_k$$



Taylor series expansion for error analysis

$$x(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_k)}{1!} h_k + \frac{\ddot{x}(t_k)}{2!} h_k^2 + O(h_k^3)$$

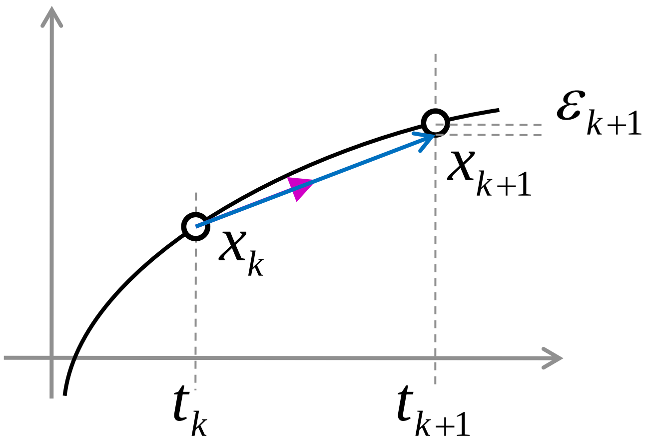
Numerical integration

Euler: step h in time along $\dot{x} = f(x,t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$

Trapezoidal: average the end points

$$\hat{x}_t(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2} h_k$$



Taylor series expansion for error analysis

$$x(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_k)}{1!} h_k + \underbrace{\boxed{\frac{\ddot{x}(t_k)}{2!} h_k^2}}_{\varepsilon_e(t_{k+1})} + O(h_k^3)$$

When $x(t)$ changes little, h_k can be large!

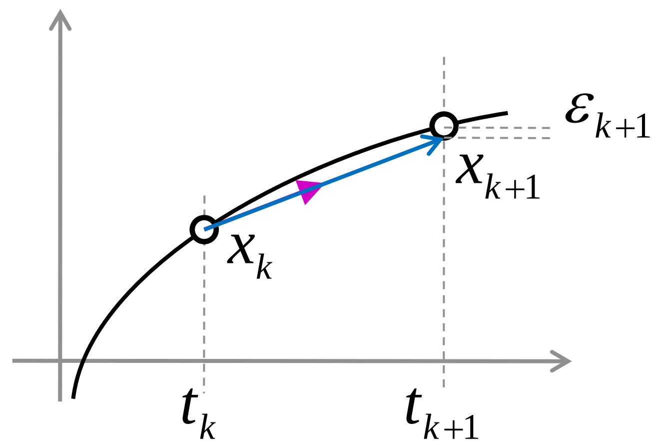
Numerical integration

Euler: step h in time along $\dot{x} = f(x,t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$

Trapezoidal: average the end points

$$\hat{x}_t(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2}h_k$$



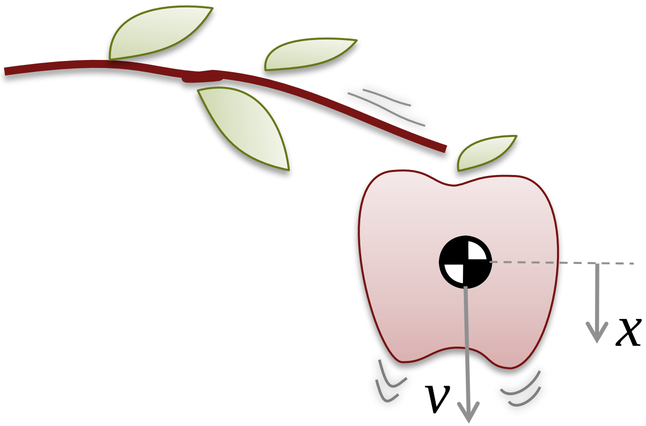
Taylor series expansion for error analysis

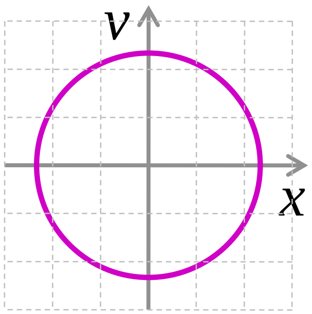
$$x(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_k)}{1!}h_k + \underbrace{\boxed{\frac{\ddot{x}(t_k)}{2!}h_k^2}}_{\varepsilon_e(t_{k+1})} + \underbrace{\boxed{O(h_k^3)}}_{\varepsilon_t(t_{k+1})}$$

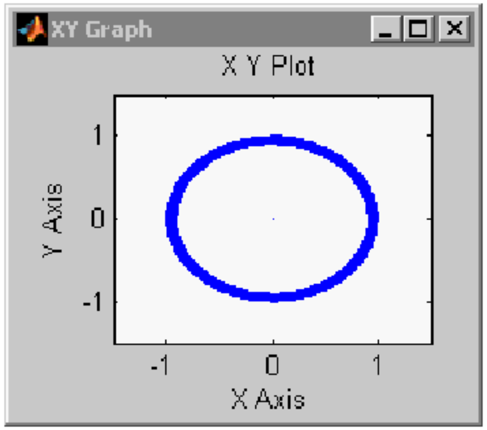
Change step size based on estimate: $\hat{x}_e(t_{k+1}) - \hat{x}_t(t_{k+1}) \approx \frac{\ddot{x}(t_k)}{2!}h_k^2$

Sophisticated solver … ?

- Let’s compute a solution to the ideal oscillator



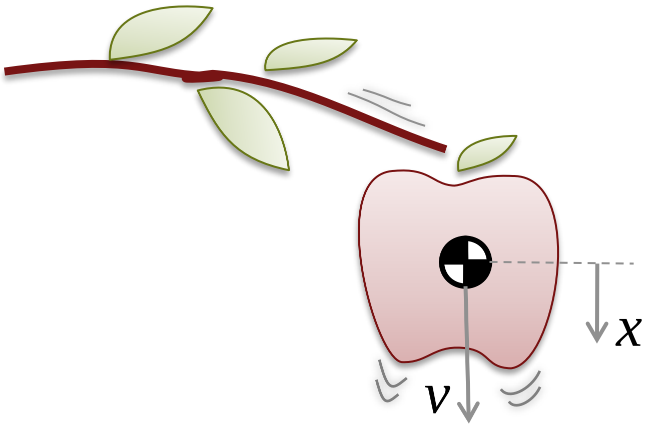


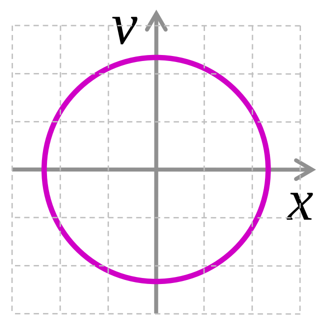


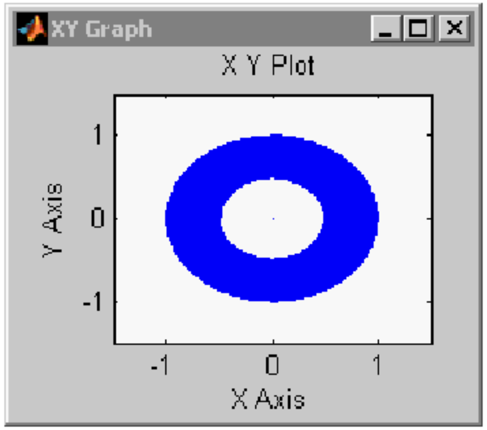
- We can make the error small … but only locally!

Sophisticated solver … ?

- Let’s compute a solution to the ideal oscillator







- We can make the error small … but only locally!
- It accumulates for ‘long time’ behavior
- *So, … how come the JSF flies?!*

Engineering an embedded system

physical

theoretical

computational

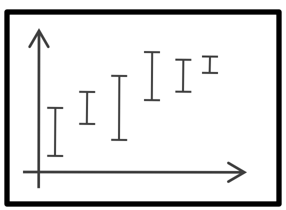
$$F(t) = m\frac{dv(t)}{dt}$$

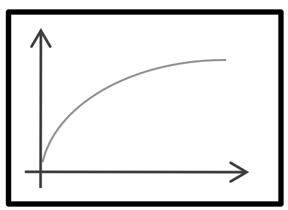
```
void main () {
  int i;
}
```

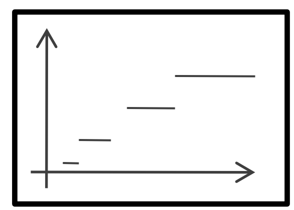
validate

$[\![\;]\!]$

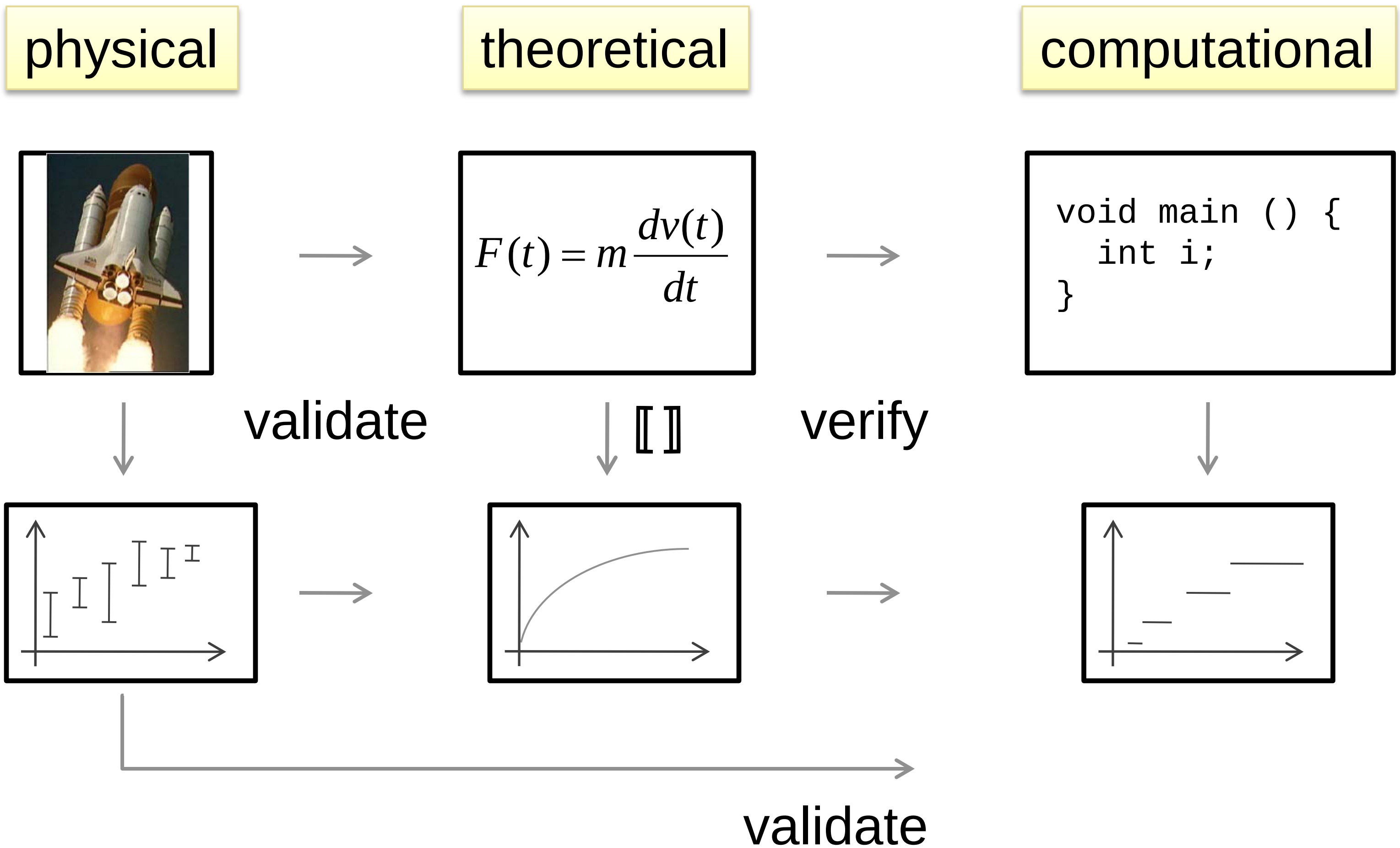
verify



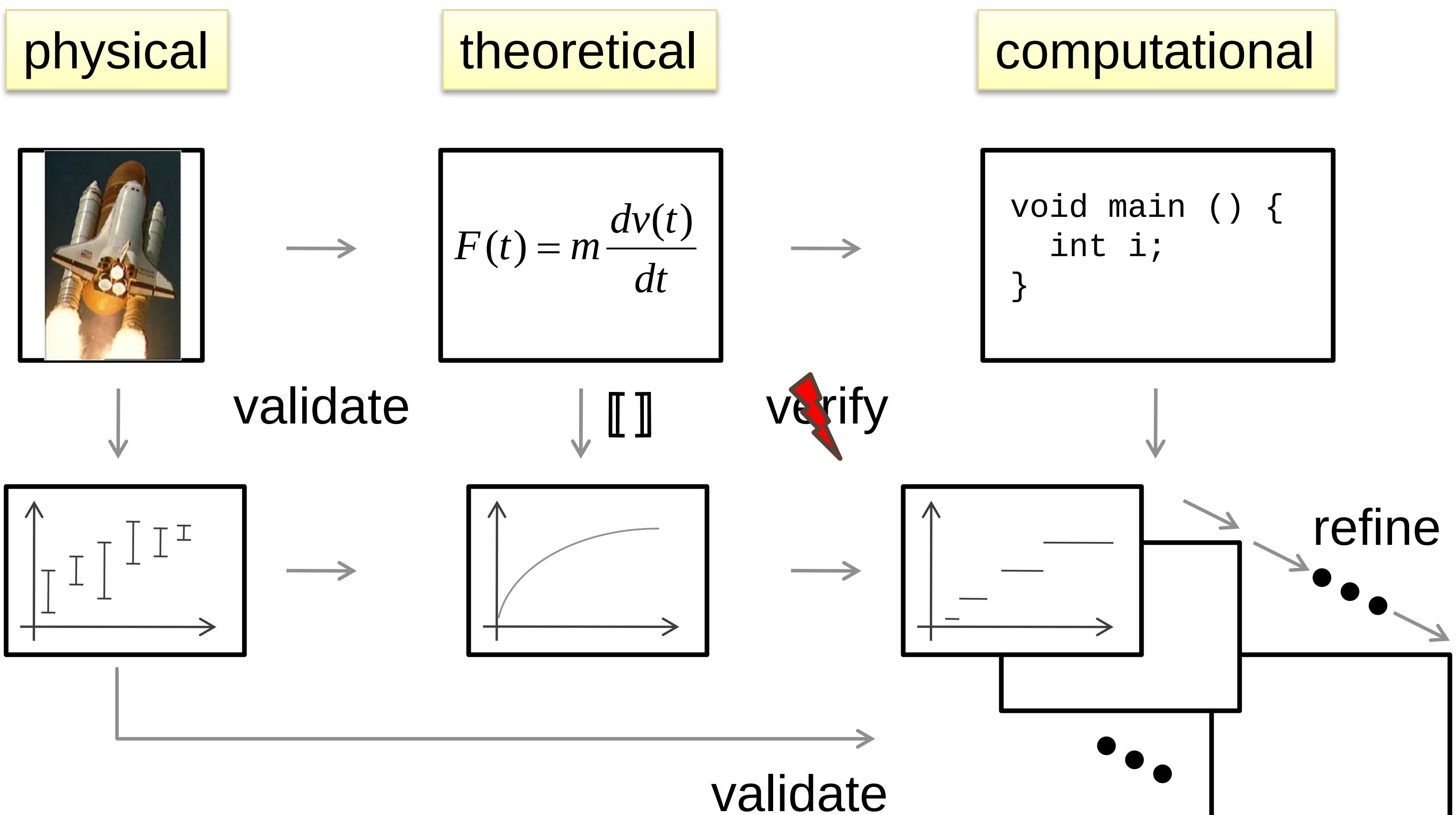




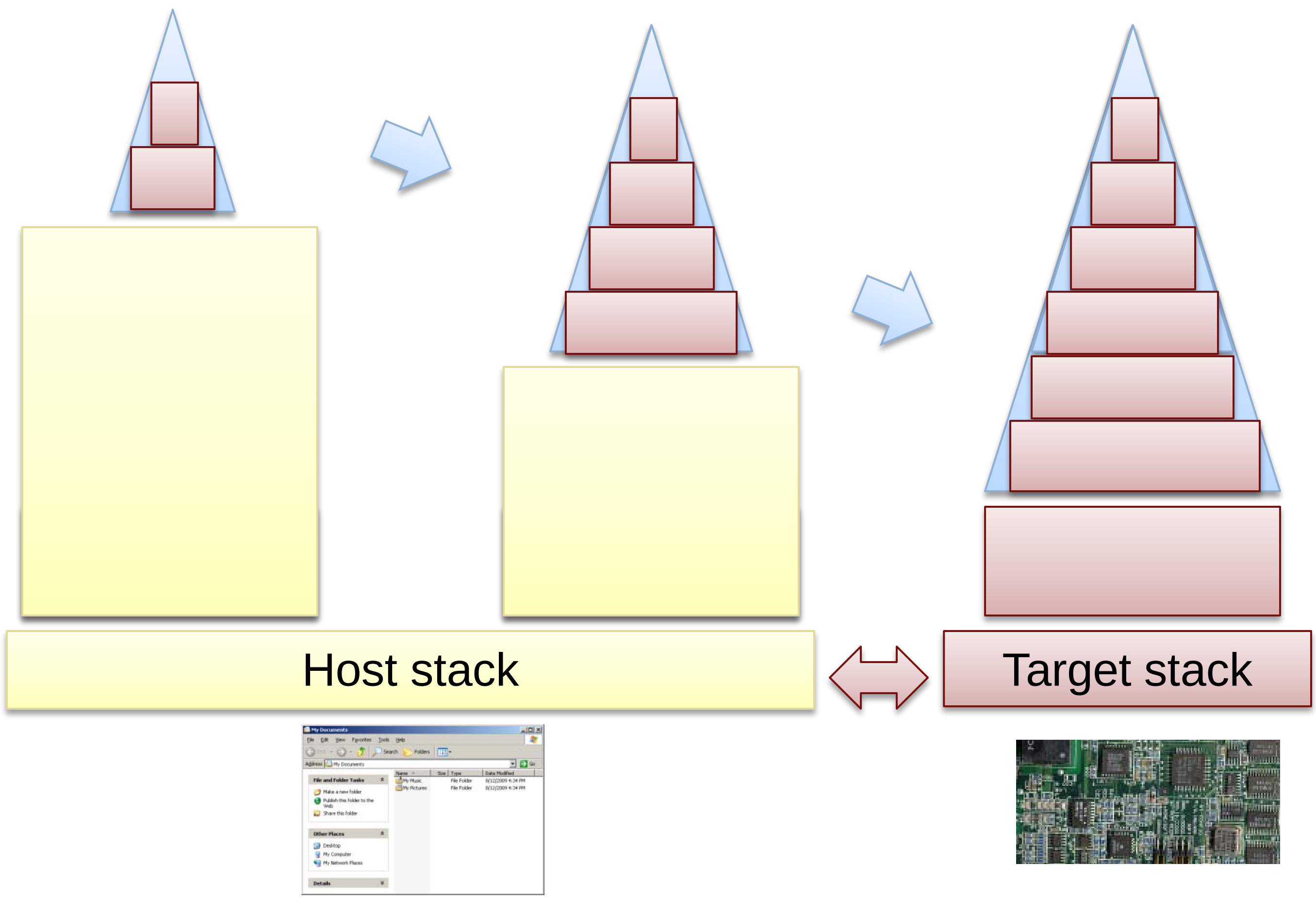
Engineering an embedded system



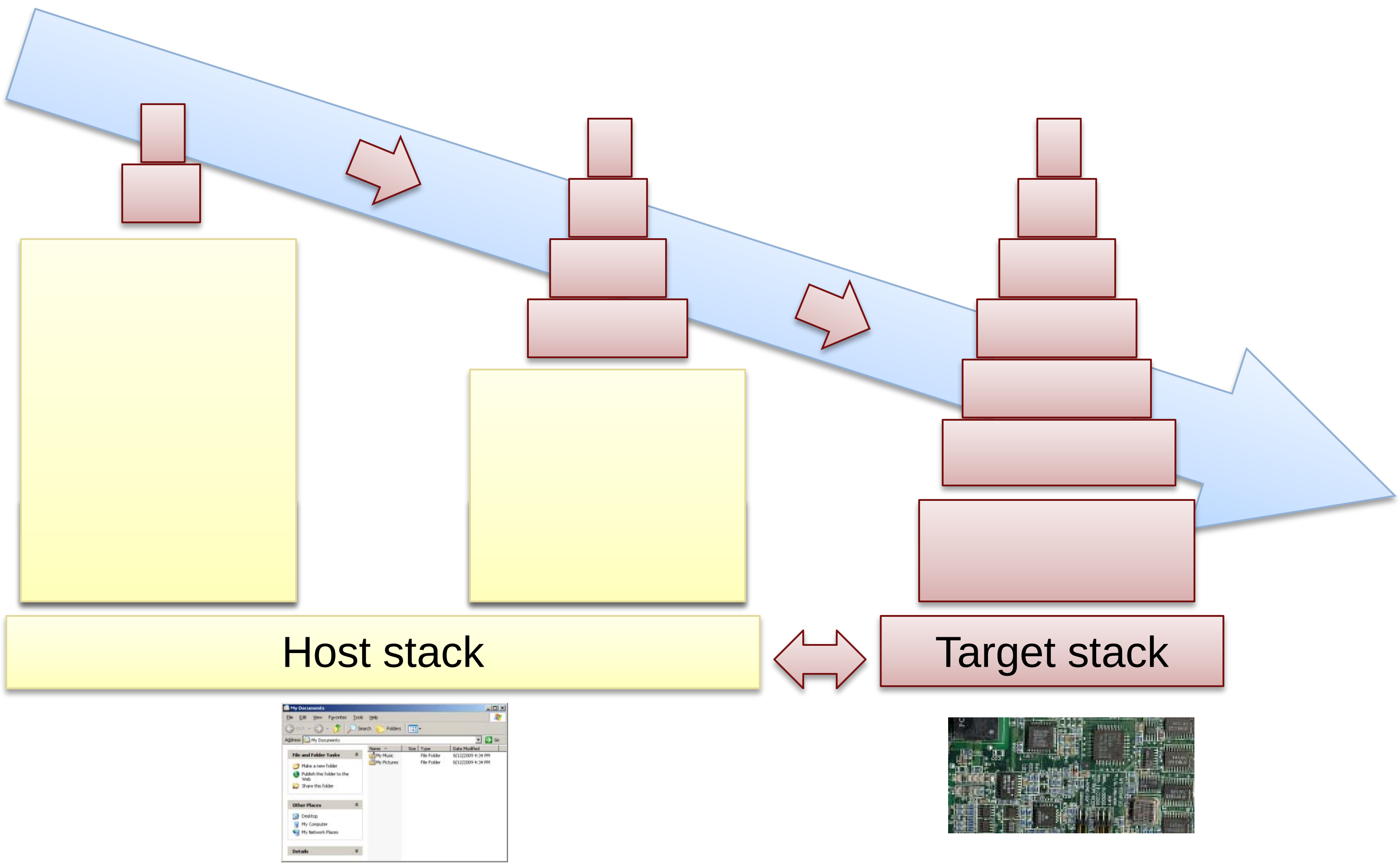
Engineering an embedded system



Create executable models in all phases



Make the computational approximation the primary design deliverable—the real thing!



So that gets the job done … but …

- More than 50% of the modeling effort is in verification, validation, and testing!
- Semantics of models is buried in the execution engine
- Engine code base is extensive and complex
  - Interaction of approximations
  - Interaction and interfacing of different formalisms
- How can we mature the field?
- Model the semantics of the execution engine!

Agenda

- Outline
- Model-Based Design
- Problem statement
- A solution approach
- Outlook

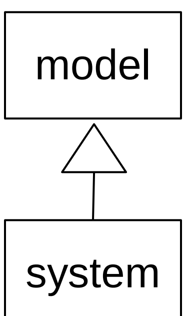
So, what is a model anyway?

Jean Bézivin

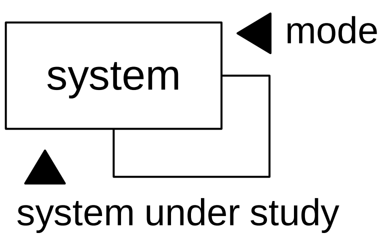
Jean-Marie Favre

Pieter J. Mosterman

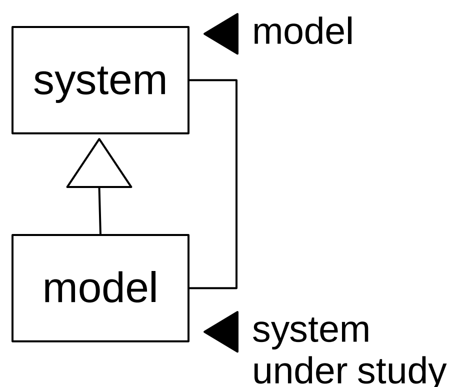
"Everything is a model"



"Nothing is a model"



"Nothing is not a model"



So, what is a model anyway?

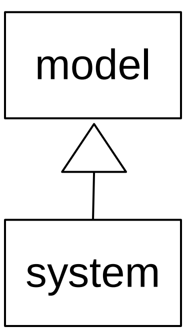
Jean Bézivin

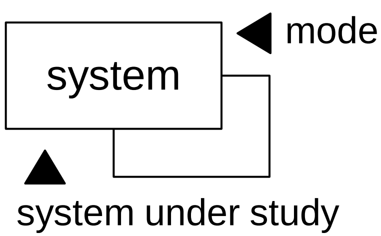
Jean-Marie Favre

Pieter J. Mosterman

Hans Vangheluwe

"Everything is a model" "Nothing is a model" "Nothing is not a model" "Model everything"





Computer Automated Multiparadigm Modeling
(CAMPaM)

In collaboration with Hans Vangheluwe, McGill University

So, what is a model anyway?

Jean Bézivin

“Everything is a model”

Jean-Marie Favre

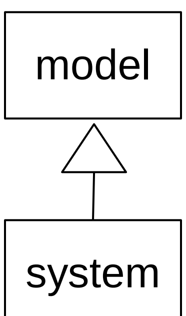
“Nothing is a model”

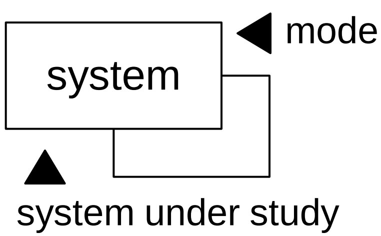
Pieter J. Mosterman

“Nothing is not a model”

Hans Vangheluwe

“Model everything”



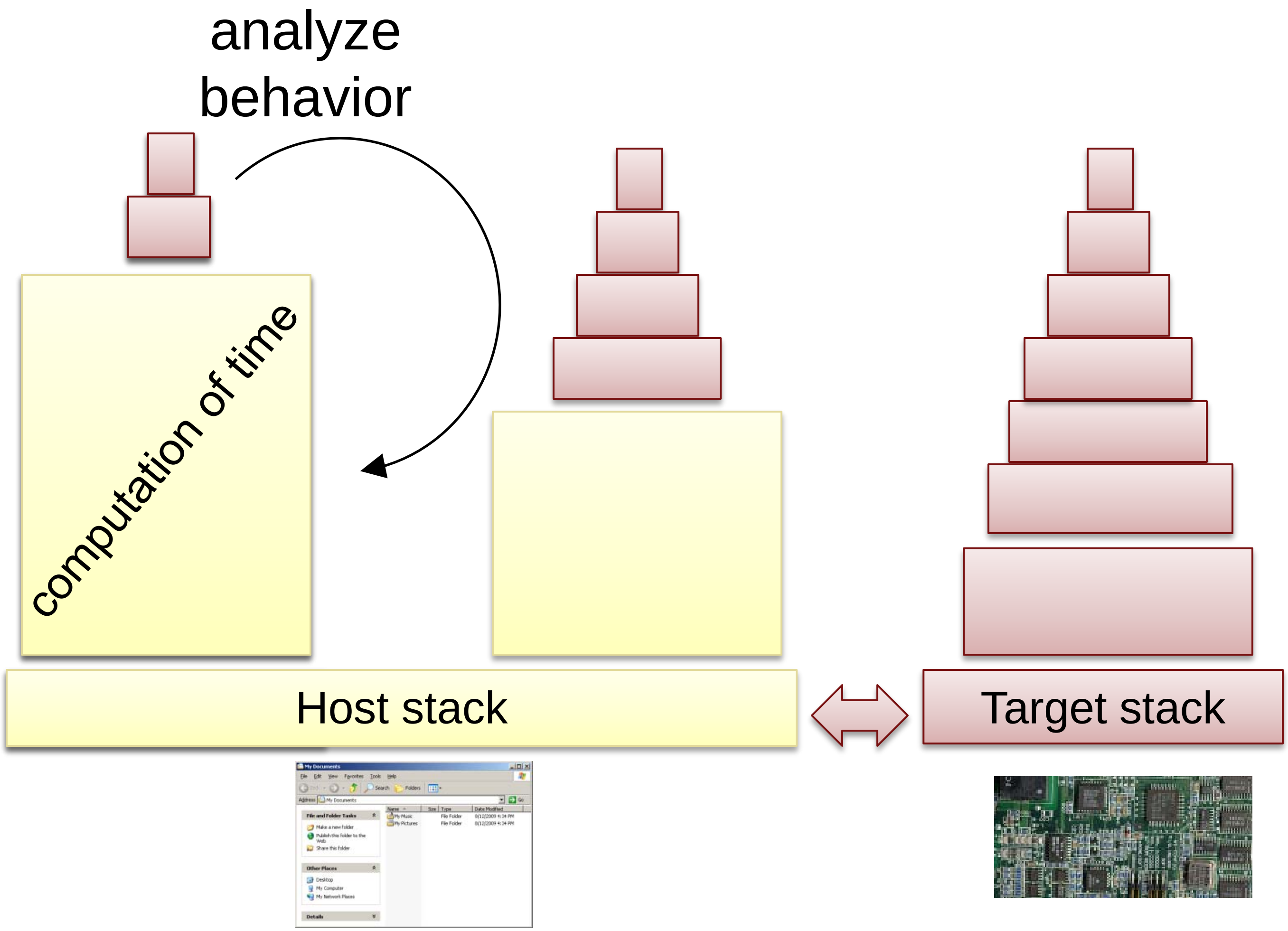


With the most appropriate formalism

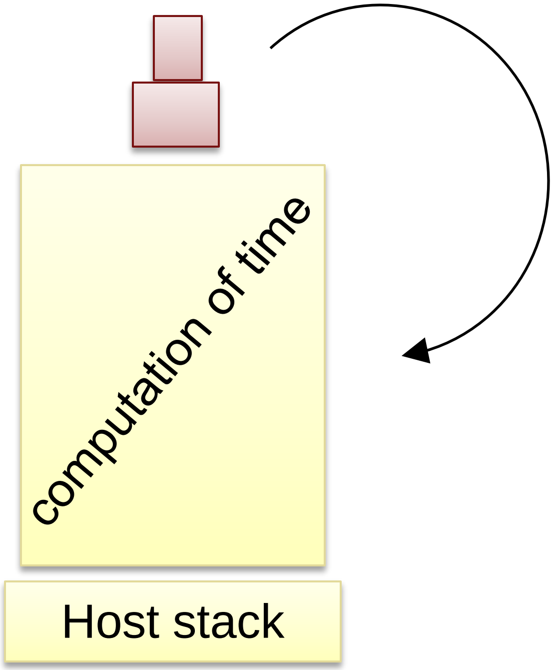
At the most appropriate level of abstraction

In collaboration with Hans Vangheluwe, McGill University

Modeling the execution engine



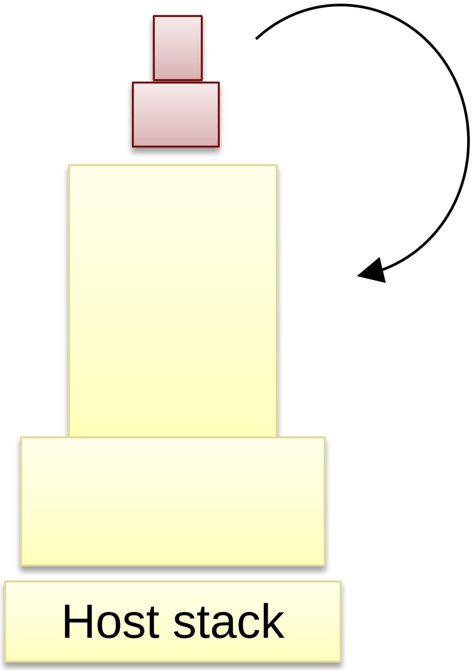
Modeling the execution engine



Create abstractions in the simulation stack …

specification

implementation

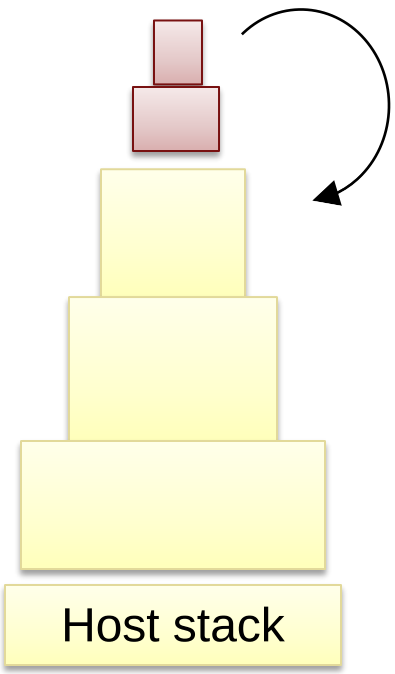


Create abstractions in the simulation stack …

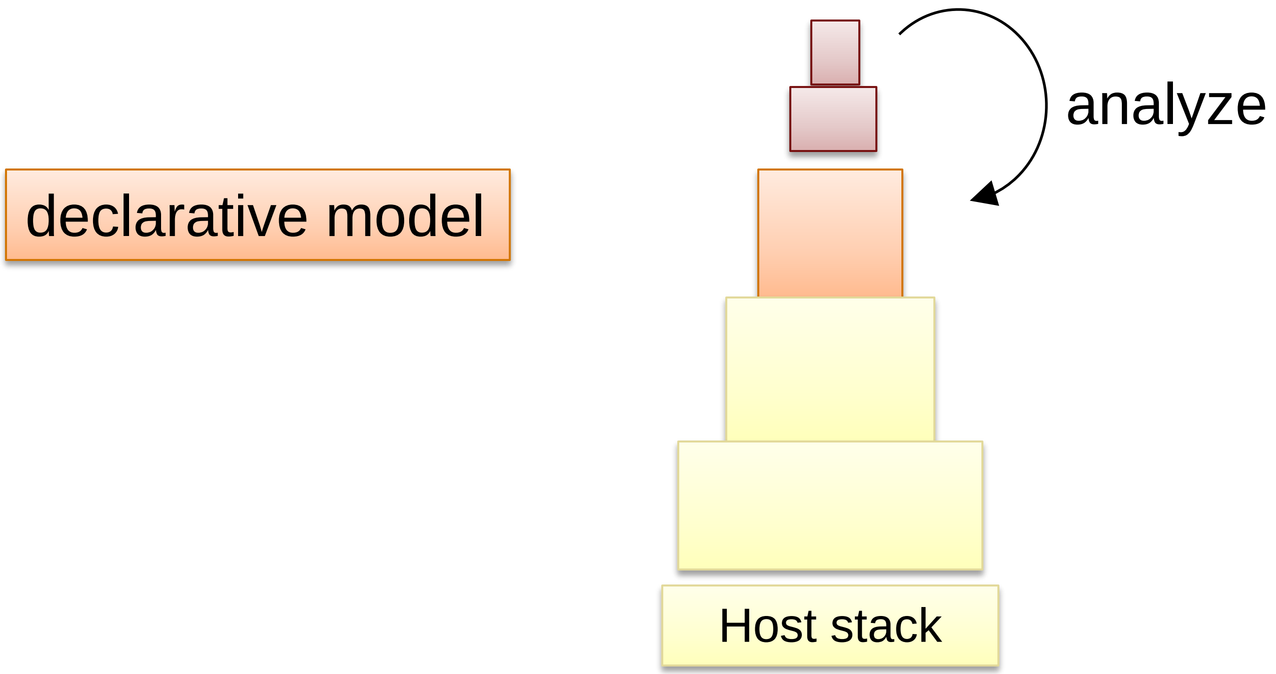
declarative model

imperative model

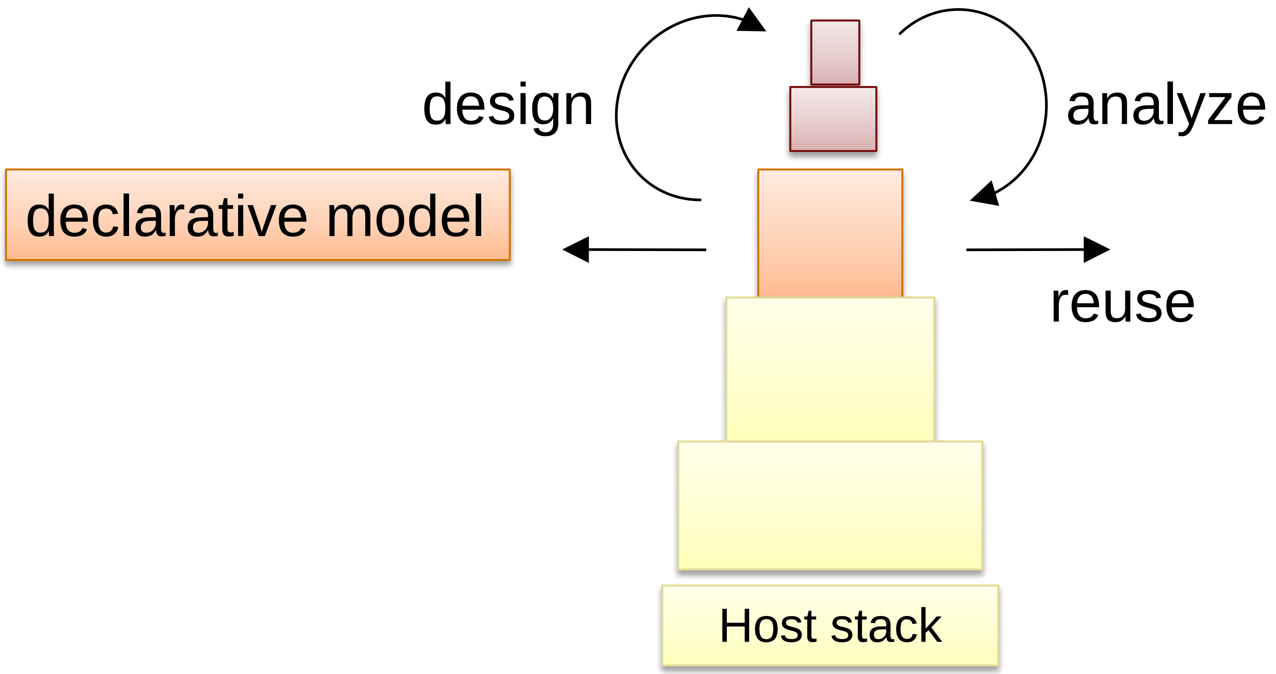
implementation



Analyze as little as possible



Further facilitate design, reuse, …



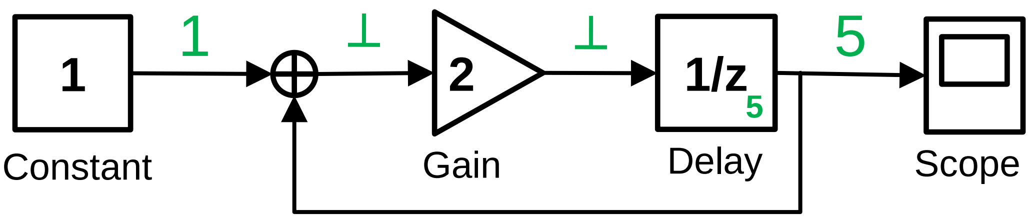
A declarative formalism with fix-point semantics

A LATTICE-THEORETICAL FIXPOINT THEOREM
AND ITS APPLICATIONS

ALFRED TARSKI

Pacific J. Math. 5 (1955), 285 - 309

– Repeated application of a monotonically increasing partial function converges to a fixed point



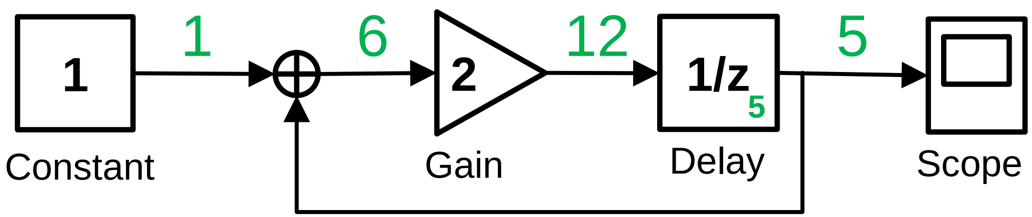
A declarative formalism with fix-point semantics

A LATTICE-THEORETICAL FIXPOINT THEOREM
AND ITS APPLICATIONS

ALFRED TARSKI

Pacific J. Math. 5 (1955), 285 - 309

– Repeated application of a monotonically increasing partial function converges to a fixed point



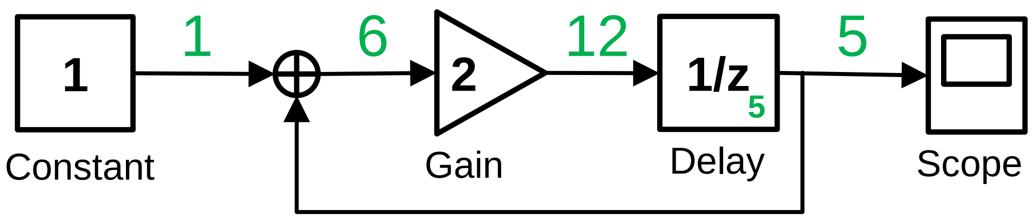
A declarative formalism with fix-point semantics

A LATTICE-THEORETICAL FIXPOINT THEOREM
AND ITS APPLICATIONS

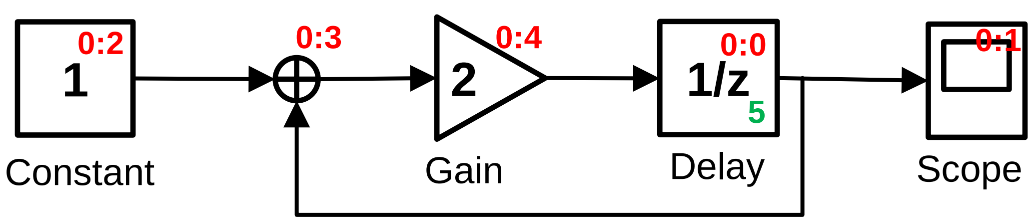
ALFRED TARSKI

Pacific J. Math. 5 (1955), 285-309

– Repeated application of a monotonically increasing partial function converges to a fixed point



– One implementation is a data dependency schedule



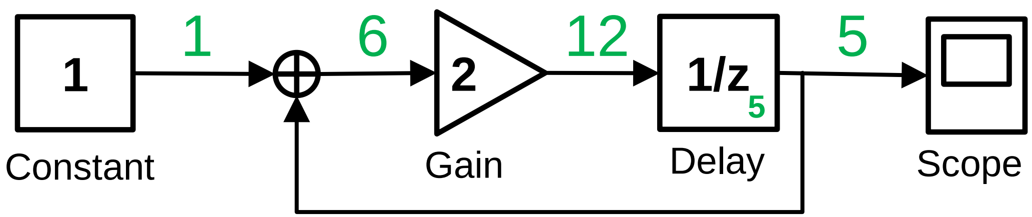
A declarative formalism with fix-point semantics


A LATTICE-THEORETICAL FIXPOINT THEOREM AND ITS APPLICATIONS

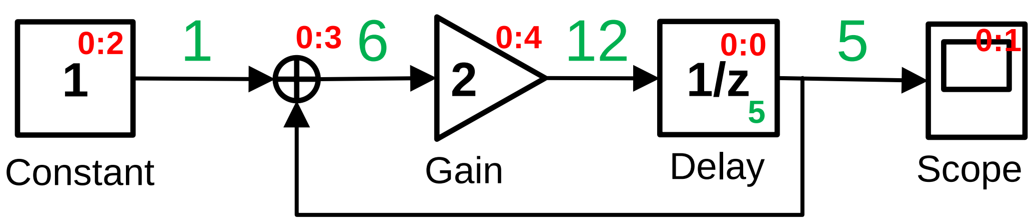
ALFRED TARSKI

Pacific J. Math. 5 (1955), 285 - 309

– Repeated application of a monotonically increasing partial function converges to a fixed point

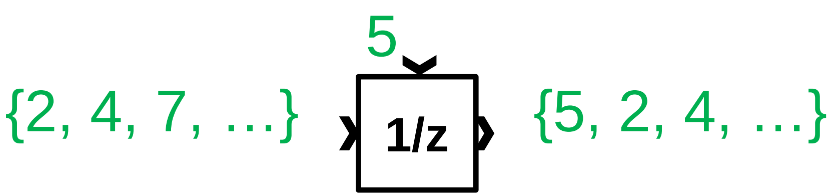


– One implementation is a data dependency schedule



Dynamic systems evolve over time

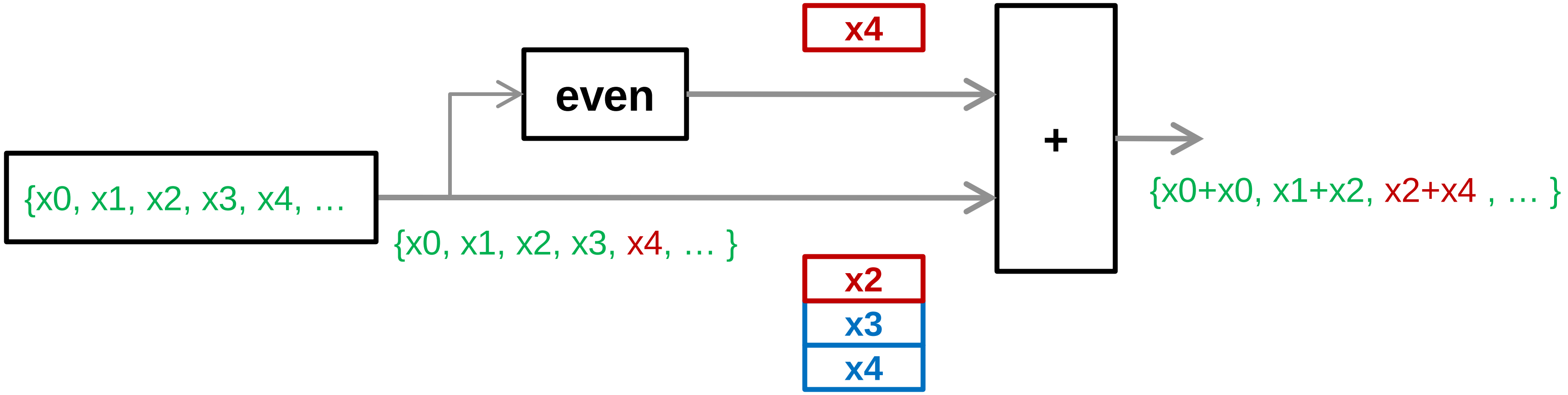
- Sequences of fix-point evaluations
- Define input and output signals as (potentially infinite) streams of values
  - Stream(Type) = Type : Stream(Type)
- Delay as a function application
  - Delay x0 u = x0 : u



  - A variable has a ‘clock’ that encodes its sample time

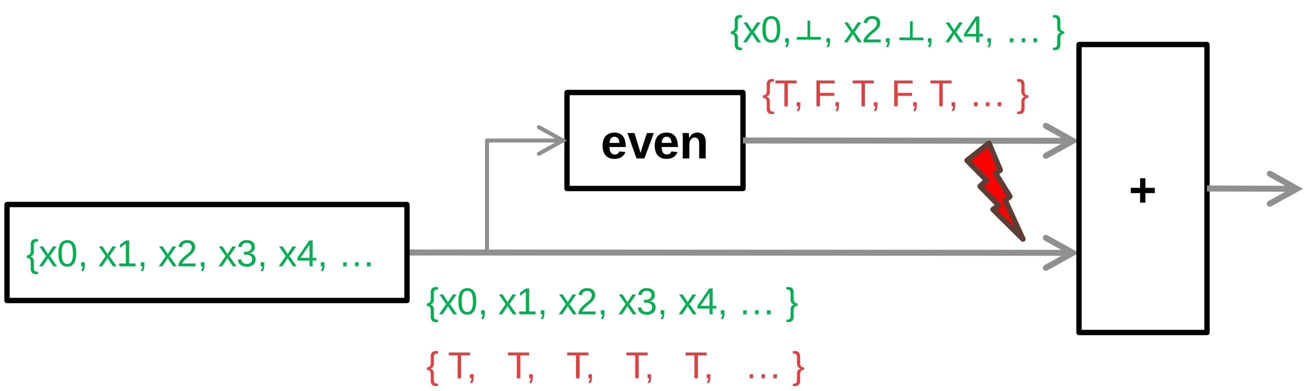
Multiple rates; a potential problem …

- Streams are only practical if we can limit the stream entries being accessed
- Not this:



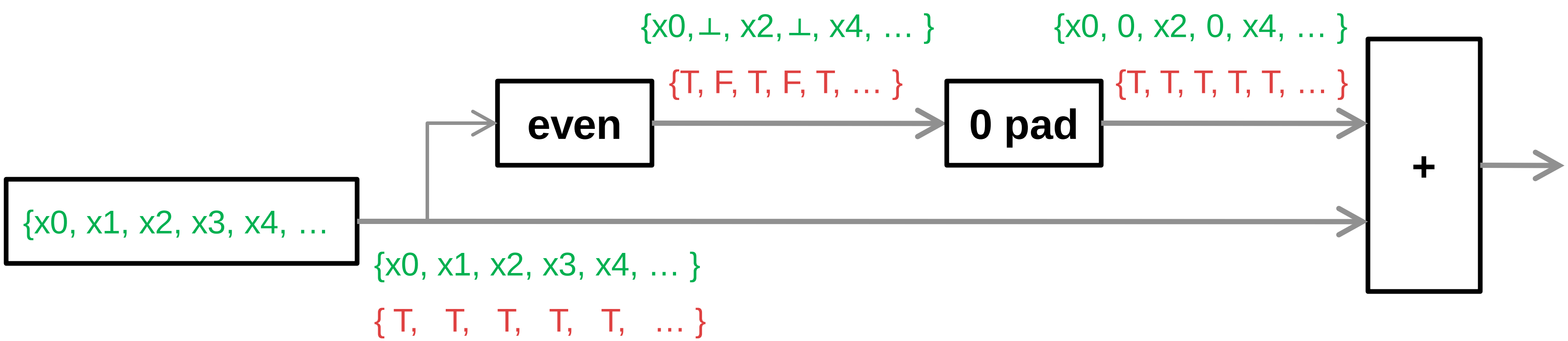
Clock calculus to detect

- Require compatible clocks: the synchronous assumption
- Match against base clock

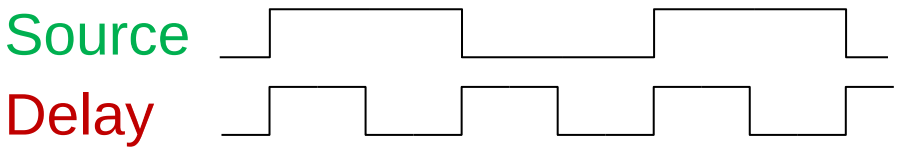


Clock calculus to detect

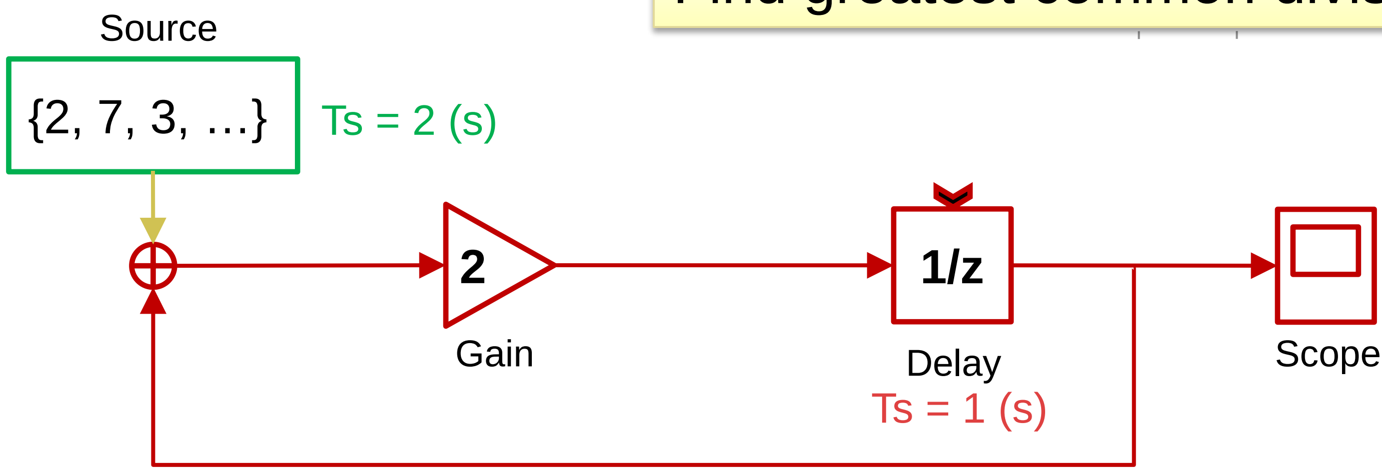
- Require compatible clocks: the synchronous assumption
- Match against base clock



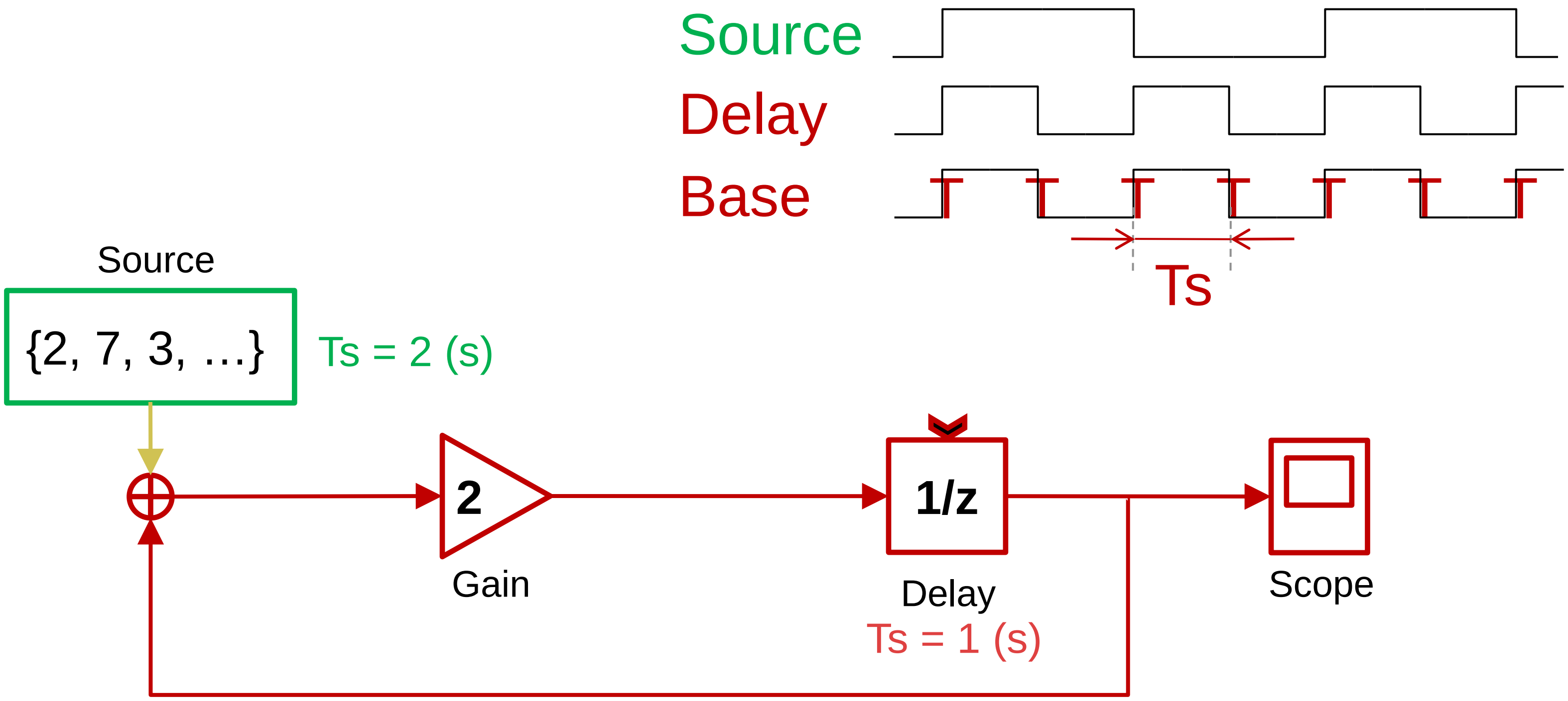
A multi-rate system example



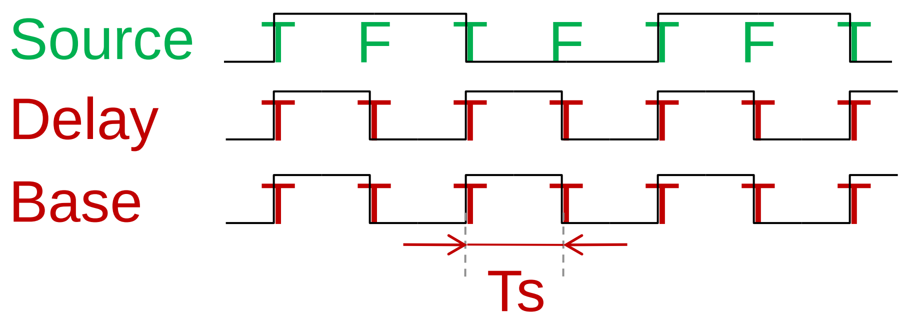
Find greatest common divisor (Ts)!

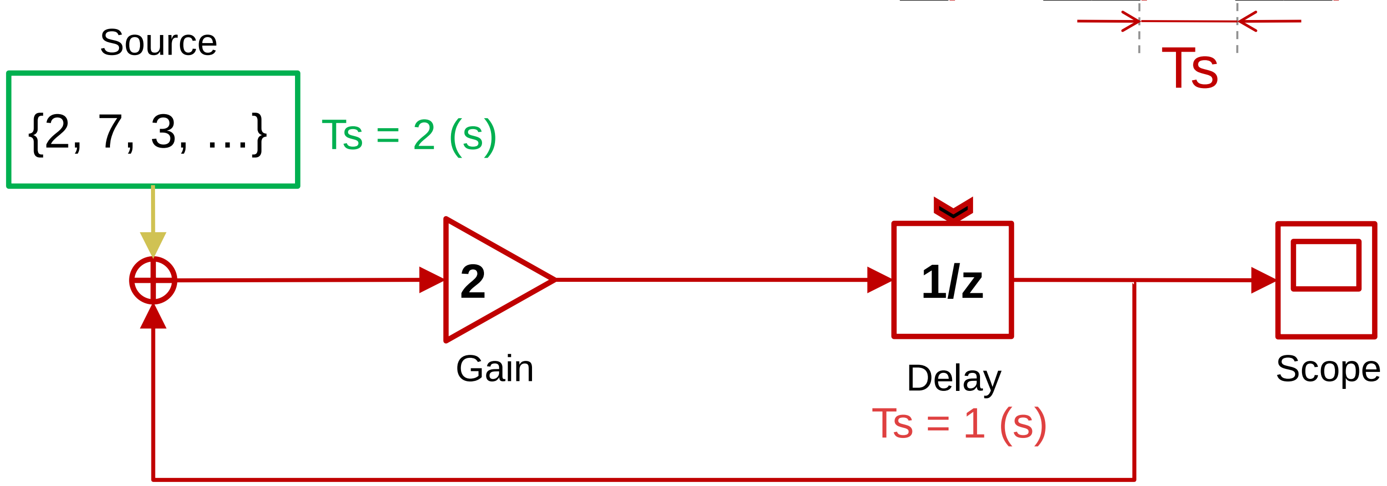


A multi-rate system example



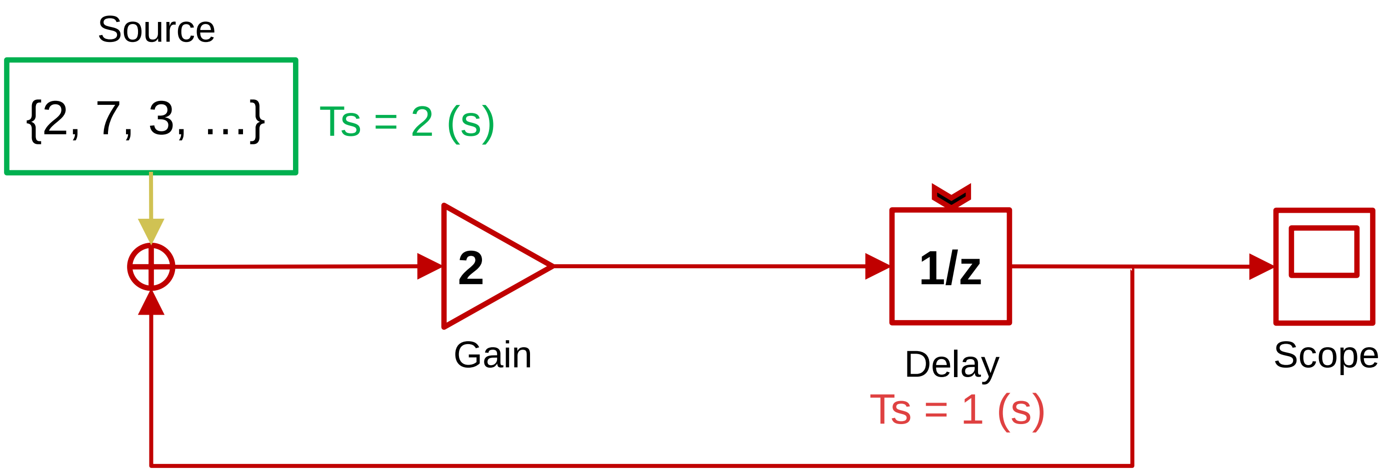
A multi-rate system example





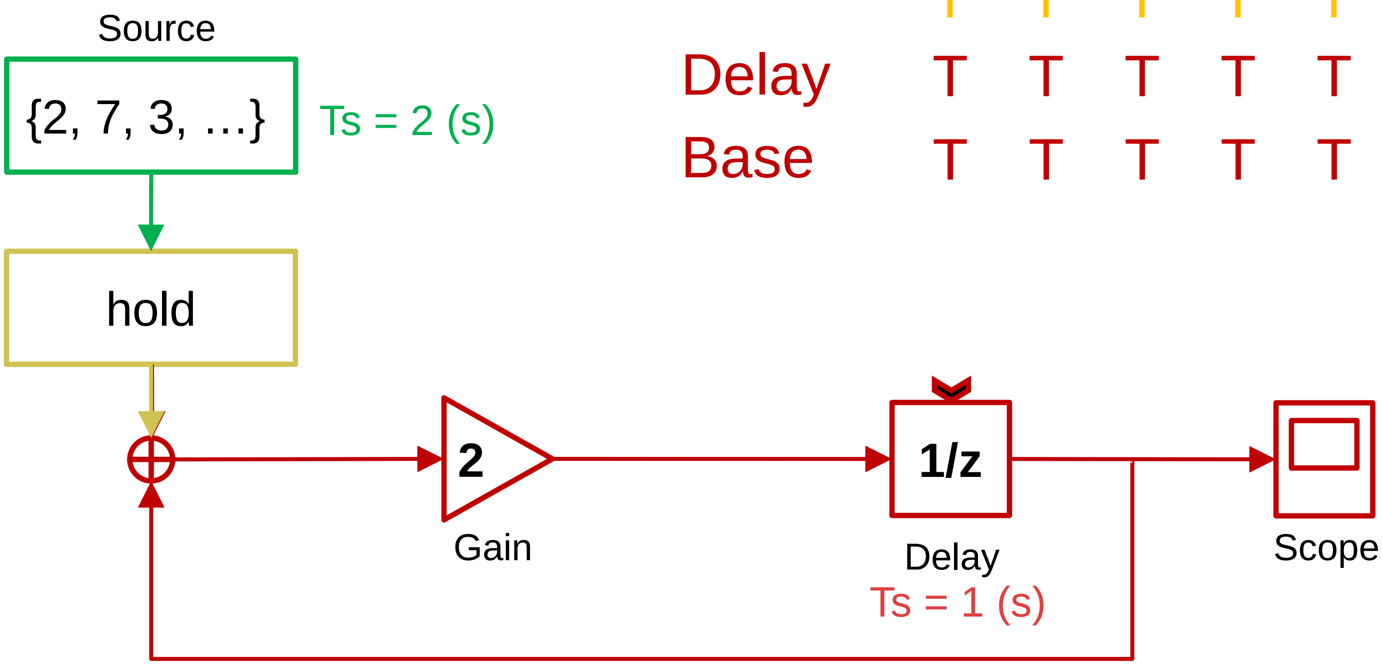
A multi-rate system example

Source	T	F	T	F	T	F	T
Delay	T	T	T	T	T	T	T
Base	T	T	T	T	T	T	T



A multi-rate system example

Source	T	F	T	F	T	F	T
RT	T	F	T	F	T	F	T
	T	T	T	T	T	T	T
Delay	T	T	T	T	T	T	T
Base	T	T	T	T	T	T	T

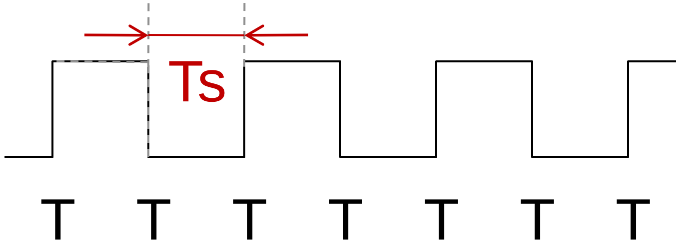


Can we use this framework to define a variable-step solver?

- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations

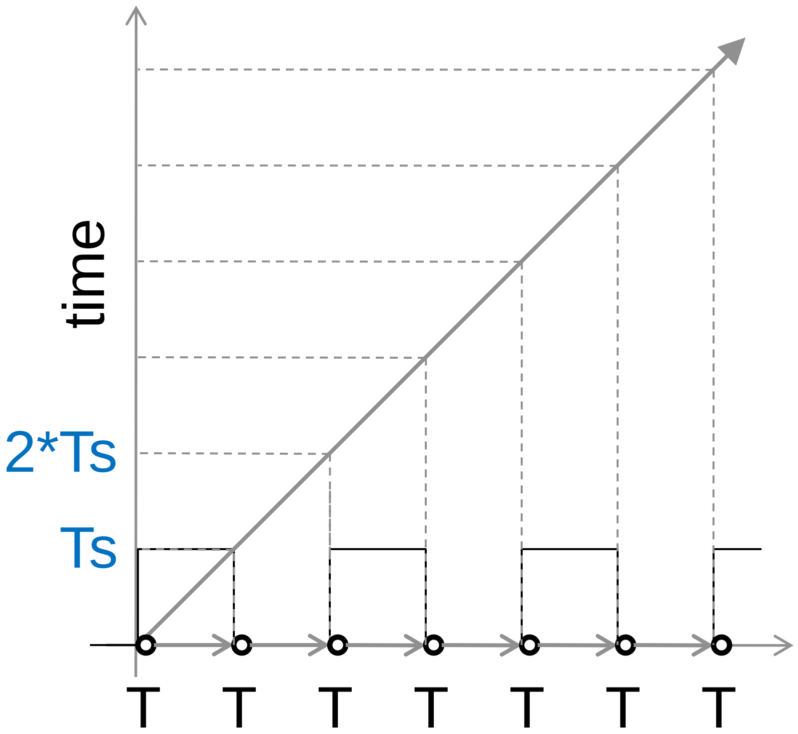
Can we use this framework to define a variable-step solver?

- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations



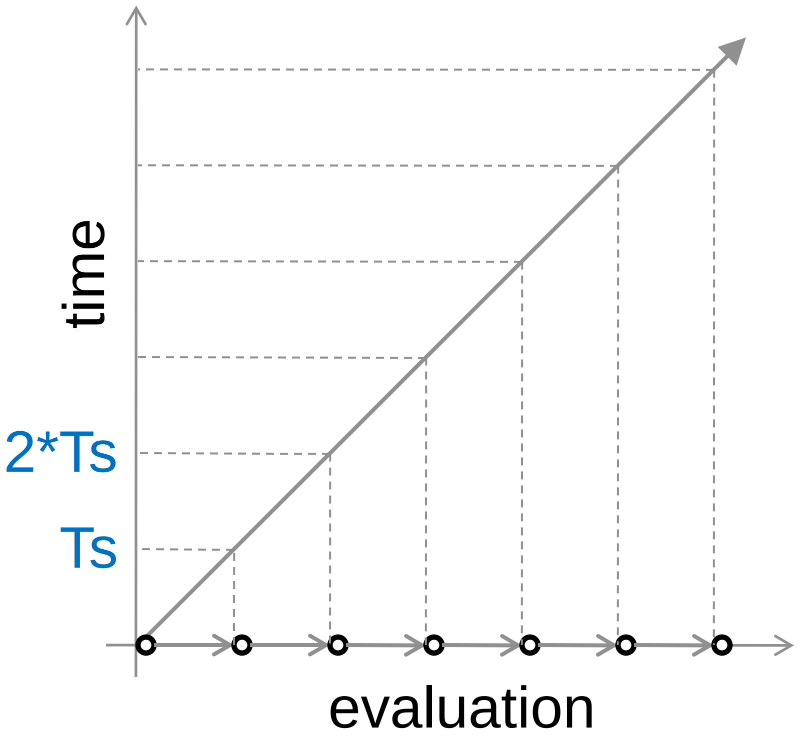
Can we use this framework to define a variable-step solver?

- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations



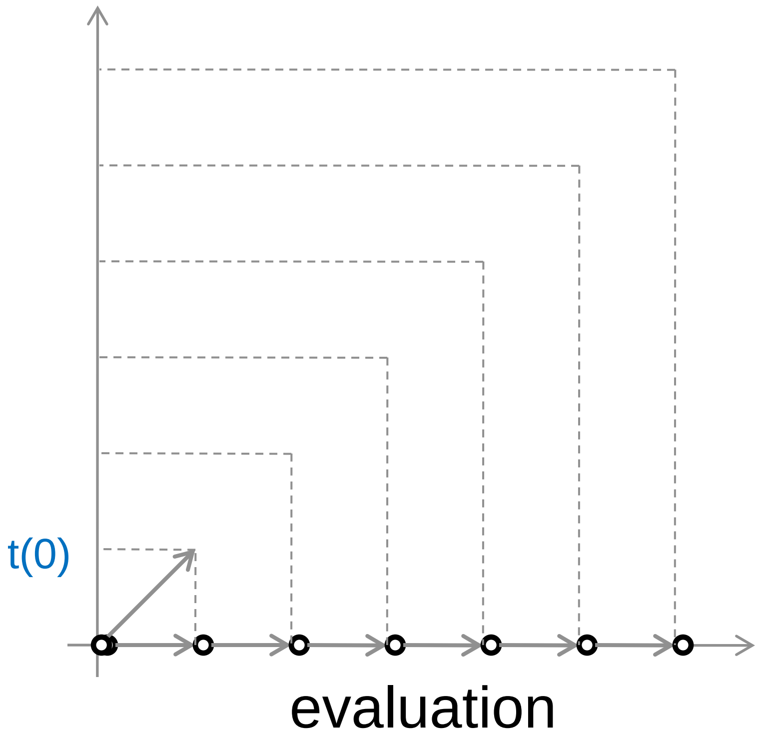
Can we use this framework to define a variable-step solver?

- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations



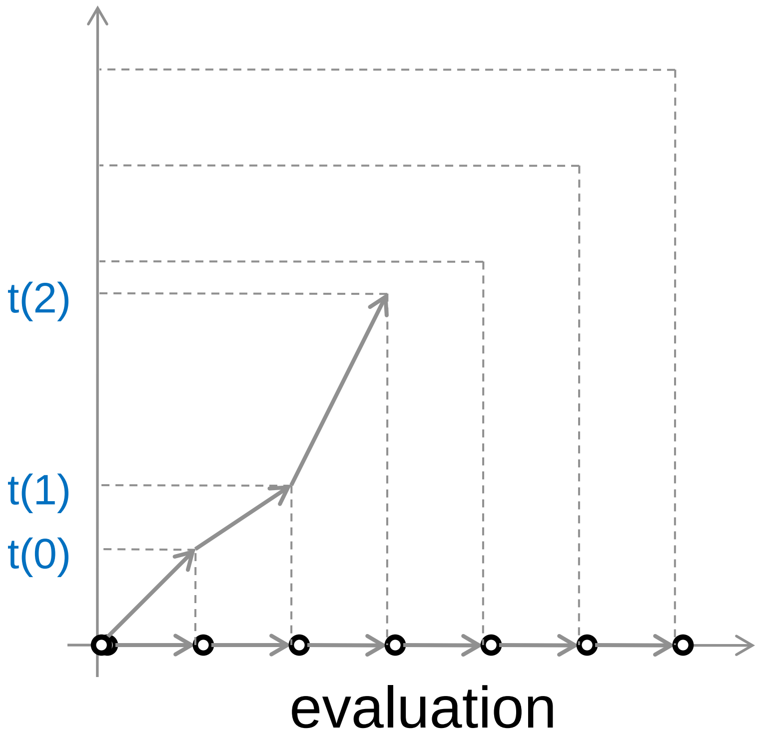
Can we use this framework to define a variable-step solver?

- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations



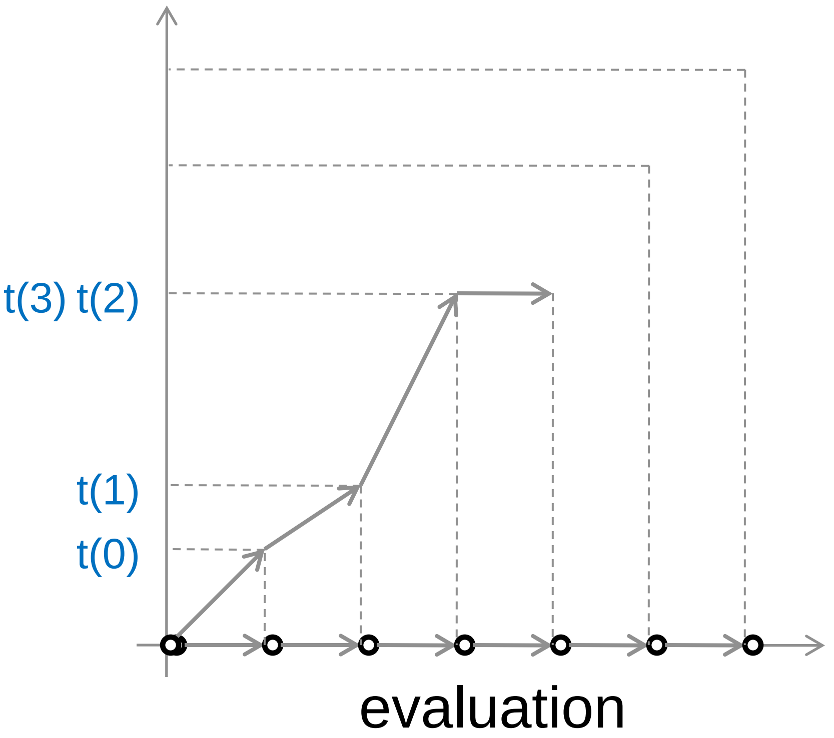
Can we use this framework to define a variable-step solver?

- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations
  - Step is variable



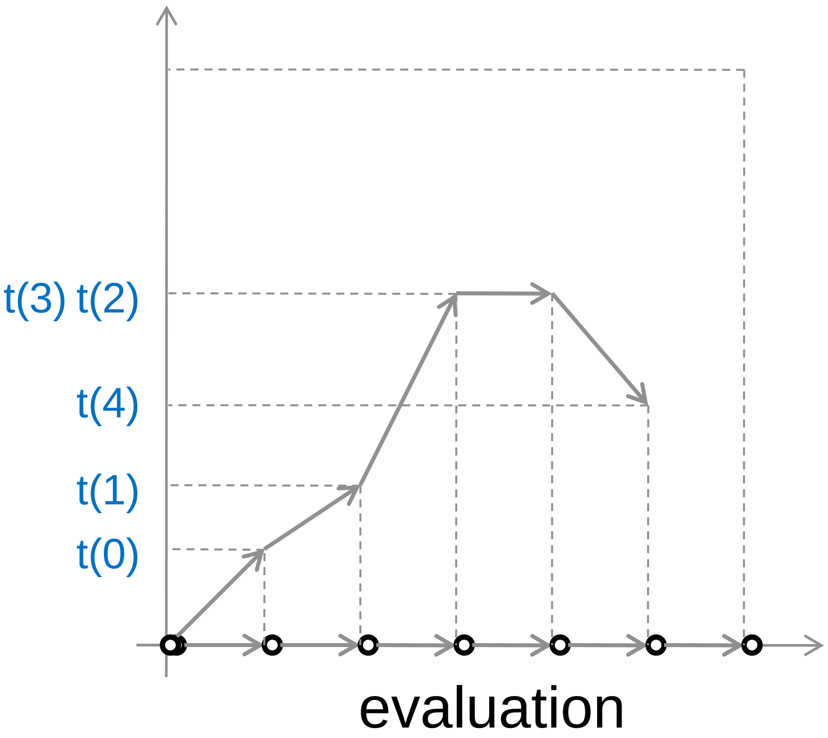
Can we use this framework to define a variable-step solver?

- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations
  - Step is variable
  - Step may be 0



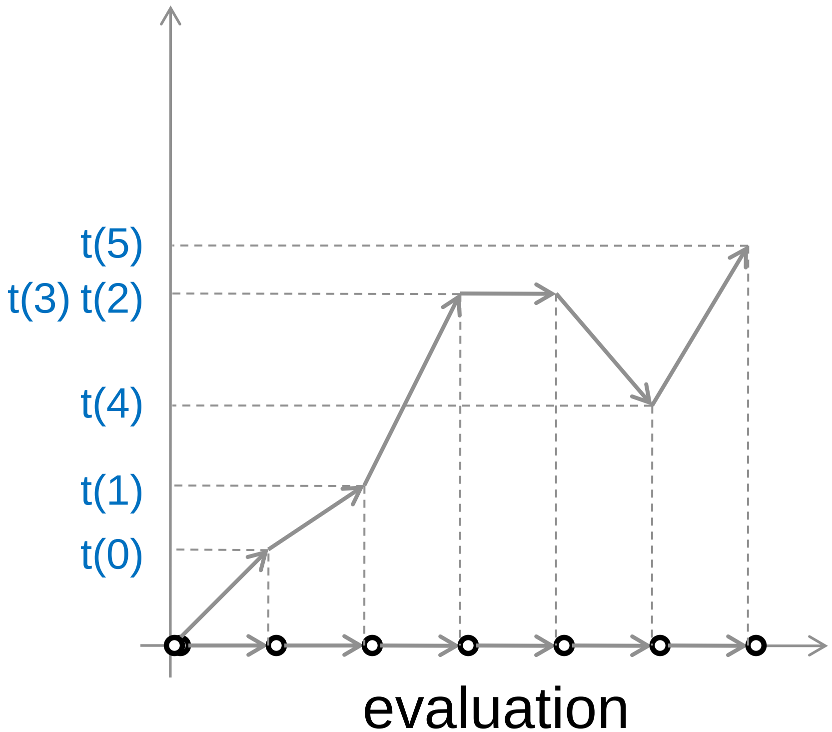
Can we use this framework to define a variable-step solver?

- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations
  - Step is variable
  - Step may be 0
  - Step may be negative
    - Time may recede



Can we use this framework to define a variable-step solver?

- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations
  - Step is variable
  - Step may be 0
  - Step may be negative
    - Time may recede



A stream based functional solver

Euler integration

$$y_e(e) = \begin{cases} \sum_{i=1}^{e} u(i)h(i) & \text{if} \quad odd(e) \\ y_e(e-1) & otherwise \end{cases}$$

Trapezoidal integration

$$y_t(e) = \sum_{i=1}^{e} \frac{\left(u(i-1)+u(i)\right)h(i-1)}{2}$$

A stream based functional solver

Euler integration

$$y_e(e) = \begin{cases} \sum_{i=1}^{e} \underbrace{u(i)h(i)}_{\text{increment}} - \underbrace{u(i-2)h(i-2)\,p(i)}_{\text{previous increment}} & if \quad odd(e) \\ y_e(e-1) & otherwise \end{cases}$$

Trapezoidal integration

$$y_t(e) = \sum_{i=1}^{e} \underbrace{\frac{\big(u(i-1)+u(i)\big)h(i-1)}{2}}_{\text{increment}} - \underbrace{\frac{\big(u(i-3)+u(i-2)\big)h(i-3)}{2}\,p(i-1)}_{\text{previous increment}}$$

Error computation

$$d(e) = \frac{\big(u(e-3)+u(e-2)\big)h(e-3)}{2} - u(e-2)h(e-2) < tol$$

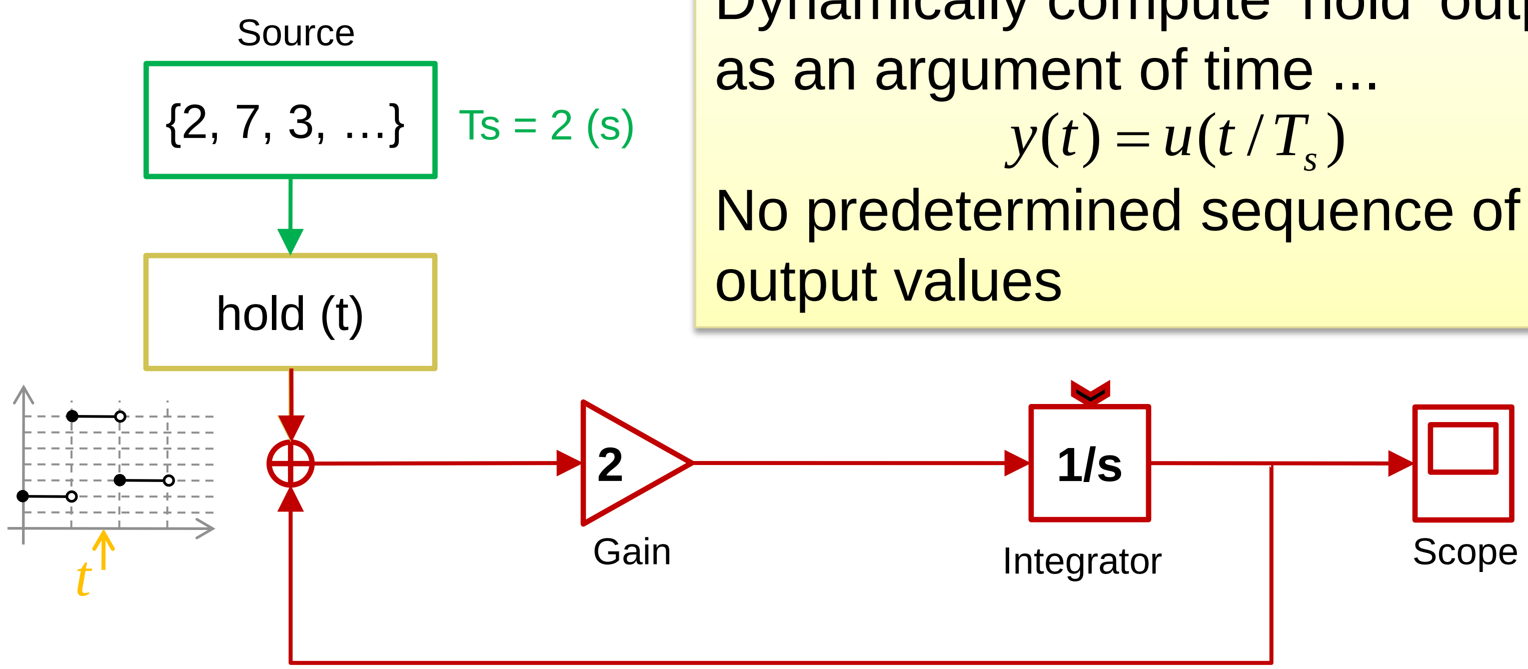
Rate transition a function of time

Now we can create a variable step solver inside 1/s that maps onto the synchronous paradigm

Dynamically compute ‘hold’ output as an argument of time ...

$$y(t) = u(t / T_s)$$

No predetermined sequence of output values

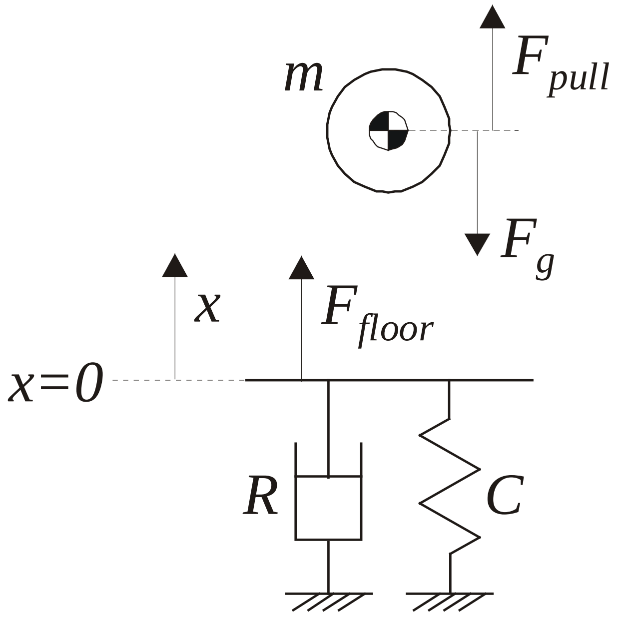


Unifying formalisms with different semantics

- Newton's Law and Hooke's Law
  - Differential equations as before
- Control behavior
  - Sampled data (periodic T_s=0.5)

$$F_{pull}(k) = \begin{cases} 20 & if & k = 0 \\ 10 & if & k = 1 \\ 0 & else & \end{cases}$$

- Contact behavior
  - Discontinuous changes …



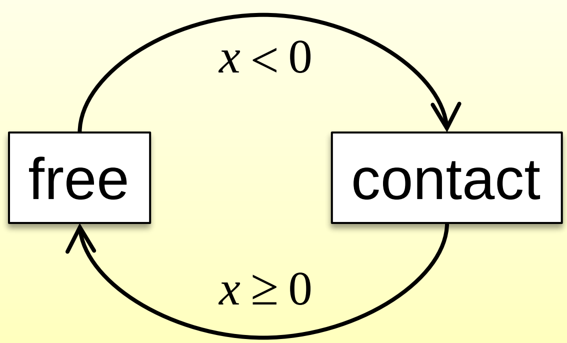
Modeling the contact behavior

- Simultaneous inequalities

$$F_{floor}(t) = \begin{cases} -\left(Rv(t) + \dfrac{x(t)}{C}\right) & \textit{if } x(t) < 0 \\ 0 & \textit{otherwise} \end{cases}$$

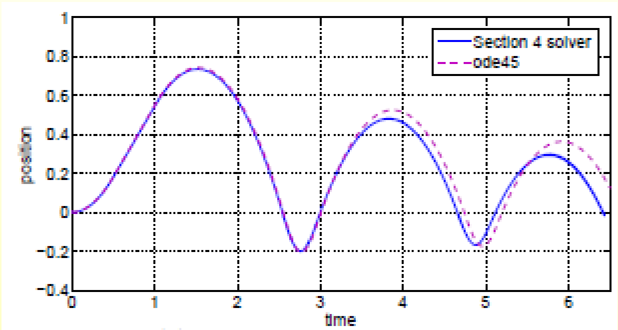
- Finite state machine

$$F_{floor}(t) = 0$$

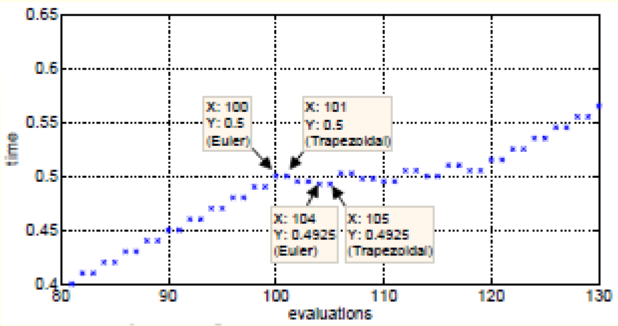


$$F_{floor}(t) = -\left(Rv(t) + \frac{x(t)}{C}\right)$$

Computational simulation



Position vs. time

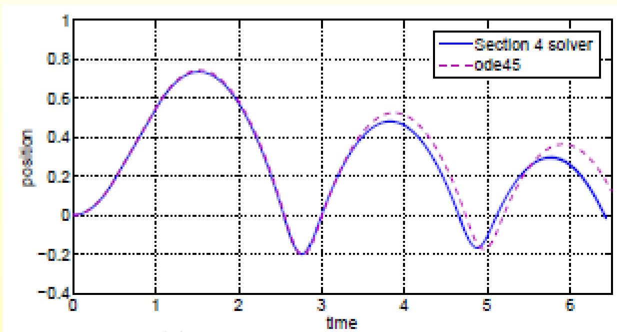


Time vs. evaluations (detail)

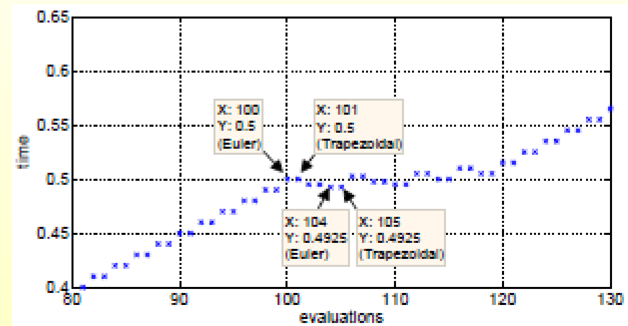
Simultaneous inequalities

Eval	Time	Position	Velocity	F_{floor}	Error
532	2.5450	0.0037	-1.4381	0	
533	2.5450	-0.0035	-1.4381	8.5888	0
534	2.5550	-0.0107	-1.4398	11.4717	
535	2.5550	-0.0179	-1.4377	14.3430	0.0021
536	2.5500	-0.0035	-1.4348	8.5700	
537	2.5500	-0.0107	-1.4369	11.4555	0.0021
538	2.5475	0.0001	-1.4375	0	
539	2.5475	-0.0071	-1.4395	10.0333	0.0020
540	2.5462	0.0019	-1.4380	0	
541	2.5462	-0.0053	-1.4389	9.3125	0.0009
542	2.5456	0.0028	-1.4381	0	
543	2.5456	-0.0044	-1.4385	8.9508	0.0004

Computational simulation



Position vs. time



Time vs. evaluations (detail)

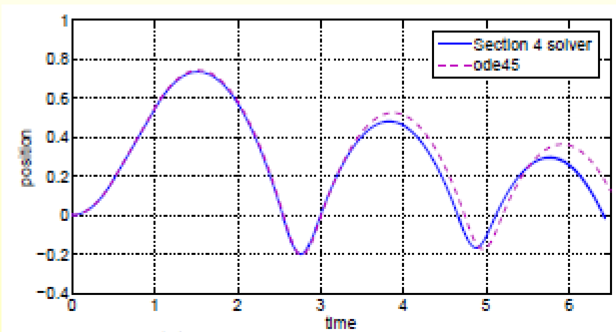
Simultaneous inequalities

Eval	Time	Position	Velocity	F_{floor}	Error
532	2.5450	0.0037	-1.4381	0	
533	2.5450	-0.0035	-1.4381	8.5888	0
534	2.5550	-0.0107	-1.4398	11.4717	
535	2.5550	-0.0179	-1.4377	14.3430	0.0021
536	2.5500	-0.0035	-1.4348	8.5700	
537	2.5500	-0.0107	-1.4369	11.4555	0.0021
538	2.5475	0.0001	-1.4375	0	
539	2.5475	-0.0071	-1.4395	10.0333	0.0020
540	2.5462	0.0019	-1.4380	0	
541	2.5462	-0.0053	-1.4389	9.3125	0.0009
542	2.5456	0.0028	-1.4381	0	
543	2.5456	-0.0044	-1.4385	8.9508	0.0004

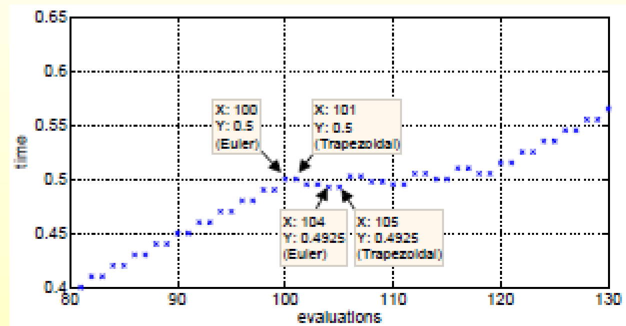
Finite state machine

Eval	Time	Position	Velocity	F_{floor}	Error	ξ_{con}
532	2.5450	0.0037	-1.4381	0		0
533	2.5450	-0.0035	-1.4381	0	0	0
534	2.5550	-0.0107	-1.4521	11.5331		1
535	2.5550	-0.0179	-1.4438	14.3980	0.0082	1
536	2.5500	-0.0035	-1.4348	8.5820		1
537	2.5500	-0.0107	-1.4369	11.4614	0.0021	1
538	2.5475	0.0001	-1.4375	7.1446		1
539	2.5475	-0.0071	-1.4382	0	0.0008	0
540	2.5462	0.0019	-1.4398	6.4386		1
541	2.5462	-0.0053	-1.4392	0	0.0006	0

Computational simulation



Position vs. time



Time vs. evaluations (detail)

Simultaneous inequalities

Eval	Time	Position	Velocity	F_{floor}	Error
532	2.5450	0.0037	-1.4381	0	
533	2.5450	-0.0035	-1.4381	8.5888	0
534	2.5550	-0.0107	-1.4398	11.4717	
535	2.5550	-0.0179	-1.4377	14.3430	0.0021
536	2.5500	-0.0035	-1.4348	8.5700	
537	2.5500	-0.0107	-1.4369	11.4555	0.0021
538	2.5475	0.0001	-1.4375	0	
539	2.5475	-0.0071	-1.4395	10.0333	0.0020
540	2.5462	0.0019	-1.4380	0	
541	2.5462	-0.0053	-1.4389	9.3125	0.0009
542	2.5456	0.0028	-1.4381	0	
543	2.5456	-0.0044	-1.4385	8.9508	0.0004

Finite state machine

Eval	Time	Position	Velocity	F_{floor}	Error	ξ_{con}
532	2.5450	0.0037	-1.4381	0		0
533	2.5450	-0.0035	-1.4381	0	0	0
534	2.5550	-0.0107	-1.4521	11.5331		1
535	2.5550	-0.0179	-1.4438	14.3980	0.0082	1
536	2.5500	-0.0035	-1.4348	8.5820		1
537	2.5500	-0.0107	-1.4369	11.4614	0.0021	1
538	2.5475	0.0001	-1.4375	7.1446		1
539	2.5475	-0.0071	-1.4382	0	0.0008	0
540	2.5462	0.0019	-1.4398	6.4386		1
541	2.5462	-0.0053	-1.4392	0	0.0006	0

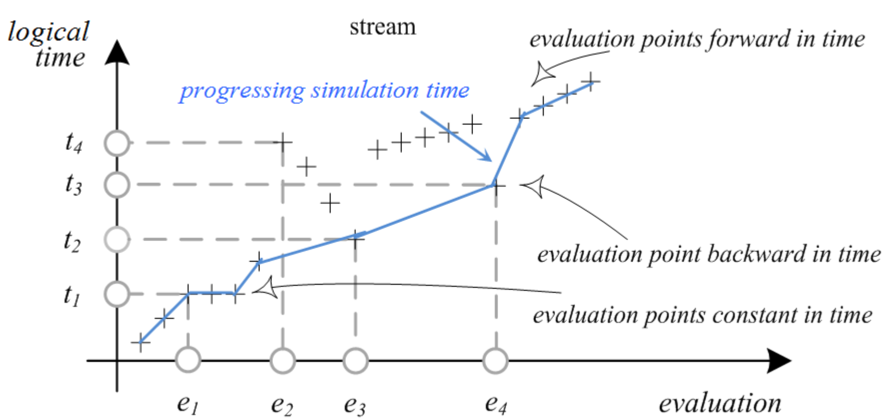
But time is a function of evaluations

- Simultaneous inequalities

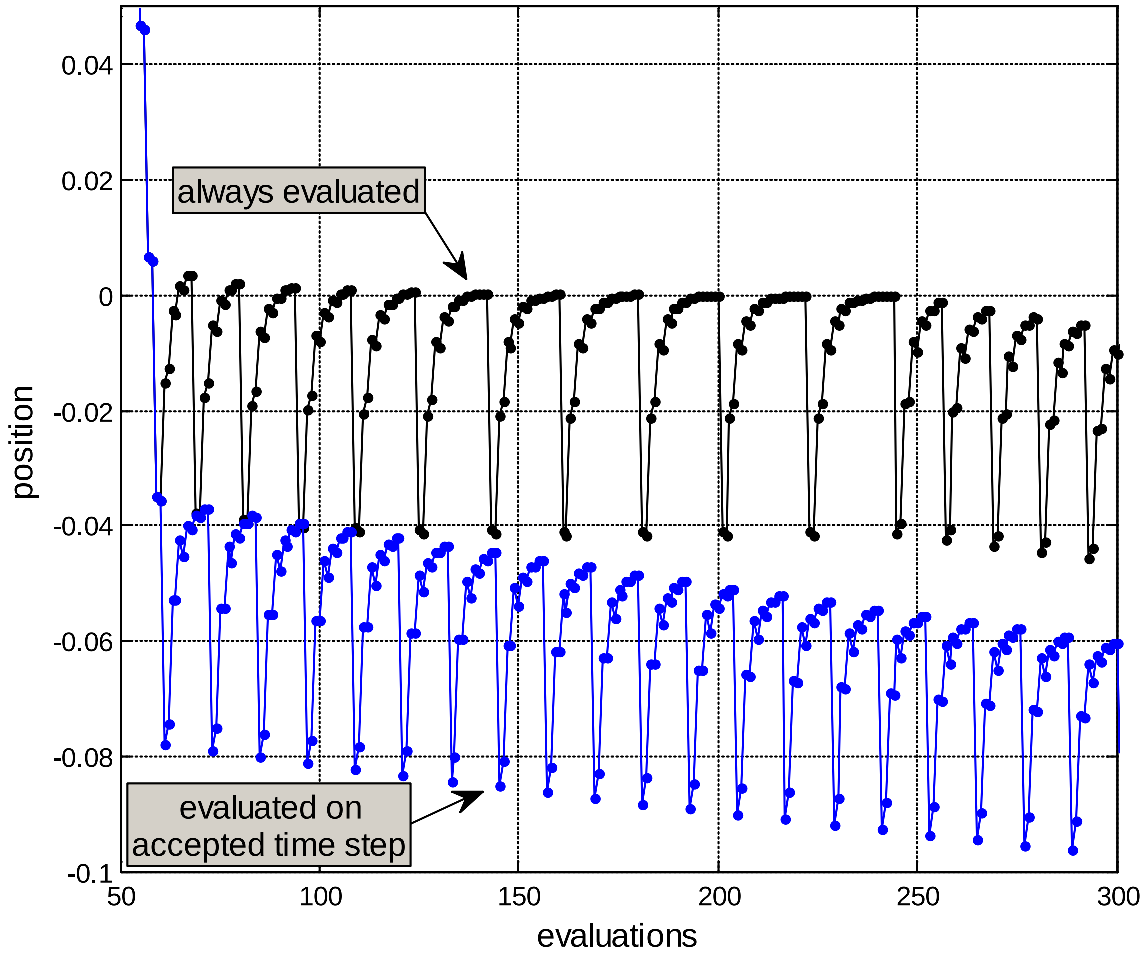
$$F_{floor}(t_{event}(e)) = \begin{cases} -\left(Rv(t_{event}(e)) + \dfrac{x(t_{event}(e))}{C}\right) & if\ x(t_{event}(e)) < 0 \\ 0 & otherwise \end{cases}$$

$$a_{ball}(t_{smooth}(e)) = g + \frac{F_{floor}(t_{smooth}(e))}{m_{ball}}$$

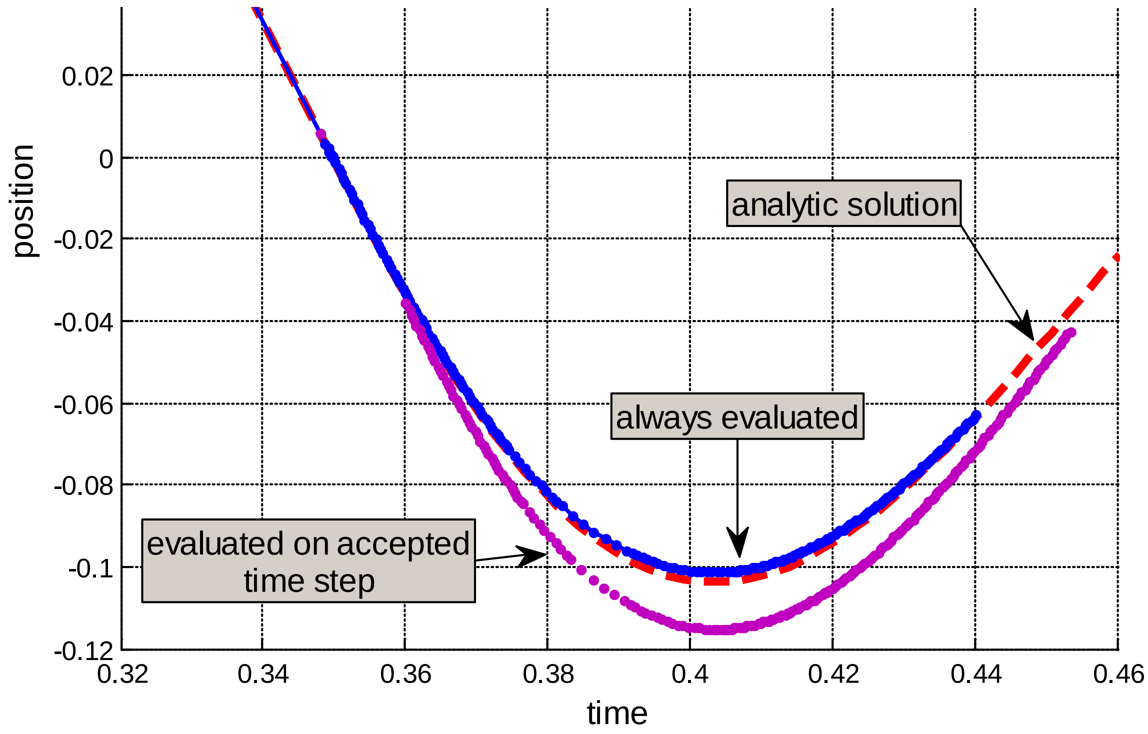
- Which t should t_{event} really map onto?



Different choice of semantics



Comparing with an analytic solution



Justyna Zander, Pieter J. Mosterman, Grégoire Hamon, and Ben Denckla, "**On the Structure of Time in Computational Semantics of a Variable-Step Solver for Hybrid Behavior Analysis**," in the *Proceedings of the IFAC World Congress*, Milan, Italy, August, 2011

Characteristics of the semantic domain

- Declarative
  - Purely functional (no side effects)
- Ordered evaluations
- Untimed
  - Time as explicit function, *t(e)*
  - Time is not strictly increasing
- Broadly applicable to dynamic systems
  - Differential equations, difference equations, discrete events

Pieter J. Mosterman, Justyna Zander, Grégoire Hamon, and Ben Denckla, "**Towards Computational Hybrid System Semantics for Time-Based Block Diagrams**," in *3rd IFAC Conference on Analysis and Design of Hybrid Systems* (ADHS'09), A. Giua, C. Mahulea, M. Silva, and J. Zaytoon (eds.), pp. 376-385, Zaragoza, Spain, September 16-18, 2009, plenary paper.

Agenda

- Outline
- Model-Based Design
- Problem statement
- A solution approach
- Outlook

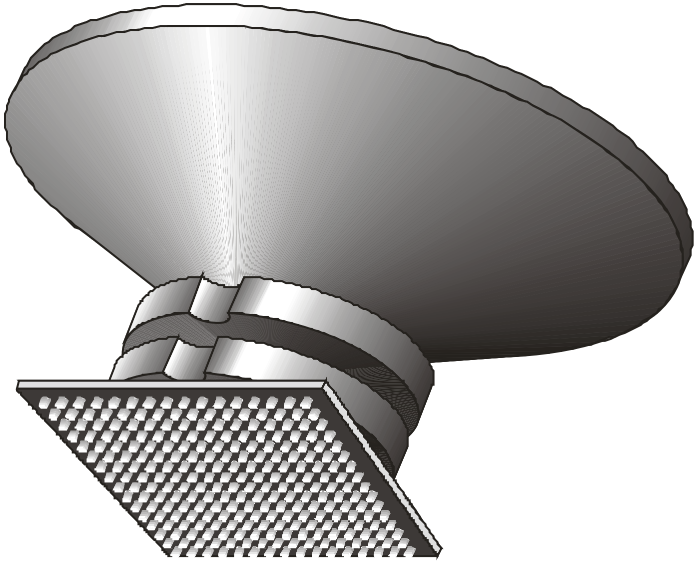
Conclusions

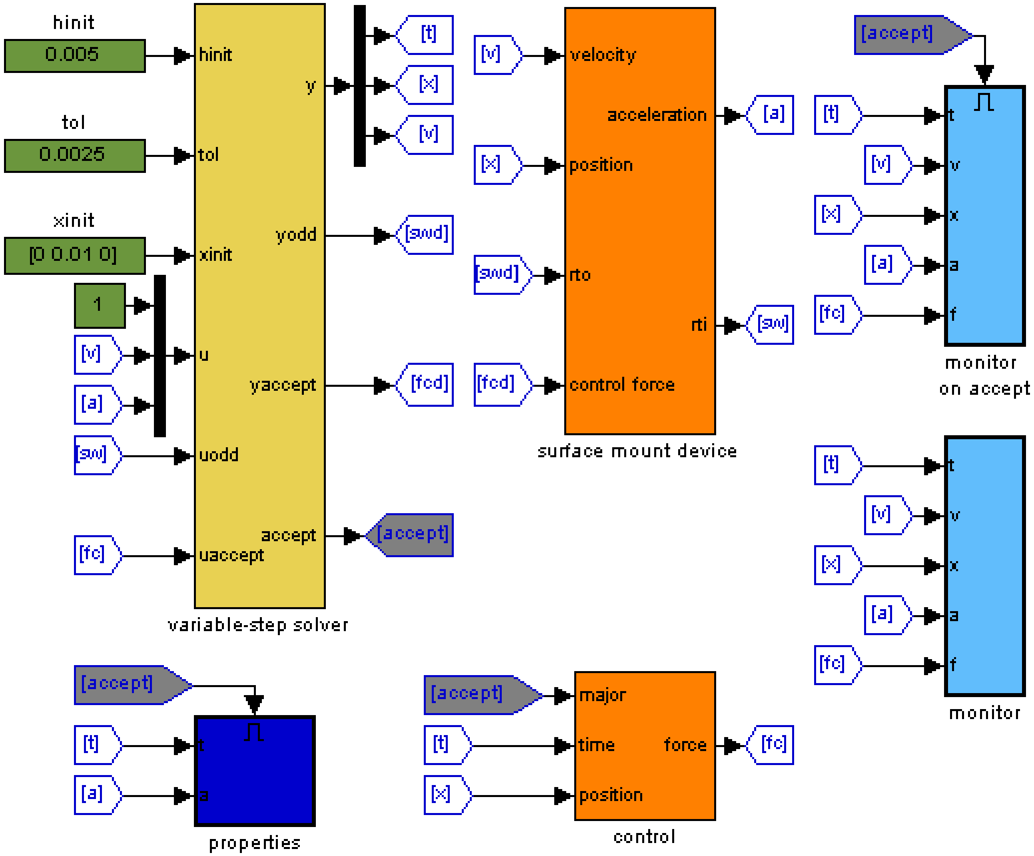
- Computation, the good and not so good
  - Quantitative approximation
  - Potential for higher quality models
- Exploit computational methods
  - We must formalize the computational execution semantics
  - Model at a declarative level
- Define solvers using a functional stream-based approach
  - Precise computational semantics of the execution engine

Opportunities

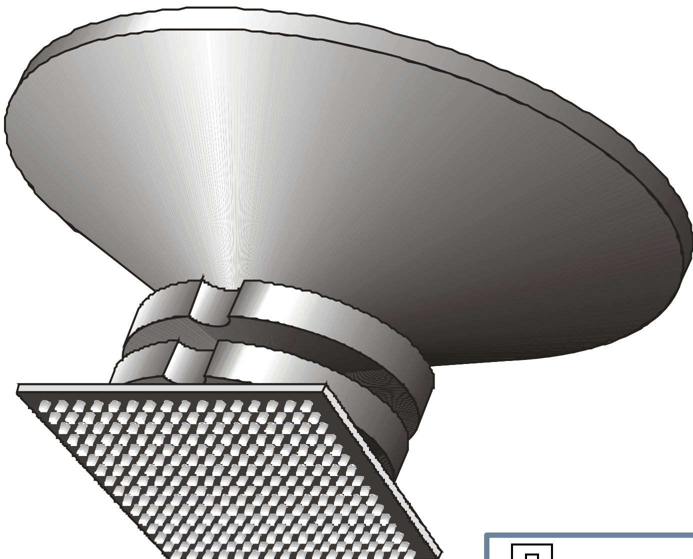
- First principles in computational form
- New generation of modeling and simulation tools
  - More robust in less time (less bugs, more reliable and consistent approximation)
- Bring disciplines together
  - Engineering, Computer Science, Physics, Mathematics
- Exploit the abstraction
  - Automatic code generation
  - Computational methods for
    - Analysis
    - Design
    - Synthesis

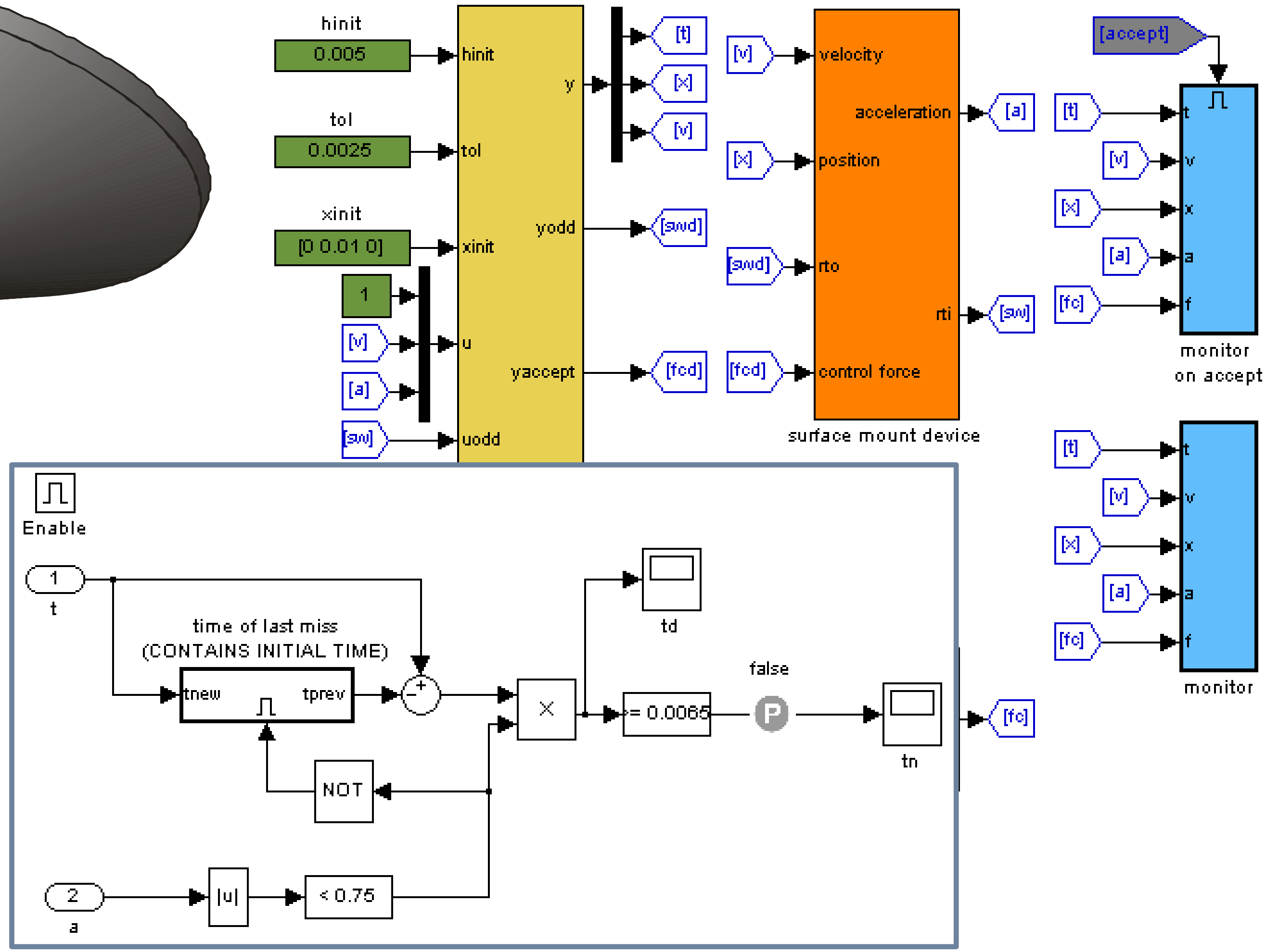
Control synthesis using model checking



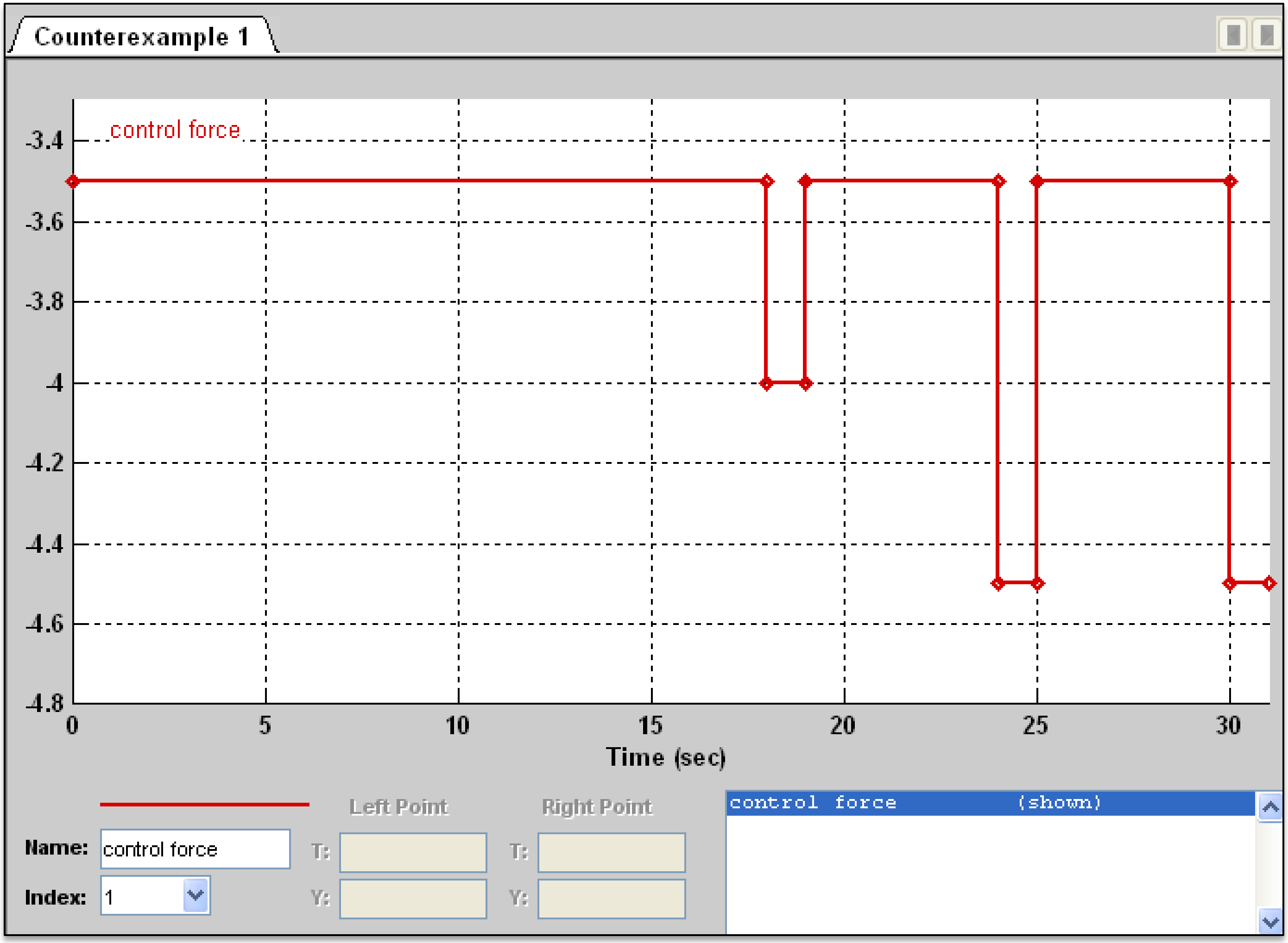


Control synthesis using model checking



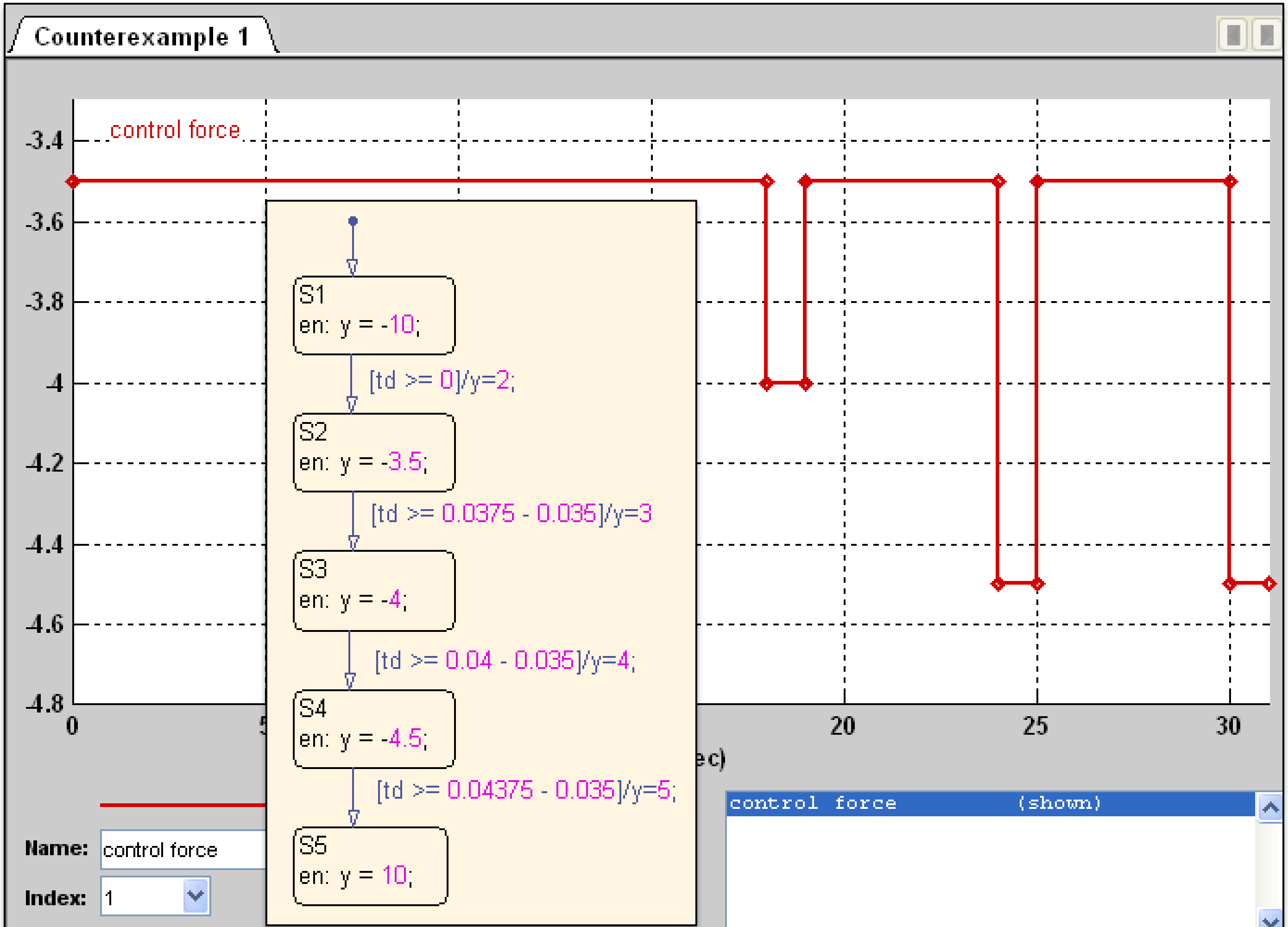


A counterexample



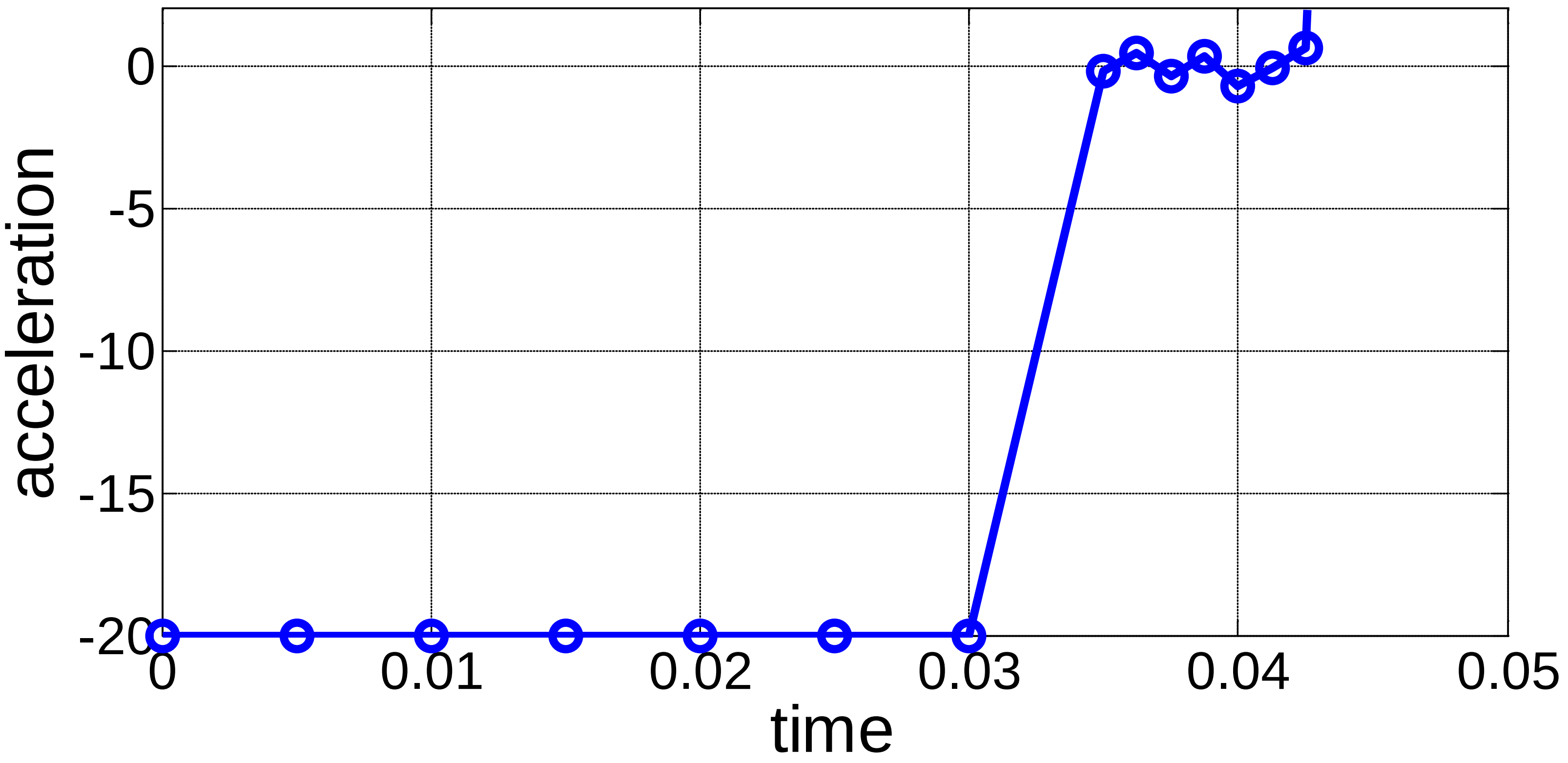
Pieter J. Mosterman, Justyna Zander, Grégoire Hamon, and Ben Denckla, "**A Computational Model of Time for Stiff Hybrid Systems Applied to Control Synthesis**," in *Control Engineering Practice*, in press

A counterexample



Pieter J. Mosterman, Justyna Zander, Grégoire Hamon, and Ben Denckla, "**A Computational Model of Time for Stiff Hybrid Systems Applied to Control Synthesis**," in *Control Engineering Practice*, in press

Controlled acceleration of mounted component



Acknowledgments

Justyna Zander
Harvard University
Fraunhofer Institute FOKUS, Berlin

Grégoire Hamon
MathWorks

Ben Denckla
Independent Thinker

Hans Vangheluwe
University of Antwerp
McGill University

Many thanks for their continuing collaboration!

MathWorks®

Accelerating the pace of engineering and science