MathWorks

Modeling Approximation of Computational Semantics for Cyber-Physical System Design

Pieter J. Mosterman

Senior Research Scientist
Design Automation Department
MathWorks

Adjunct Professor
School of Computer Science
McGill

MathWorks

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

2

MathWorks

Modeling Approximation of Computational Semantics for Cyber-Physical System Design

Pieter J. Mosterman

Senior Research Scientist
Design Automation Department
MathWorks

Adjunct Professor
School of Computer Science
McGill

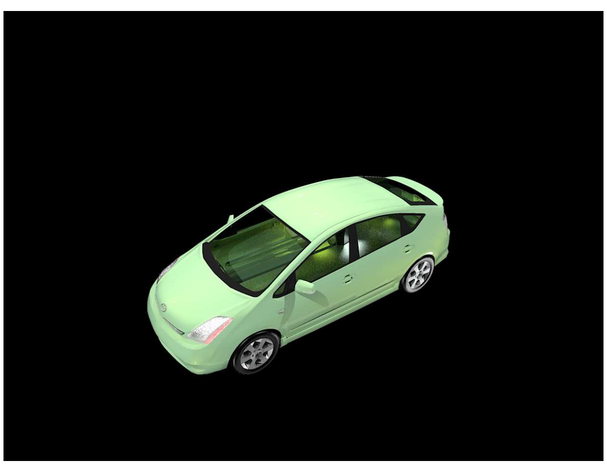
© 2012 The MathWorks, Inc.

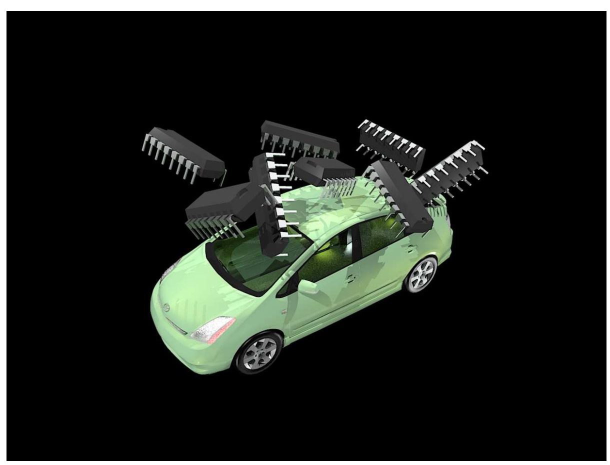
MathWorks

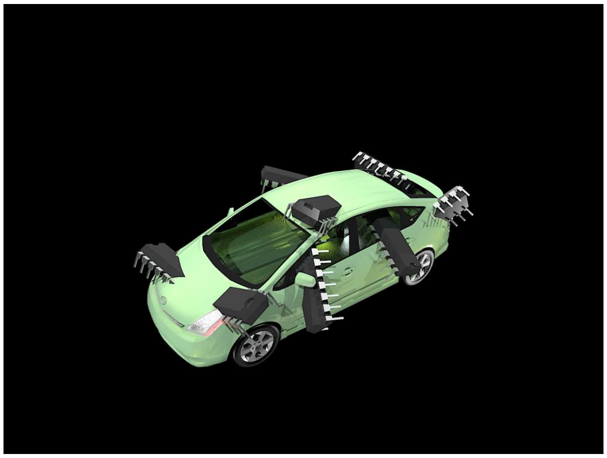
Agenda

- Cyber-physical systems
- Modeling cyber-physical systems
- Modeling approximations
- A solver model for control synthesis
- Conclusions

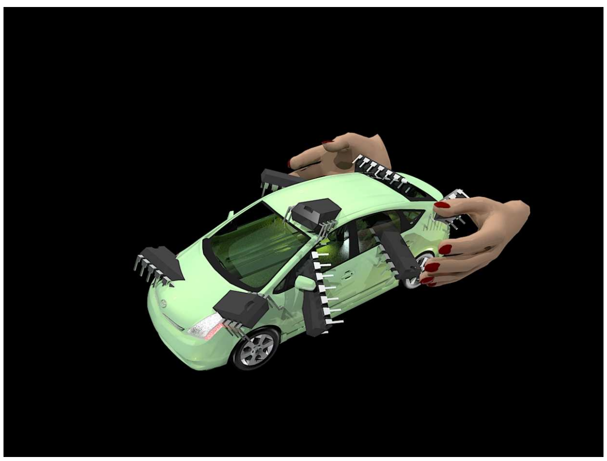
4

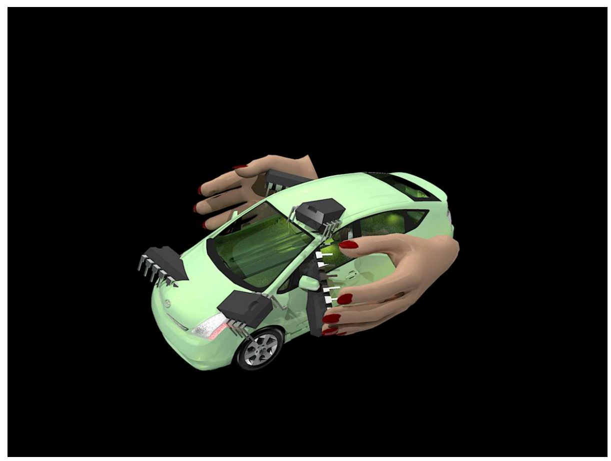


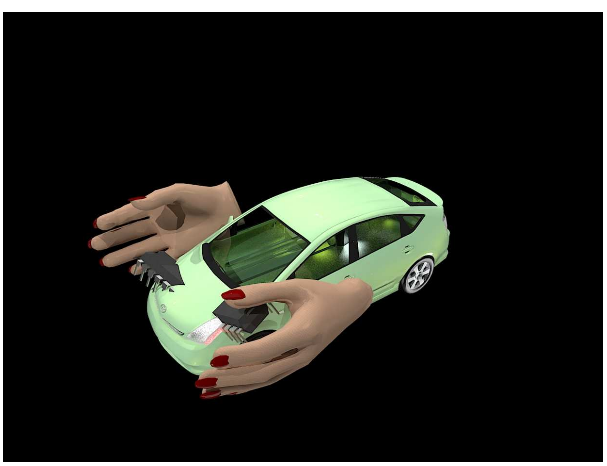


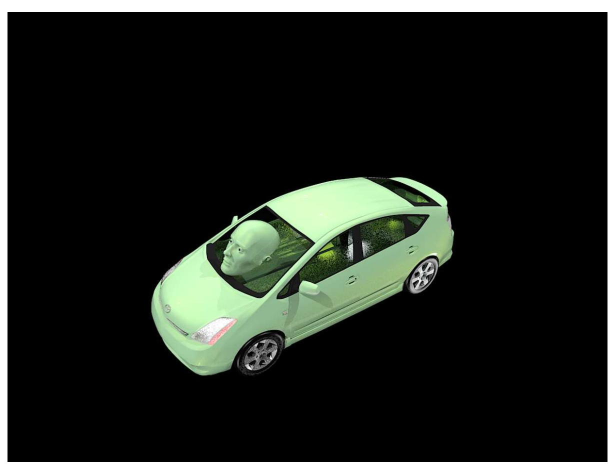



System Integration
Timing
Concurrency
Interfaces
Shared resources
…





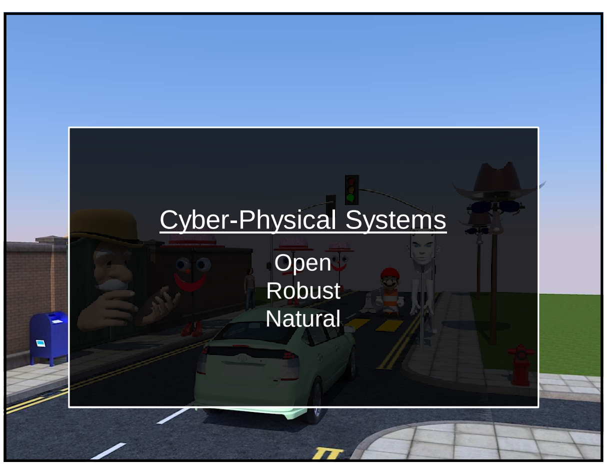


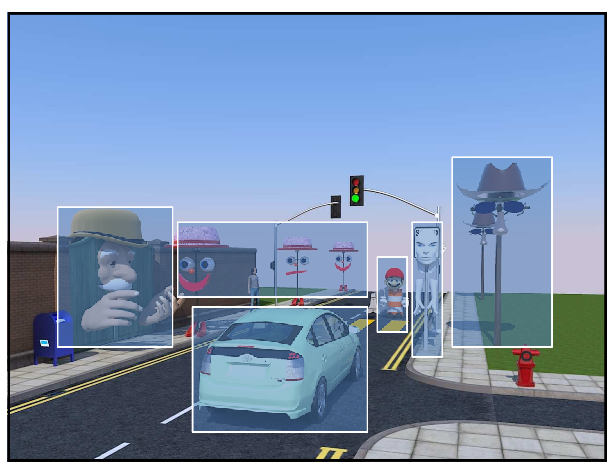


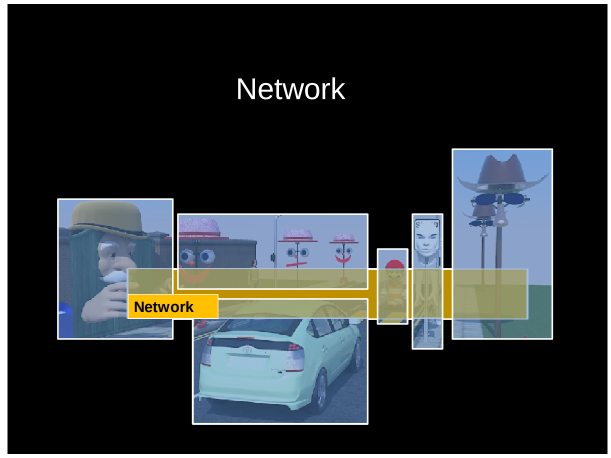

SPEED
LIMIT
35

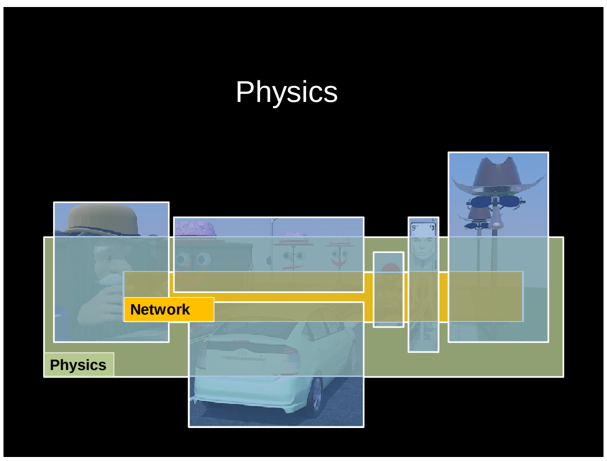

SPEED
LIMIT
35

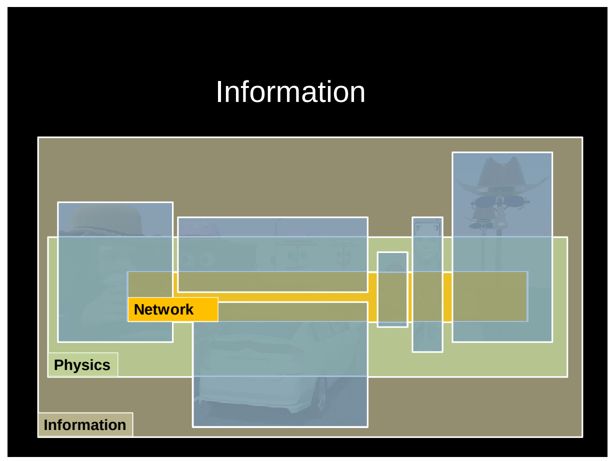

SPEED
LIMIT
35

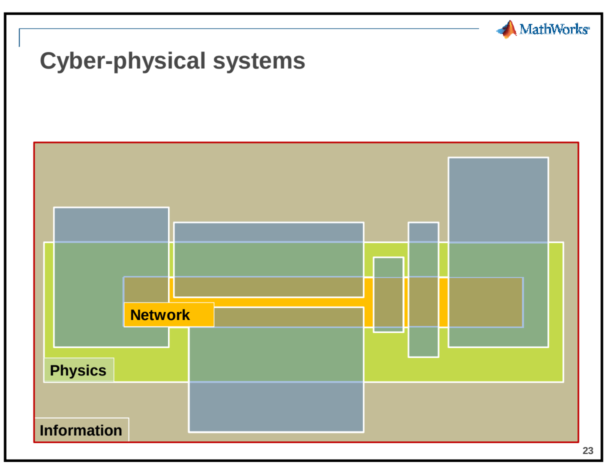

Cyber-Physical Systems
Open
Robust
Natural

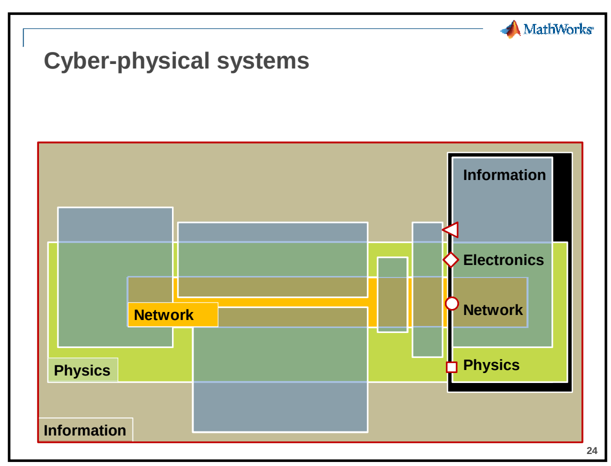



Network
Network


Physics
Network
Physics


Information
Network
Physics
Information


MathWorks
Cyber-physical systems
Network
Physics
Information
23


MathWorks
Cyber-physical systems
Information
Electronics
Network
Physics
Network
Physics
Information
24

MathWorks

Agenda

- Cyber-physical systems
- Modeling cyber-physical systems
- Modeling approximations
- A solver model for control synthesis
- Conclusions

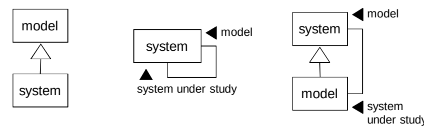
MathWorks

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"



MathWorks

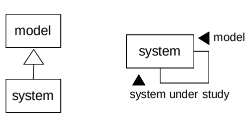
What is a model anyway?

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

MathWorks

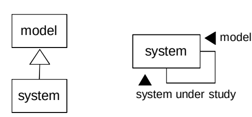
What is a model anyway?

Jean Bézivin — "Everything is a model"

Jean-Marie Favre — "Nothing is a model"

Pieter J. Mosterman — "Nothing is not a model"

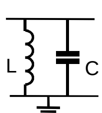
Hans Vangheluwe — "Model everything"



Computer Automated Multiparadigm Modeling (CAMPaM)

MathWorks

Modeling a physical system



Capacitor: $V = \frac{q}{C}$

Inductor: $I = \frac{p}{L}$

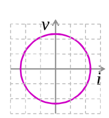
Maxwell: $i(t) = \frac{dq(t)}{dt}$

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

An ideal oscillator:

$$i(t) = C\frac{dv(t)}{dt}$$

$$v(t) = L\frac{di(t)}{dt}$$

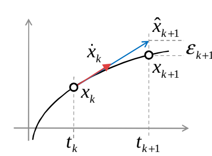


MathWorks

Numerical integration

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

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



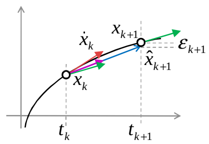
Numerical integration

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

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



Taylor series expansion for error analysis

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

$\varepsilon_e(t_{k+1})$ $\varepsilon_t(t_{k+1})$

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

31

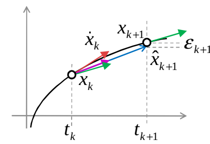
Numerical integration

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

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

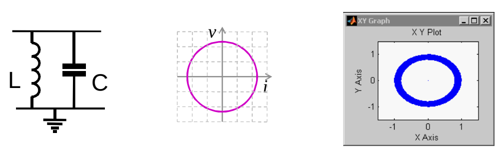
$\varepsilon_e(t_{k+1})$ $\varepsilon_t(t_{k+1})$

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

32

Sophisticated solver … ?

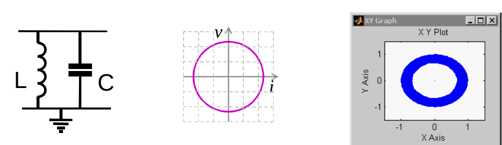
- Let's compute a solution to an ideal oscillator



33

Sophisticated solver … ?

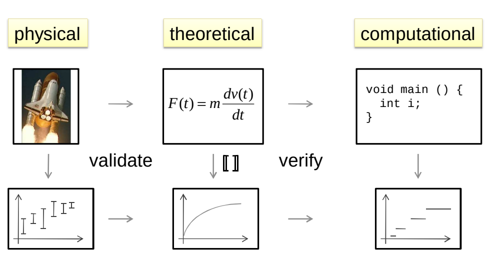
- Let's compute a solution to an ideal oscillator



- We can make the error small … but only locally!
- It accumulates for long time behavior
- *So, … how come we can engineer today's complex systems?!*

34

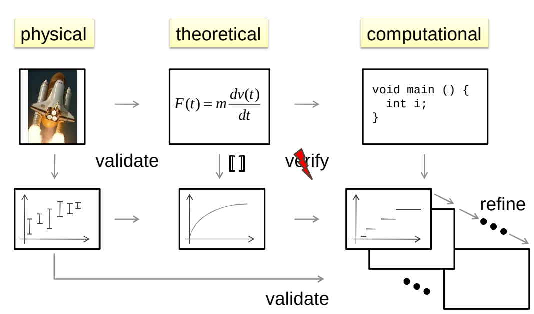
The models in engineering an embedded system



In collaboration with Hans Vangheluwe, McGill University

35

The models in engineering an embedded system



In collaboration with Hans Vangheluwe, McGill University

36

Agenda

- Cyber-physical systems
- Modeling cyber-physical systems
- Modeling approximations
- A solver model for control synthesis
- Conclusions

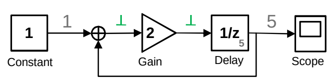
Desiderata of an execution engine model

- Declarative
  - No implementation details
- Stateless
  - State explicitly formulated (e.g., as input)
- Function composition

A declarative formalism with fix-point semantics

A LATTICE-THEORETICAL FIXPOINT THEOREM AND ITS APPLICATIONS
ALFRED TARSKI
Pacific J. Math. 5 (1955), 285-309

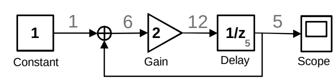
- Repeated application of a monotonically increasing partial function converges to a fixed point



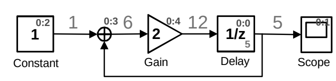
A declarative formalism with fix-point semantics

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

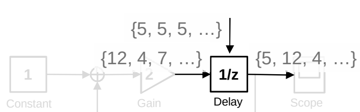


- One implementation is a data dependency schedule



Time as a potentially infinite stream of values

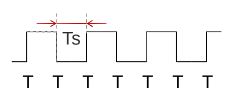
- Stream(Type) = Type : Stream(Type)
- Blocks as a function application
  - Delay x0 u = x0 : u



- Resulting formalism
  - Causal block diagrams
  - Stream-based semantics

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

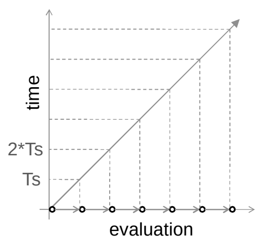
- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations



MathWorks

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

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

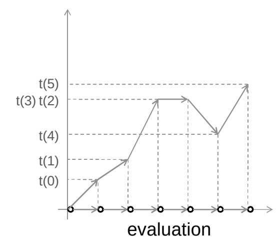


43

MathWorks

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



Pieter J. Mosterman, Justyna Zander, Grégoire Hamon, and Ben Denckla, "**Towards Computational Hybrid System Semantics for Time-Based Block Diagrams,**" in *Proceedings of the 3rd IFAC Conference on Analysis and Design of Hybrid Systems*, pp. 376-385, Zaragoza, Spain, 2009

44

MathWorks

The two stages of a stream based functional solver

Euler integration

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

Trapezoidal integration

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

45

MathWorks

The two stages of a stream based functional solver

Euler integration

increment

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

Trapezoidal integration

increment

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

Error computation

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

46

MathWorks

The two stages of a stream based functional solver

Euler integration

increment — previous increment

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

Trapezoidal integration

increment — previous increment

$$y_t(e) = \sum_{i=1}^{e} \frac{(u(i-1)+u(i))h(i-1)}{2} - \frac{(u(i-3)+u(i-2))h(i-3)}{2} p(i-1)$$

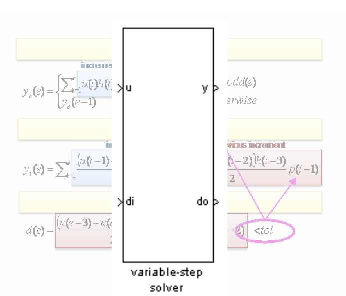
Error computation

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

47

MathWorks

The two stages of a stream based functional solver



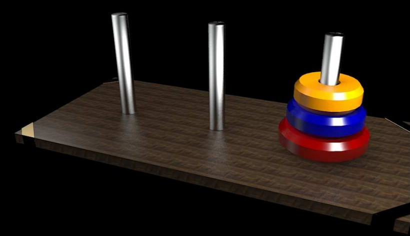
48

Agenda

- Cyber-physical systems
- Modeling cyber-physical systems
- Modeling approximations
- ➤ A solver model for control synthesis
- Conclusions

49

Towers of Hanoi



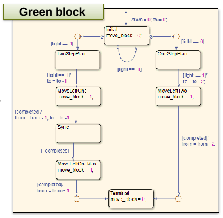
A Cyber-Physical System?

A Cyber-Physical System!

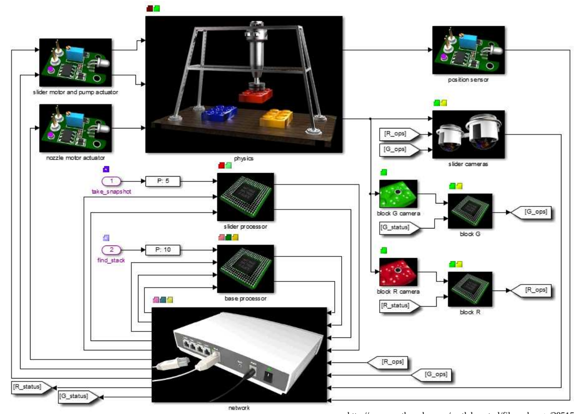
Justyna Zander and Pieter J. Mosterman, "**Technical Engine for Democratizing Modeling, Simulation, and Prediction**," in *Winter Simulation Conference*, December, 2012

Local control laws

- Green block
  - Should be on top
  - If it is on top, move one spot over and then move one spot over
  - If it is at bottom, move two spots over
- Red block
  - Should be on bottom
  - If it is on top, move two spots over
  - If it is on bottom, move two spots over
  - Should have the highest priority



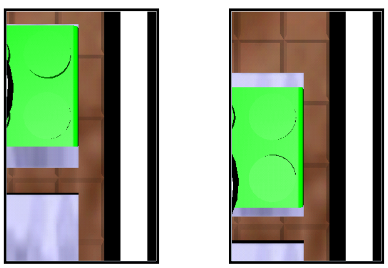
55



http://www.mathworks.com/matlabcentral/fileexchange/38515

Stereoscopic analysis to find the stack of blocks

- Multiple values at one time step



left video frame

right video frame

57

Stereoscopic analysis to find the stack of blocks

- Multiple values at one time step

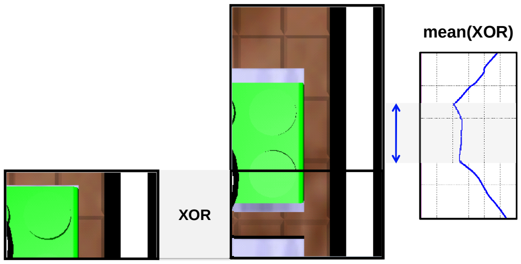
left video frame

right video frame

58

Stereoscopic analysis to find the stack of blocks

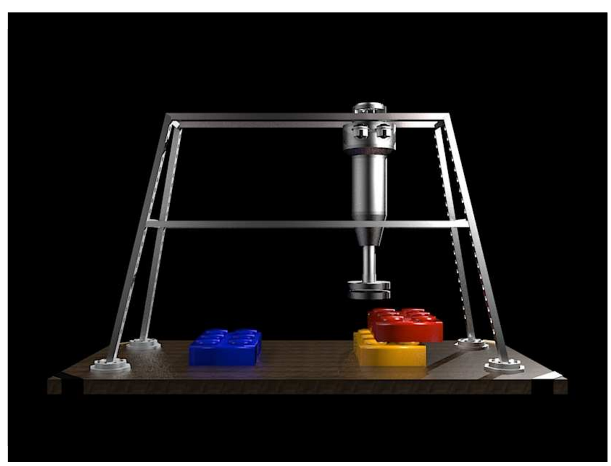
- Multiple values at one time step

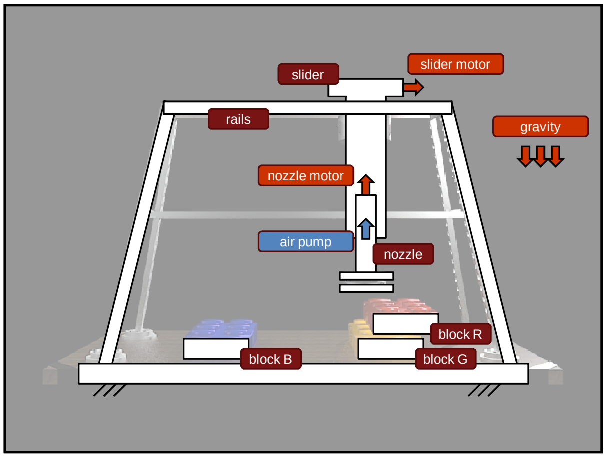


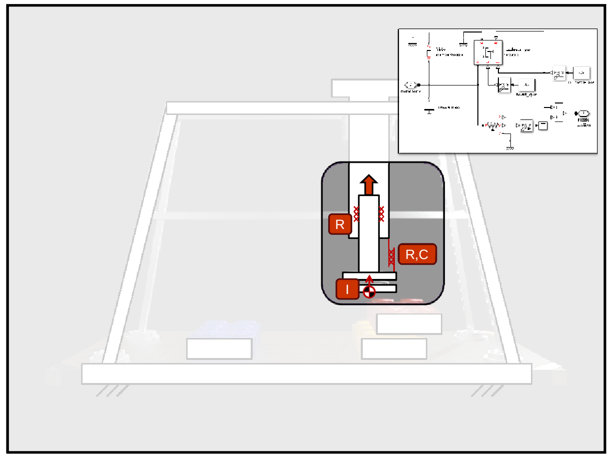
left video frame

right video frame

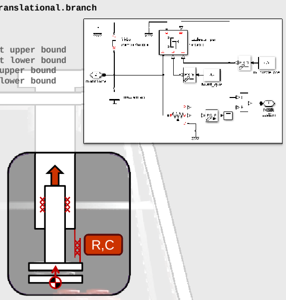
59



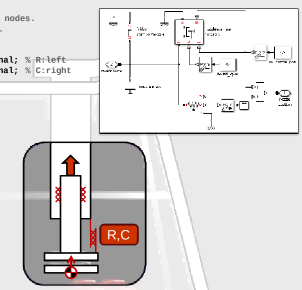


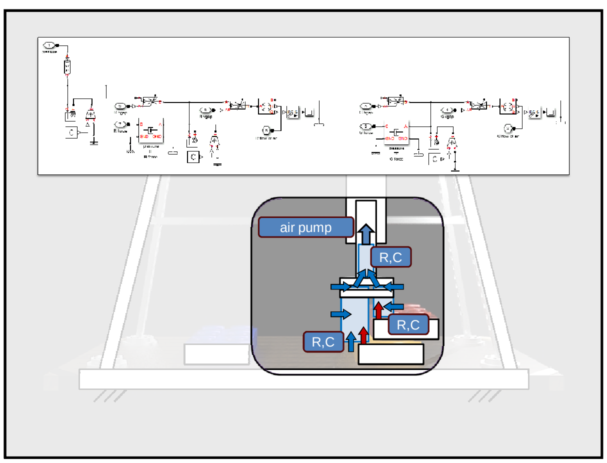


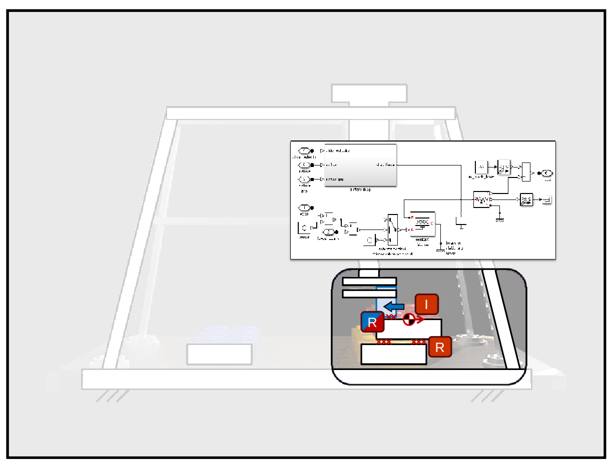

```
component hardstop_external_ini < foundation.mechanical.translational.branch
% Translational Hard Stop With External Initial Position

  parameters
    stiff_up = { 1e6, 'N/m' };       % Contact stiffness at upper bound
    stiff_low = { 1e6, 'N/m' };      % Contact stiffness at lower bound
    D_up = { 150, 'N*s/m'};          % Contact damping at upper bound
    D_low = { 150, 'N*s/m'};         % Contact damping at lower bound
  end

  inputs
    upper_bnd = { 0.1, 'm' };        % U:right
    lower_bnd = { -0.1, 'm' };       % L:left
    x_initial = { 0.0, 'm' };        % I:left
  end

  variables
    x = { 0, 'm'};
  end

  function setup
    if stiff_up <= 0
      pm_error('simscape:GreaterThanZero','Stiffness')
    end
    x = 0.0;
  end

  equations
    if ((x + x_initial) > upper_bnd)
        % Slider hits upper bound
        f == stiff_up * ((x + x_initial) - upper_bnd) + D_up * v;
    elseif ((x + x_initial) < lower_bnd)
        % Slider hits lower bound
        f == stiff_low * ((x + x_initial) - lower_bnd) + D_low * v;
    else
        % Slider is between hardstops
        f == {0 'N'};
    end
    x.der == v;
  end
end
```




```
component(Hidden=true) branch
% Translational Branch
% Defines a translational branch with R and C external nodes.
% Also defines associated through and across variables.

  nodes
    R = foundation.mechanical.translational.translational; % R:left
    C = foundation.mechanical.translational.translational; % C:right
  end

  variables
    f = { 0, 'N' };
    v = { 0, 'm/s' };
  end

  function setup
    through( f, R.f, C.f );
    across( v, R.v, C.v );
  end

end
```

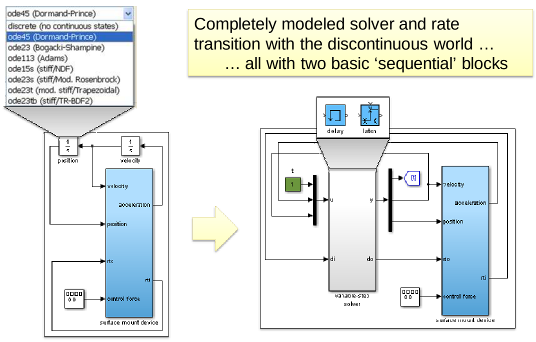




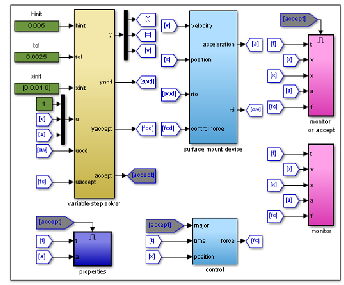


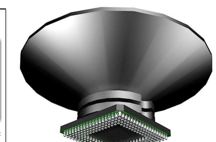
Explicitly modeling the execution engine

Completely modeled solver and rate transition with the discontinuous world …
… all with two basic 'sequential' blocks



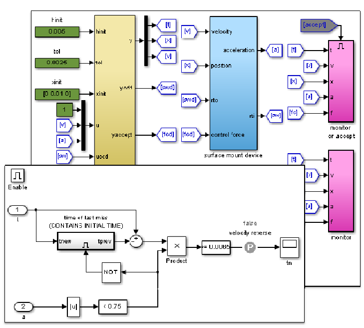
Control synthesis for a surface mount device using model checking

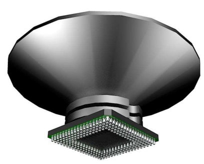




http://www.mathworks.com/matlabcentral/fileexchange/authors/4449

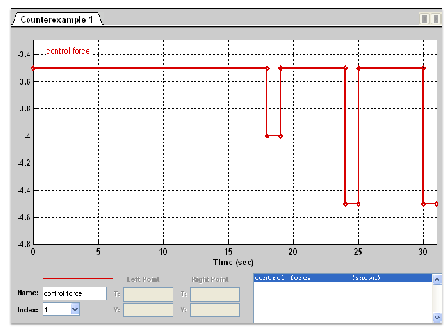
Control synthesis for a surface mount device using model checking





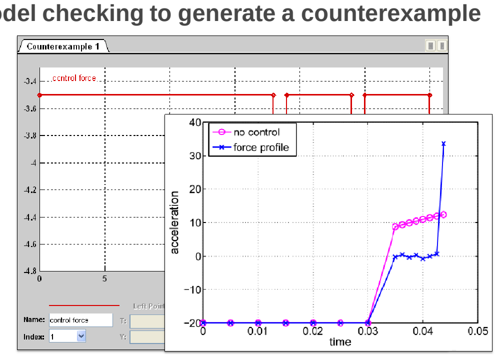
http://www.mathworks.com/matlabcentral/fileexchange/authors/4449

Model checking to generate 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*, , vol. 20, nr. 1, pp. 2-13, January 2012

Model checking to generate a counterexample



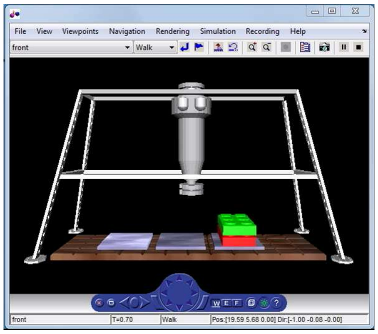
Pieter J. Mosterman, Justyna Zander, Grégoire Hamon, and Ben Denckla, "**A Computational Model of Time for Stiff Hybrid Systems Applied to Control Synthesis**," in *Control Engineering Practice*, , vol. 20, nr. 1, pp. 2-13, January 2012

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.

Scenarios—emerging behavior



73

Agenda

- Cyber-physical systems
- Modeling cyber-physical systems
- Modeling approximations
- A solver model for control synthesis
- Conclusions

74

Conclusions

- Today's systems are open
  - Interact across various modalities
- Computational models include a variety of semantics
  - Many interacting approximations
- We should understand our computational methods
- Model solvers
  - A functional stream-based approach
  - Formalize computational semantics of the execution engine
- Exploit the abstraction
  - Computational methods for analysis, design, and synthesis
- Bring disciplines together
  - Engineering, Computer Science, Physics, Mathematics

75

Acknowledgments

Justyna Zander
Harvard University
SimulatedWay, Berlin

Hans Vangheluwe
University of Antwerp
McGill University

Many thanks for their continuing collaboration!

76

MathWorks®
Accelerating the pace of engineering and science

77