

# Rigorous Simulation

Walid Taha - Halmstad University

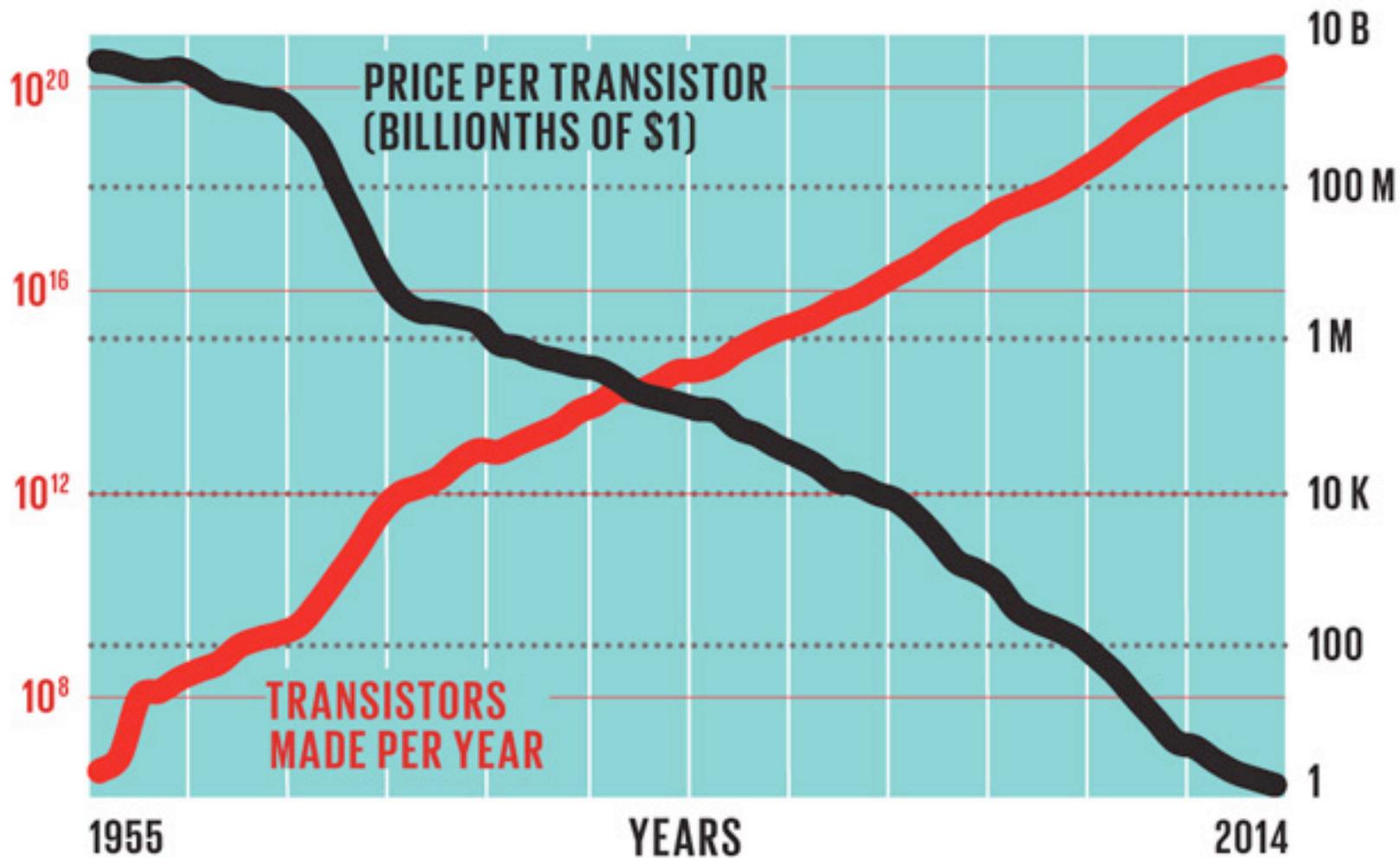


# Our World is Changing



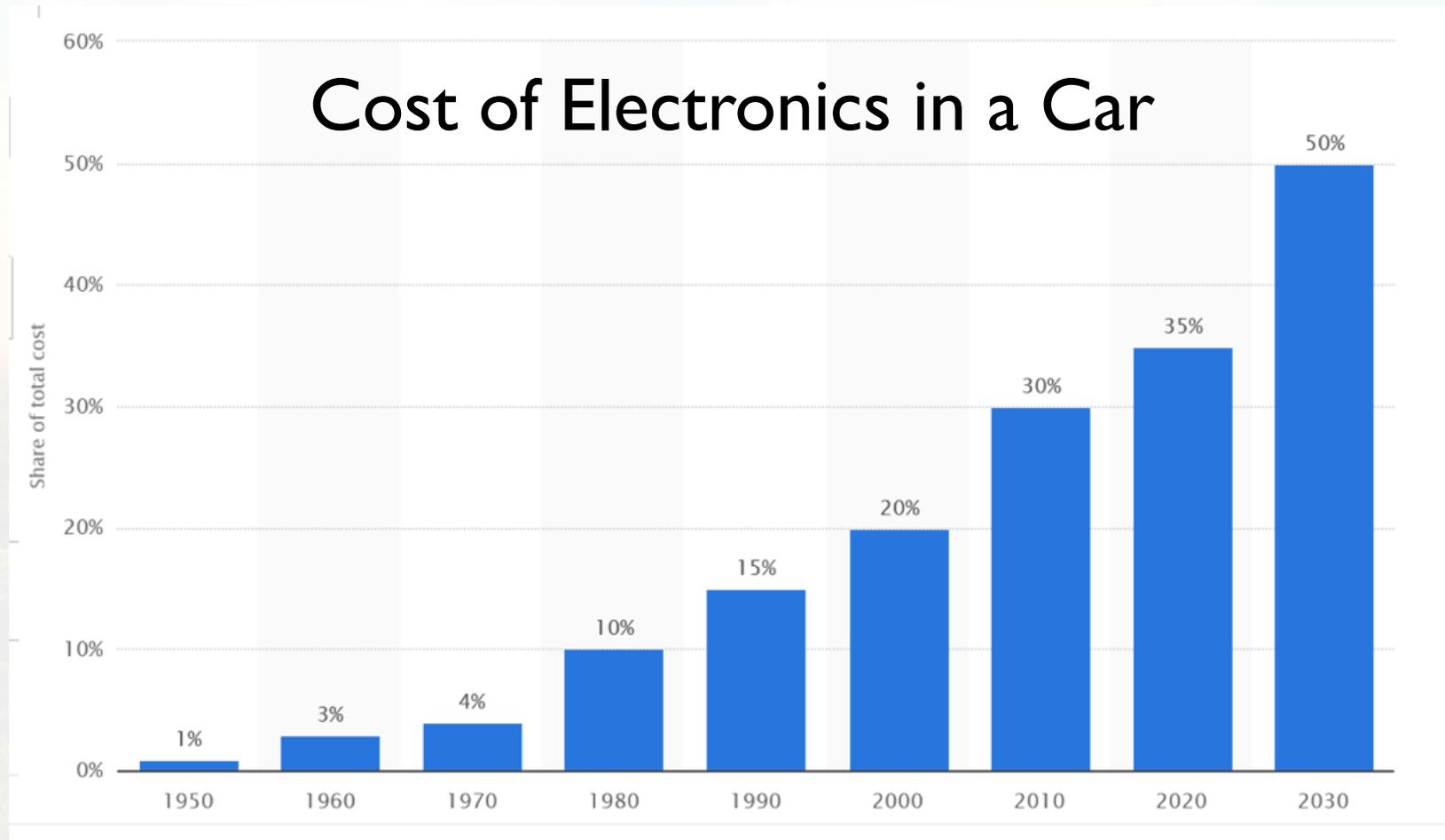
Illustration Copyright with National Science Foundation. Used by permission.

# Our World is Changing



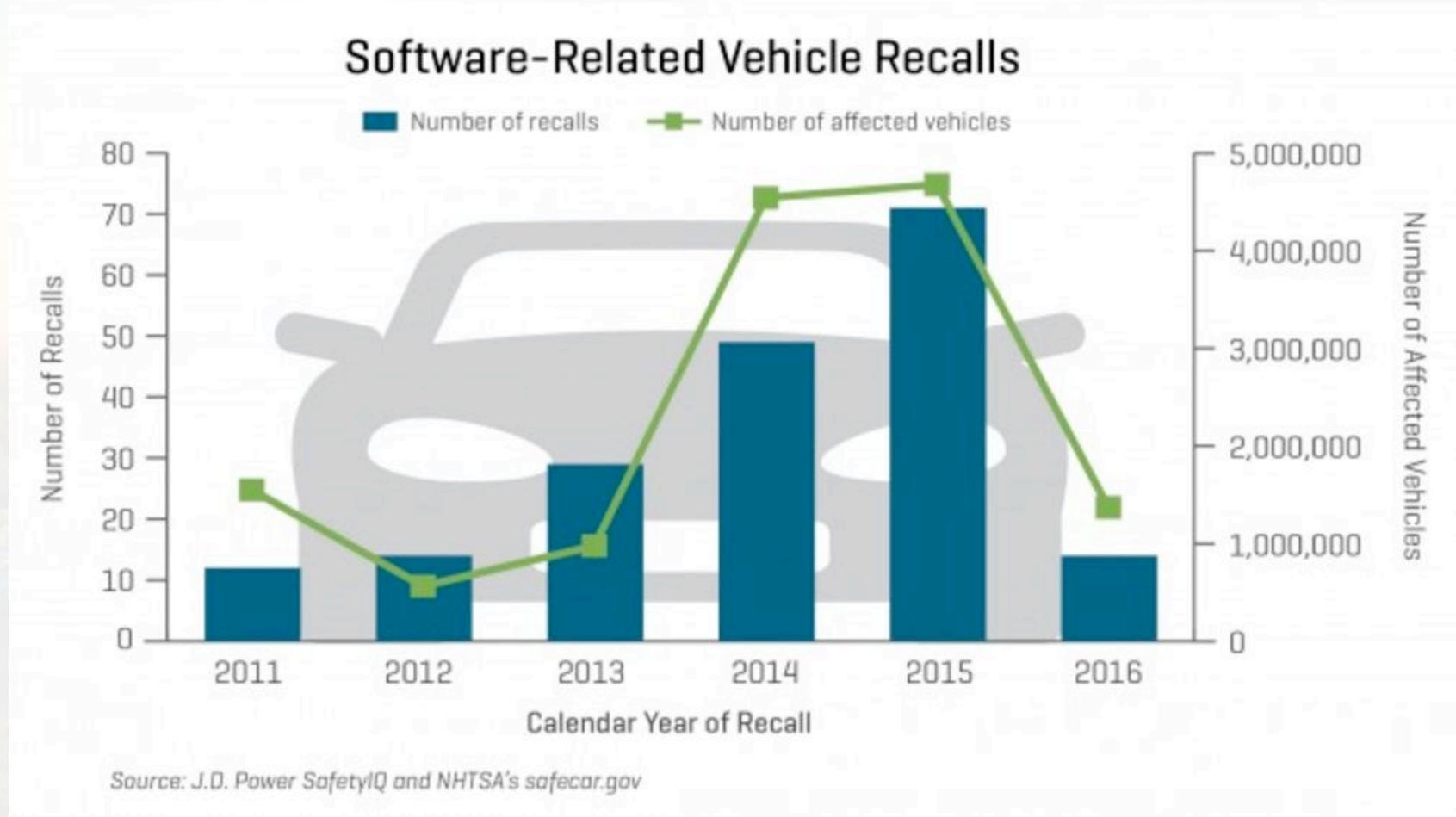
<http://spectrum.ieee.org/computing/hardware/transistor-production-has-reached-astronomical-scales>

# Our World is Changing



<http://electronicscomponentsworld.com/test-implications-arising-from-the-advent-of-connected-cars/>

# Our World is Changing



# Problem

- Broadly: design technology is fragmented:
  - Software engineering environments
  - Simulation/Co-simulation
  - Symbolic algebra methods
  - Formal methods
- Specifically: formally correct simulation

# Our Approach

"... if thought corrupts language, language can also corrupt thought."

George Orwell, *Politics and the English Language*, 1946 .

# Domain-Specific Languages

- Our approach to DSL development:
  - Formalize the **notation & semantics**
  - **Implement, test, evaluate**, and
  - ... repeat
- 1st iteration 2006-2010
- 2nd iteration 2010-2016 - Today's focus

$$(1'') \quad \left(\frac{\partial \psi}{\partial x}\right)^2 + \left(\frac{\partial \psi}{\partial y}\right)^2 + \left(\frac{\partial \psi}{\partial z}\right)^2 - \frac{2m}{K^2} \left(E + \frac{e^2}{r}\right) \psi^2 = 0 .$$

$$r = \sqrt{x^2 + y^2 + z^2} .$$

Und unser Variationsproblem lautet

$$(3) \quad \delta J = \delta \iiint dx dy dz \left[ \left(\frac{\partial \psi}{\partial x}\right)^2 + \left(\frac{\partial \psi}{\partial y}\right)^2 + \left(\frac{\partial \psi}{\partial z}\right)^2 - \frac{2m}{K^2} \left(E + \frac{e^2}{r}\right) \psi^2 \right] = 0 ,$$

das Integral erstreckt über den ganzen Raum. Man findet daraus in gewohnter Weise

$$(4) \quad \left\{ \begin{array}{l} \frac{1}{2} \delta J = \int df \delta \psi \frac{\partial \psi}{\partial n} - \iiint dx dy dz \delta \psi \left[ \Delta \psi + \right. \\ \left. + \frac{2m}{K^2} \left(E + \frac{e^2}{r}\right) \psi \right] = 0 . \end{array} \right.$$

Es muß also erstens

$$(5) \quad \Delta \psi + \frac{2m}{K^2} \left(E + \frac{e^2}{r}\right) \psi = 0$$

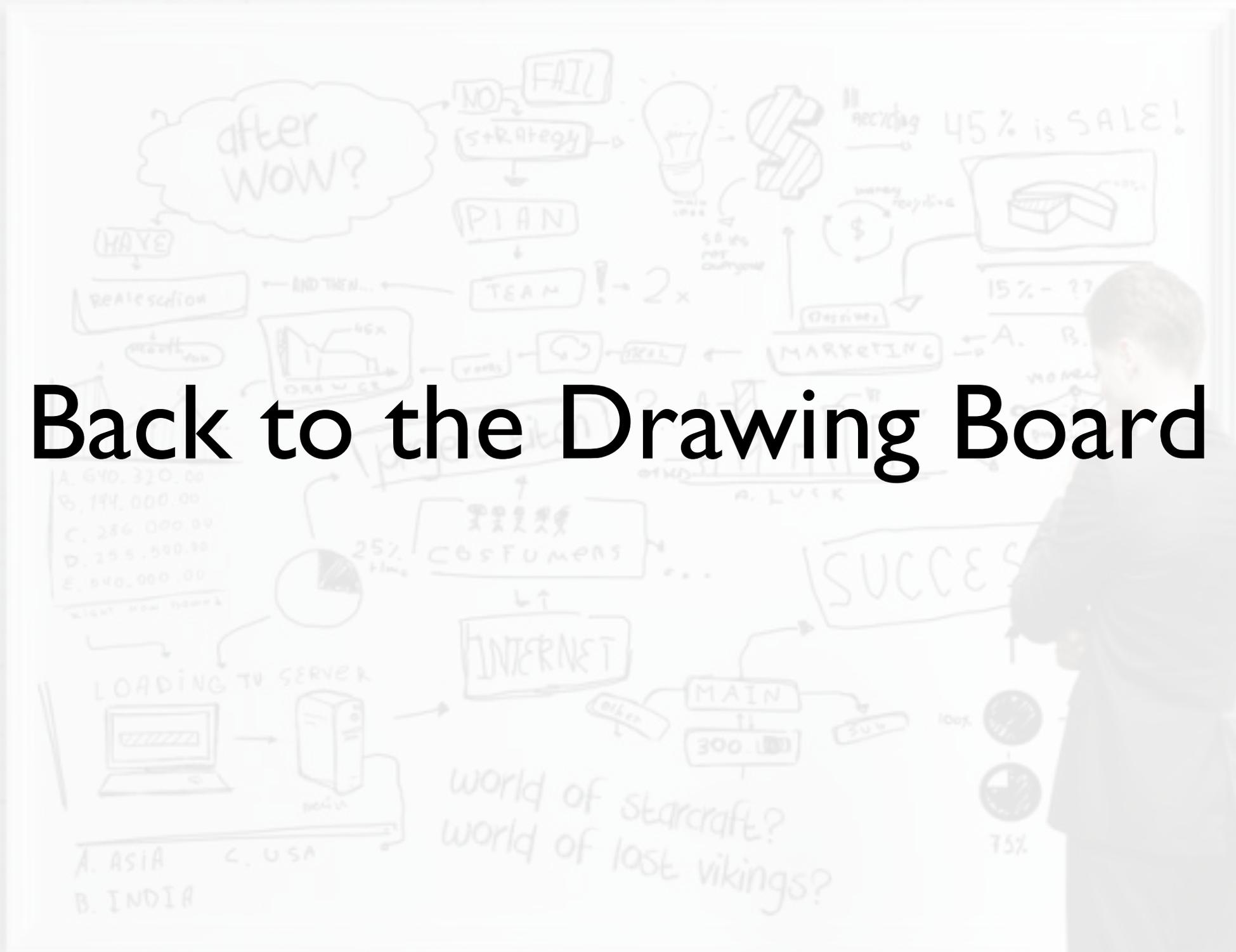
# First Iteration Concept

- Two different languages
  - One for embedded controllers (P-FRP)
    - Discrete events & strictly terminating
  - One for physical world outside
    - Time derivatives, partial derivatives, and undirected equations.

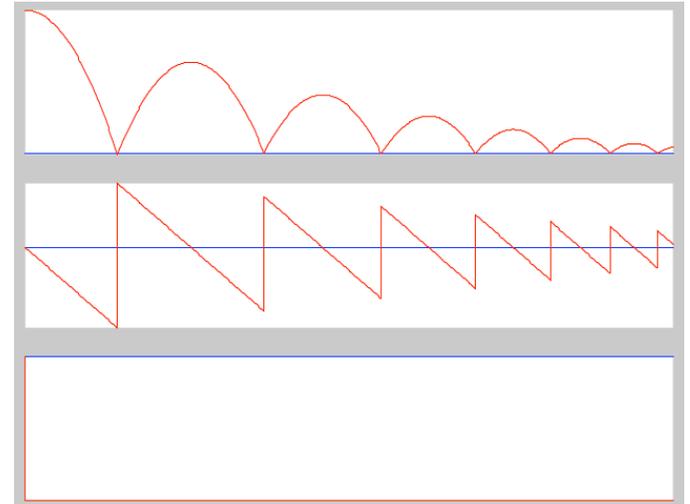
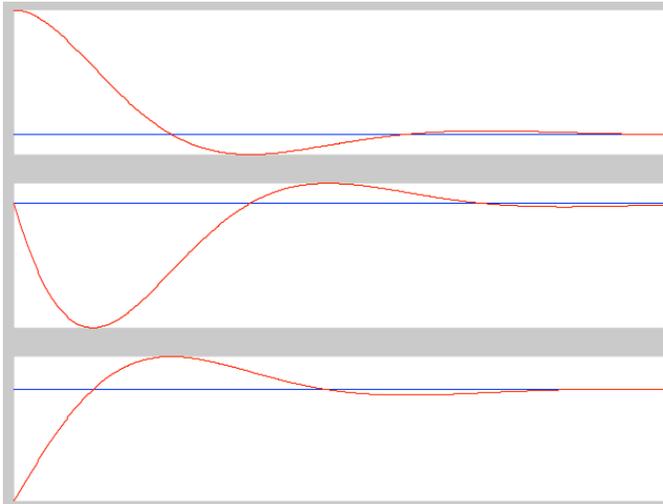
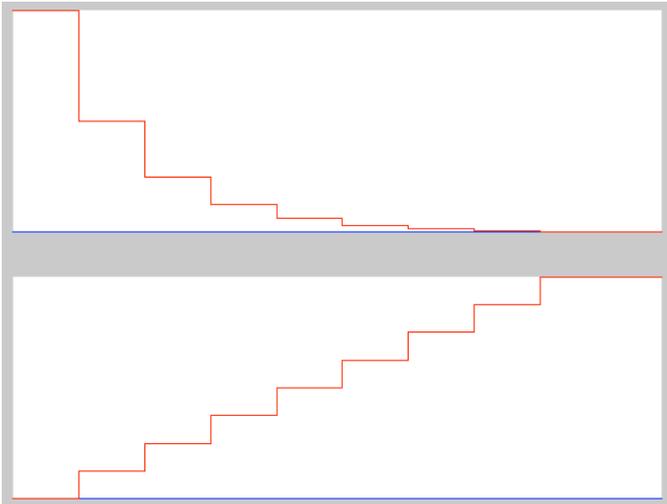
# 1st Iteration Concept

- After design, several interactions with
  - practitioners
  - other researchers
- Emerging questions:
  - “Why this particular controller language?”
  - “What if we want a continuous controller?”

# Back to the Drawing Board



# Discrete+Continuous=Hybrid



```
if x=>2 then  
  x+ = x/2,  
  n+ = n+1  
noelse
```

$$x'' = -x - x'$$

```
if (x<0 && x'<0) then  
  x'+ = -0.8*x'  
else  
  x'' = -10
```

# Minimal Core Calculi

- MicroAcumen Disjoint guards, no nesting
  - Deno. Sem. (Moggi, Taha, Bartha)
  - Oper. Sem. (Duracz, Taha, Moggi, Bartha)
- MiniAcumen Adds nesting of conditionals
  - Quadratically shorter (Duracz, Taha)
- More on technical details later

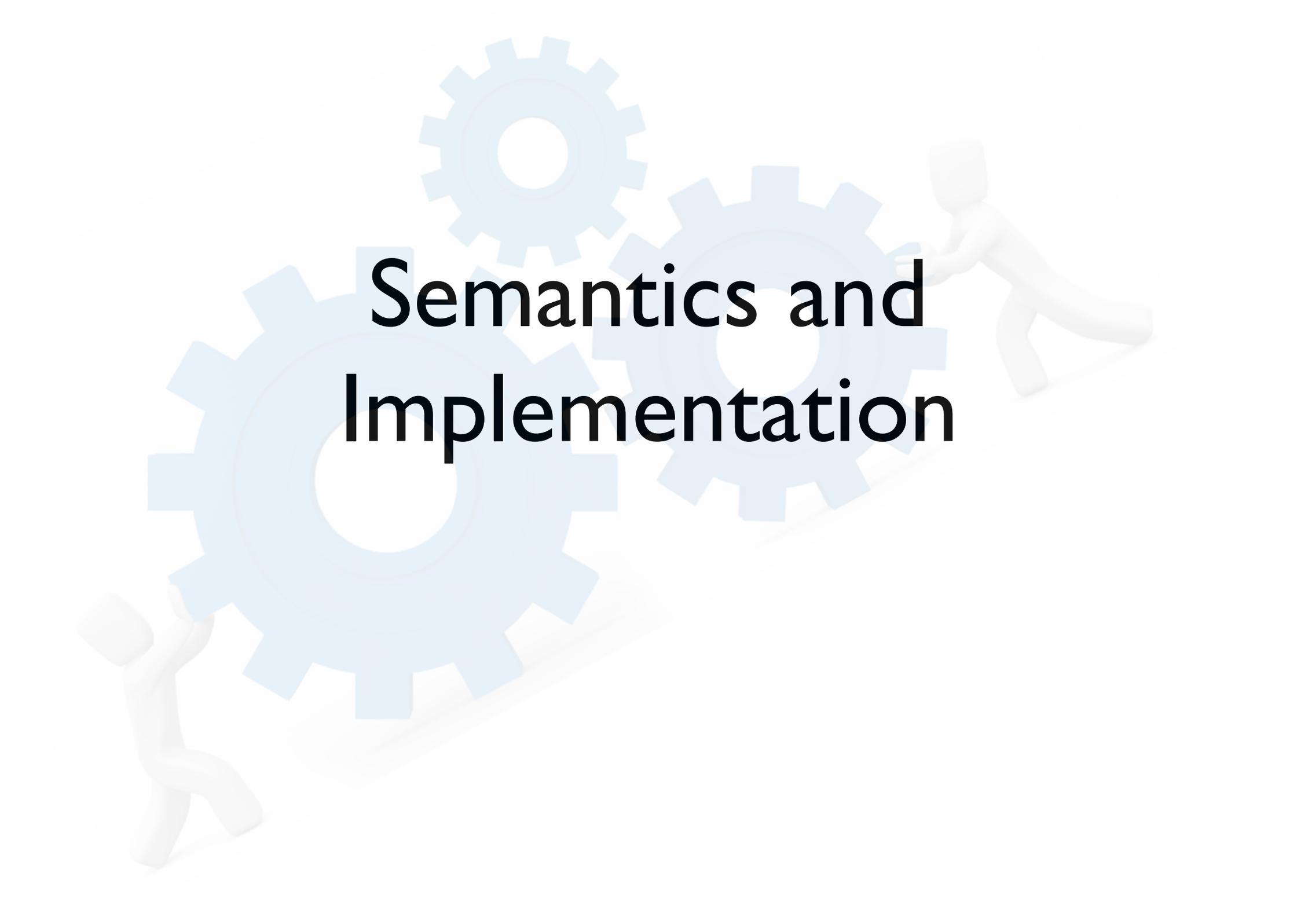
# Conservative Extensions

- Binding Time Analysis (BTA) and Automatic Differentiation enable two key extensions:
  - Equational constraints
  - Partial derivatives
- The combination is highly expressive
- Key achievement: Significant static checking and modular user feedback (Zeng, Taha)

# Expressivity by Example

- Continuous plant, discrete controller
- A mechanical cam
- Exoskeleton arms (w O'Malley)
- Mechanics of bipedal robots (w Ames)
- Bouncing/sliding ball on a continuously varying surface

# Semantics and Implementation



# Symptoms and Problems

- Symptoms pointing to semantics questions
  - MATLAB/Simulink goes into infinite loop
  - Modelica semantics impl. dependent
- Problems
  - Hybrid systems introduce new pathologies
  - Unlike software and hardware, simulation cannot even be used for falsification

# Pieces of the Puzzle

- Validated Numerics
  - Techniques for computing results with rigorous (guaranteed) bounds
  - Available today mainly as "solver" libraries
- Reachability Analysis
  - Most use symbolics, some val. numerics
  - Don't deal well with pathologies

# Hybrid Systems Pathologies

- Two main types:
  - Zeno: Infinite events in “finite” time
  - Chattering: Infinite events in zero time
- Cause simulators to either loop infinitely or produce incorrect results
- Cause reachability tools AND semantics to produce incorrect results

# Rigorous Simulation of Zeno systems

- Root cause of problem:
  - Simulators AND semantics use "events" as simulation "steps"
- Insight: These pathologies involve infinite events, but they are often "inconsequential"
- Validated methods are about sets, which facilitates checking fixed points (Konecny, Taha, Bartha, Duracz, Duracz, Ames)

A group of seven construction workers wearing hard hats and safety vests are gathered on a job site, reviewing plans. The scene is outdoors with a clear sky and a flat, open landscape in the background. The workers are dressed in various work clothes, including jackets and pants. One worker in the foreground is kneeling, while others are standing around him, some holding large sheets of paper, likely blueprints. The overall atmosphere is one of collaborative work and site evaluation.

# Evaluating Emerging Design

# Second Iteration

- Sources of user feedback and observation
  - Used in a course on CPS for five years
  - Early domain expert feedback
  - Case study on advanced driving functions
  - Benchmarking support for robust design
- We say a bit more about the last two

# Automotive Case Study

The screenshot displays the Acumen simulation environment. The main window shows a 3D plot of a road intersection with a red car and a blue truck. The code editor on the left contains the following code:

```
Braking_or_Collision_2.acm
rot2' = 0,
x2' = 0,
y2'' = a2*u2a
else
state2+ = "4-Stopped"
| "4-Stopped" ->
rot2' = 0,
x2' = 0, y2' = 0
],
],

/* Compute when the sensor detects V2,
detection is assumed during collision though it is
sense = caseSenseA || caseSenseB || caseSenseC || cas
|| caseSenseNR,

// front of the sensor intersects the left side of V2
caseSenseA = ((fly2 - frys) * (rly2 - frys) <= 0)
```

The console output shows the simulation results:

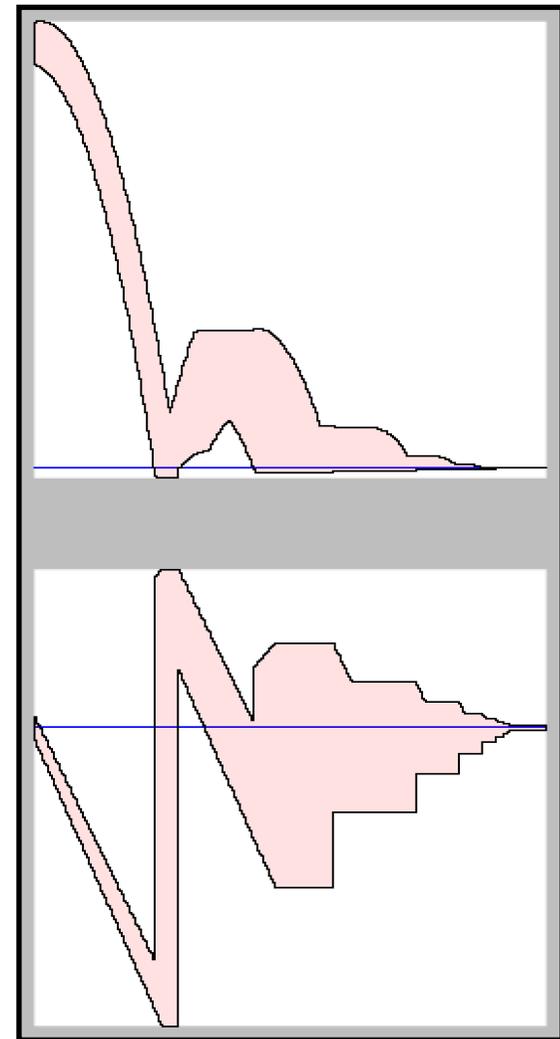
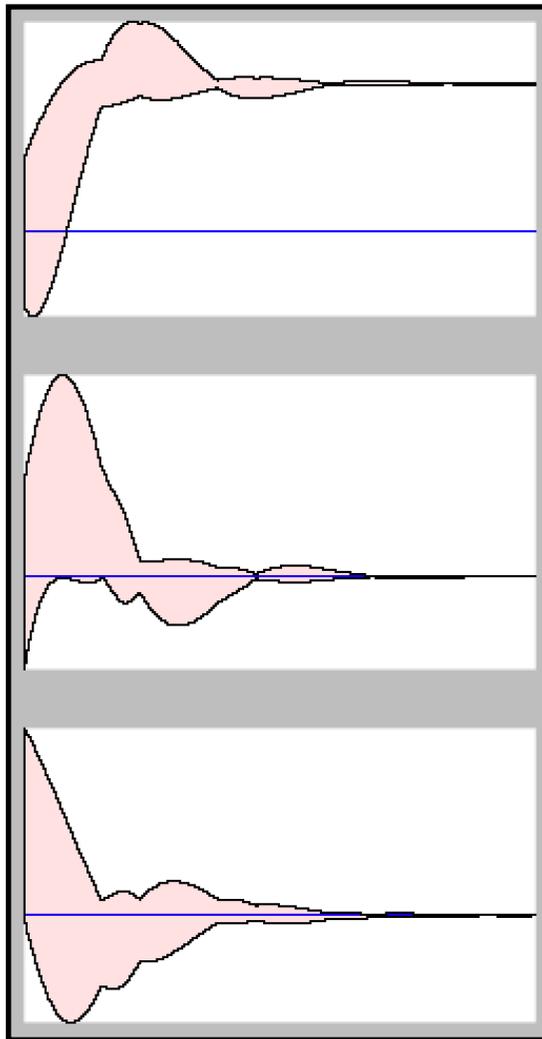
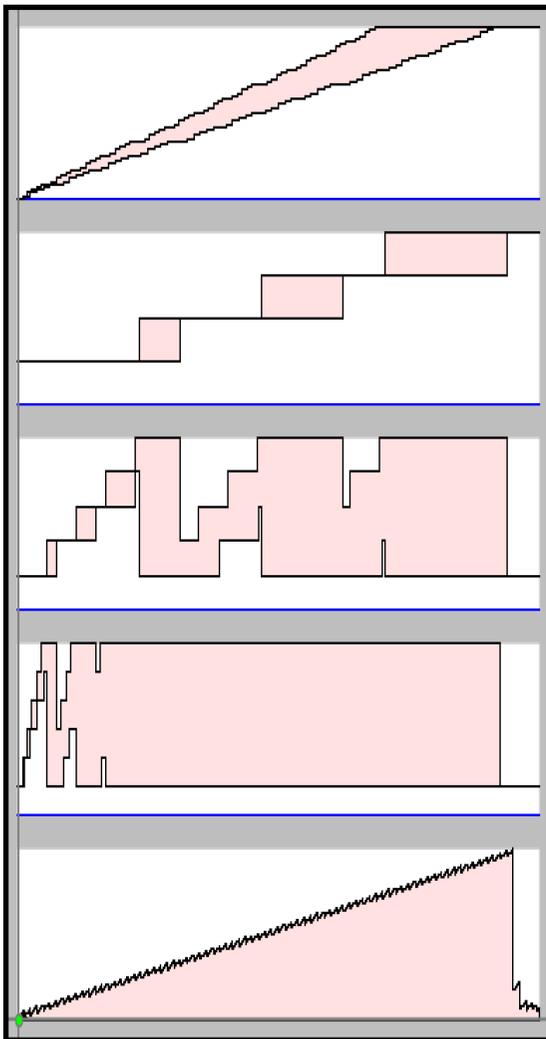
```
Stopped.
NO HYPOTHESES FALSIFIED OVER [0.0..10.0]
1 TRUE, 0 FALSE, 0 INCONCLUSIVE
```

The bottom console window shows the final results:

```
Stopped.
ALL HYPOTHESES PROVED OVER [0.0..2.3935546875]
1 TRUE, 0 FALSE, 0 INCONCLUSIVE
+ (#0:Main) 'Collision cases' Proved
Time to run simulation: 63.297s
```

The 3D plot includes a camera view with coordinates: Camera X: 12.26, Y: 23.78, Z: 10.76; Look At X: 1.26, Y: 0.79, Z: 1.44. The plot also shows a progress bar and a speed control slider set to 1.0x.

# Support for Robust Design



# Ongoing & Future Work

- Key areas for for third iteration are:
  - Soundness of op./deno. Semantics
    - & of core validated numerics algorithms
  - Minimization (of scalars and functions)
  - Rigorous bounds on prob. distributions

# Publications & Acumen

- All papers available online
  - [www.effective-modeling.org](http://www.effective-modeling.org)
- Acumen open source code:
  - [www.acumen-language.org](http://www.acumen-language.org)
- Most recent dissertation from group:
  - [bit.ly/adam-thesis](http://bit.ly/adam-thesis)

# Conclusions

- Rigorous simulation
  - addresses foundational problems
  - provides fully automated proofs
    - about behavior of hybrid systems
    - which are useful models of many CPSs
- Open source implementation (Acumen)