Enclosing Hybrid Behavior

Walid Taha, Halmstad University and Rice University
The Effective Modeling Group (EMG):
Adam Duracz, Yingfu Zeng, Chad Rose, Kevin Atkinson, Jan Duracz, Jawad Masood, Paul Brauner, Corky Cartwright, Marcie O'Malley, Roland Philippsen, Aaron Ames, Michal Konecny, and Veronica Gaspes from Halmstad, Rice, Texas A&M, and Aston.

cps-vo.org

What is innovation?

Idea ← Requirements

↓

Model

↓

Prototype

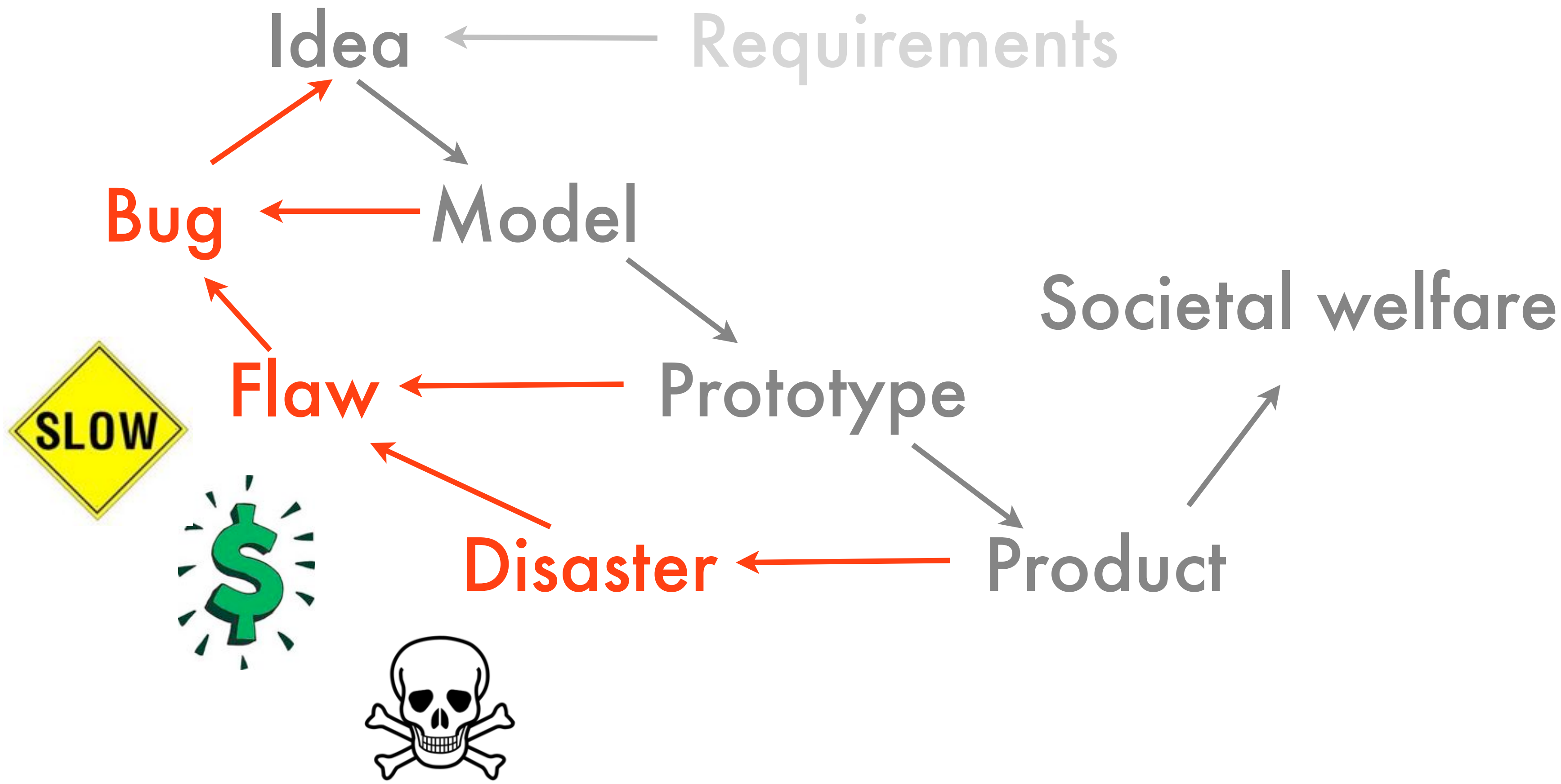
↓

Product

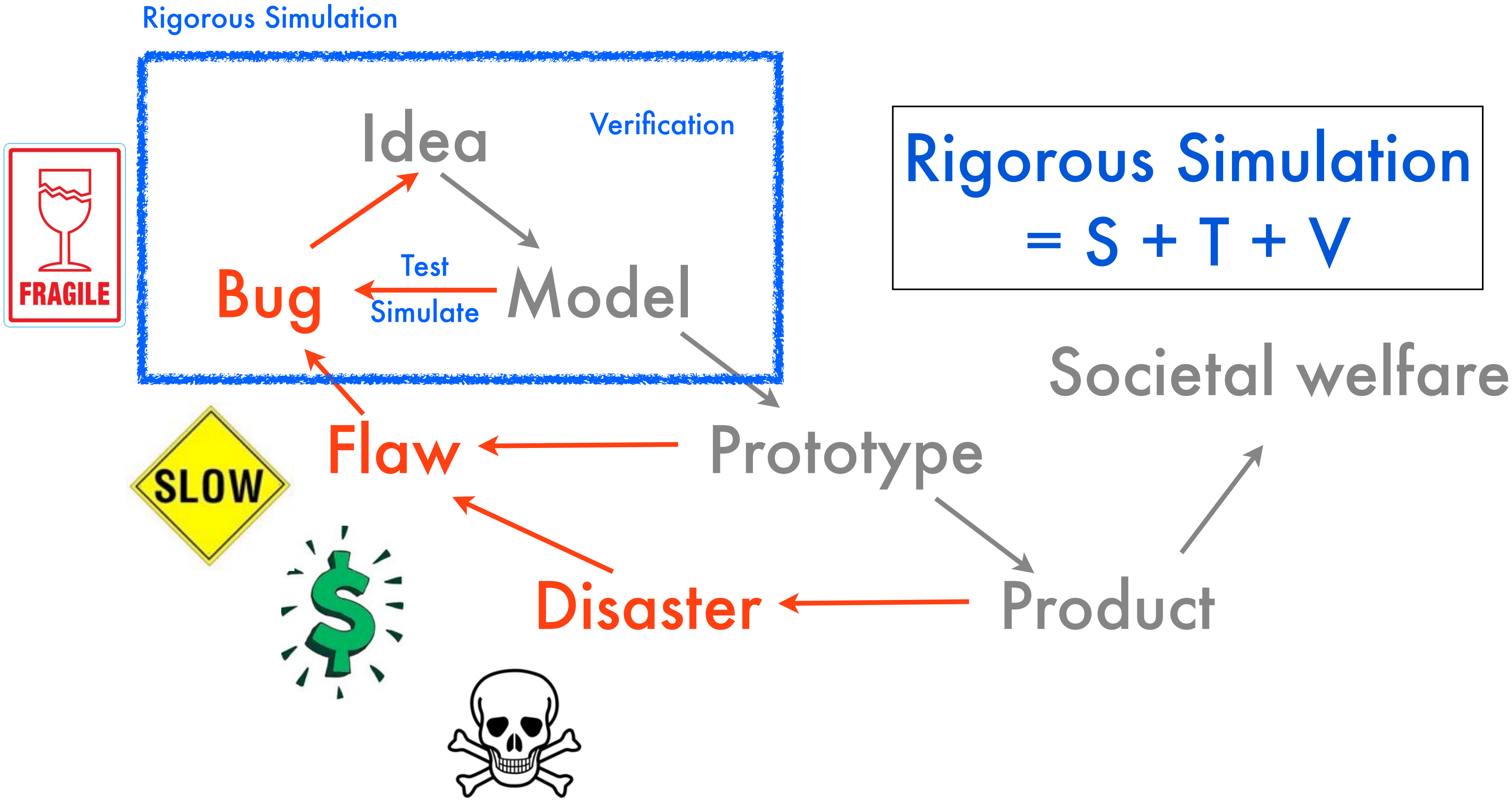
↓

Societal welfare

Innovation theory

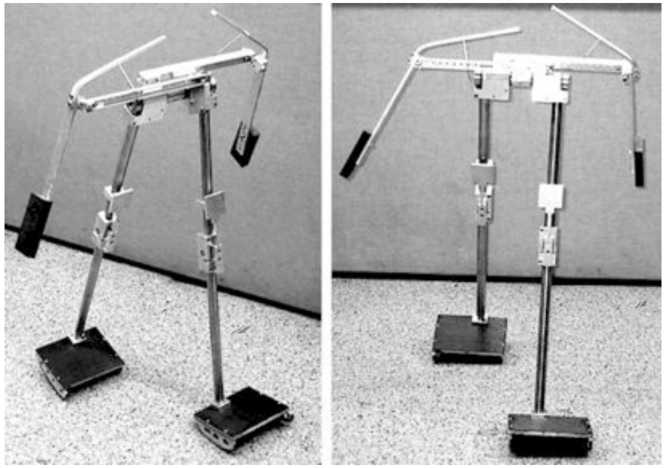


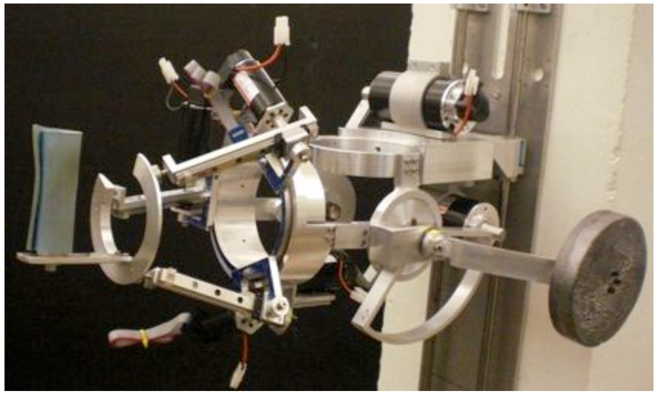
Rigorous Simulation



Rigorous Simulation
$= S + T + V$

Robot design





Traditional simulation

- No guarantee that behavior computed is consistent with model used.
  - Numerical artifacts
  - Integration drift
  - Singularities often ignored
  - Zeno behavior

Related Fields: Strengths

- Simulink: **Lots of models (IP)**
- Mathematica, Maple: **Symbolic alg.**
- ML, Haskell: **Expressivity, concurrency**
- HA, e.g. CHARON: **Formal proof**
- Modelica, Verilog-AMS: **Equations**

Related Fields: Weakness

- Simulink: **Numeric semantics...**
- Mathematica, Maple: **Req. closed form**
- ML, Haskell: “**Indeterminism**”
- HA, e.g. CHARON: **Scalability**
- Modelica, Verilog-AMS: **Semantics...**

Rest of this talk

- Enclosure methods
- Enclosing continuous behaviors
- Enclosing hybrid systems
  - Basic theory, and dealing with Zeno
- Remark on industrial applications
- Current activity: BTA

Idea: Enclosure methods

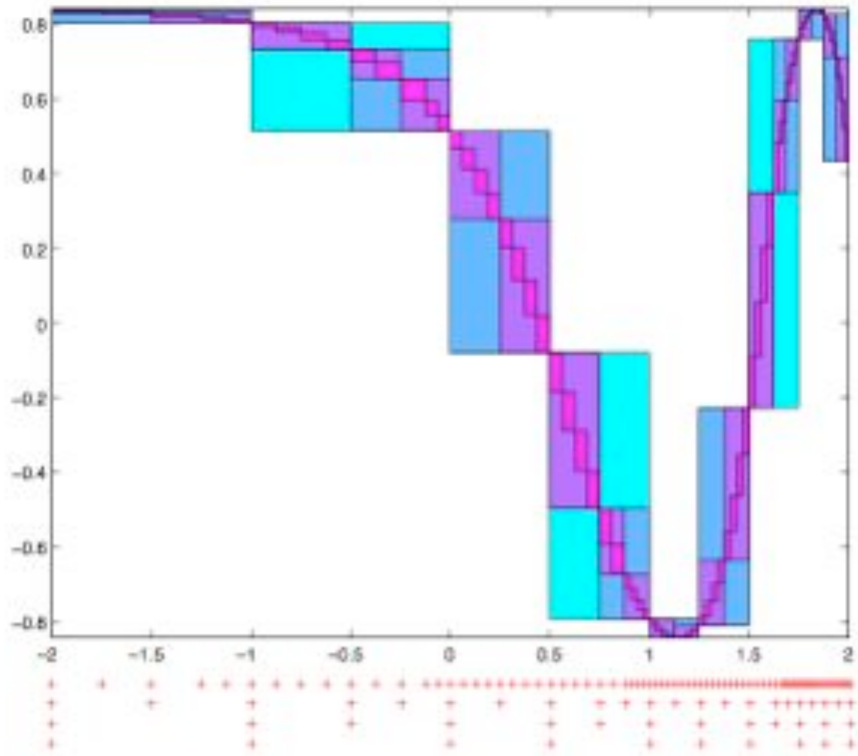


Figure 5.14: Refined enclosures of the integrand $f(x) = \sin(\cos(e^x))$.

- Always guarantee that solution is enclosed
- Can compute more precise answers as needed
- But can they be mechanized?

Continuous behaviors

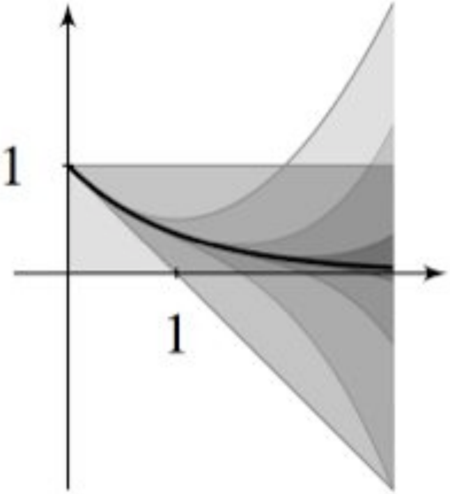
- An elegant, very general method exists:
  - Picard iteration
- Key challenge: Extending to proper enclosures

Set

$$\varphi_0(t) = y_0$$

and

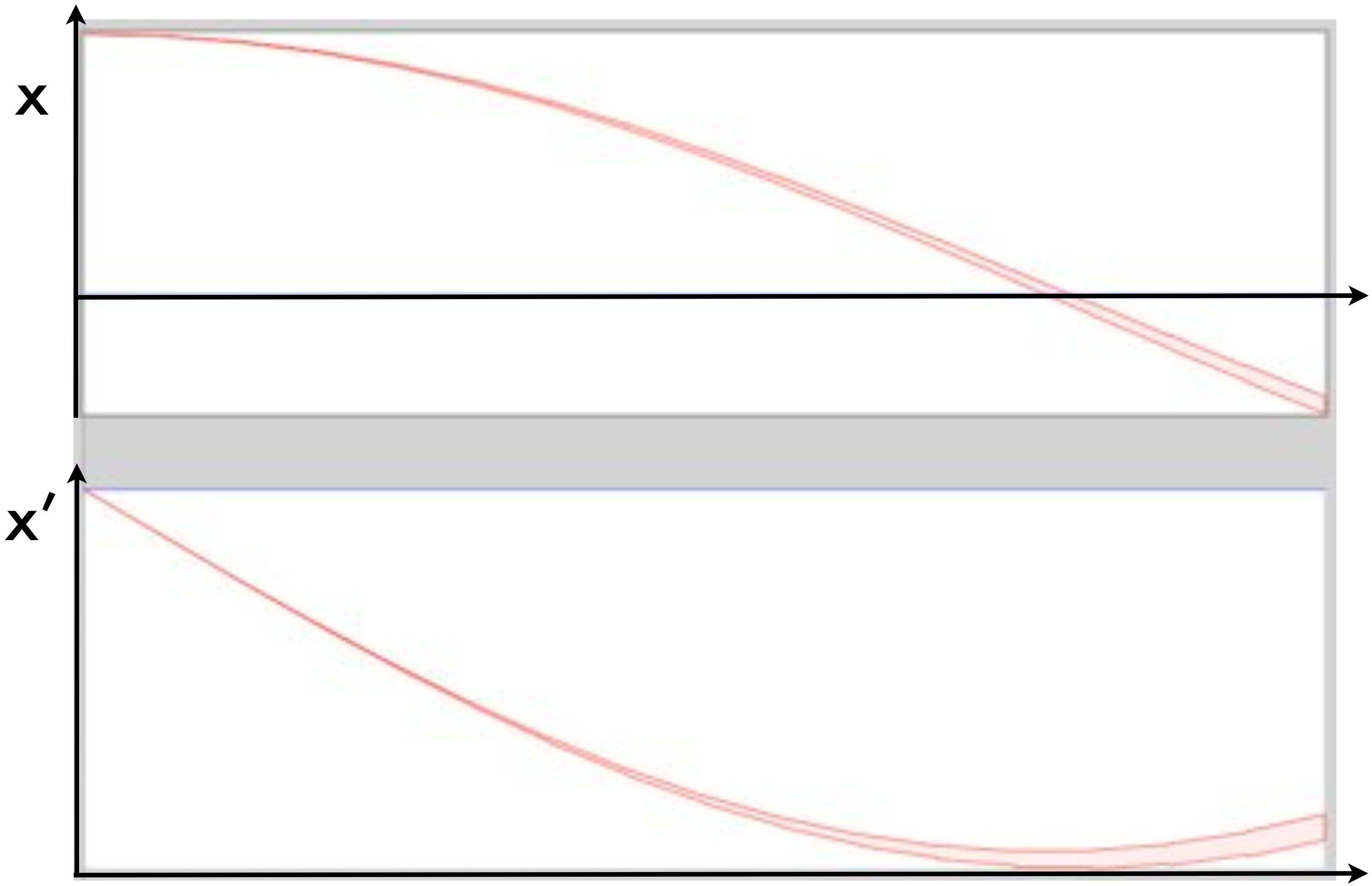
$$\varphi_{k+1}(t) = y_0 + \int_{t_0}^{t} f(s, \varphi_k(s))\, ds.$$



Example

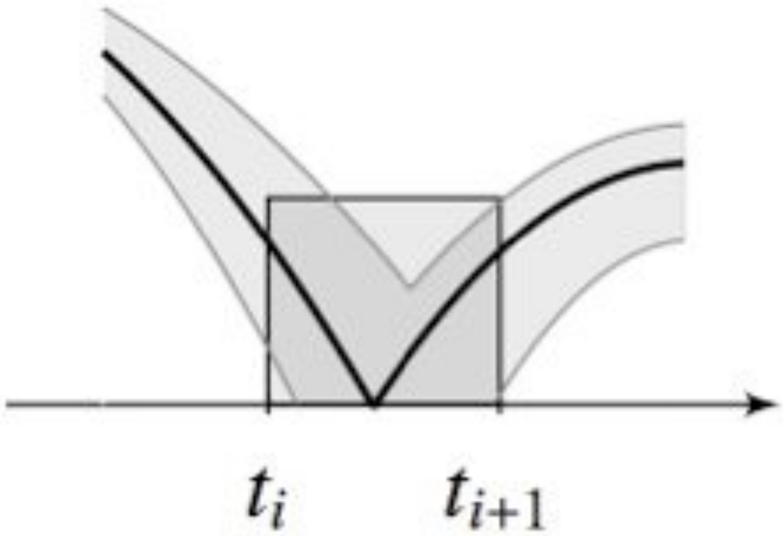
```
class Main(simulator)
private x:=1; x':=0; x'':=0; mode := ""; end
  switch mode
    case ""
      x'' = -x
  end;
  simulator.endTime := 2.0;
  simulator.minSolverStep := 0.1;
end
```

Example



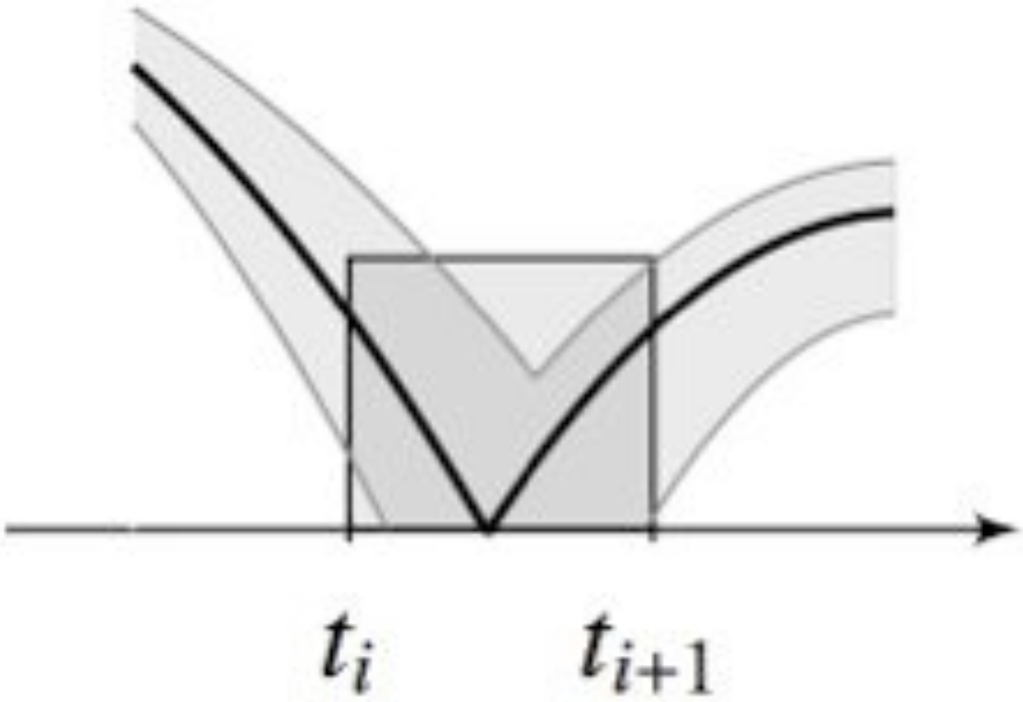
Event detection

- Enclosures provide a natural method for event detection (root find)
- Basic idea:
  - Mean value theorem
  - It's OK to say "I don't know"



Reset maps

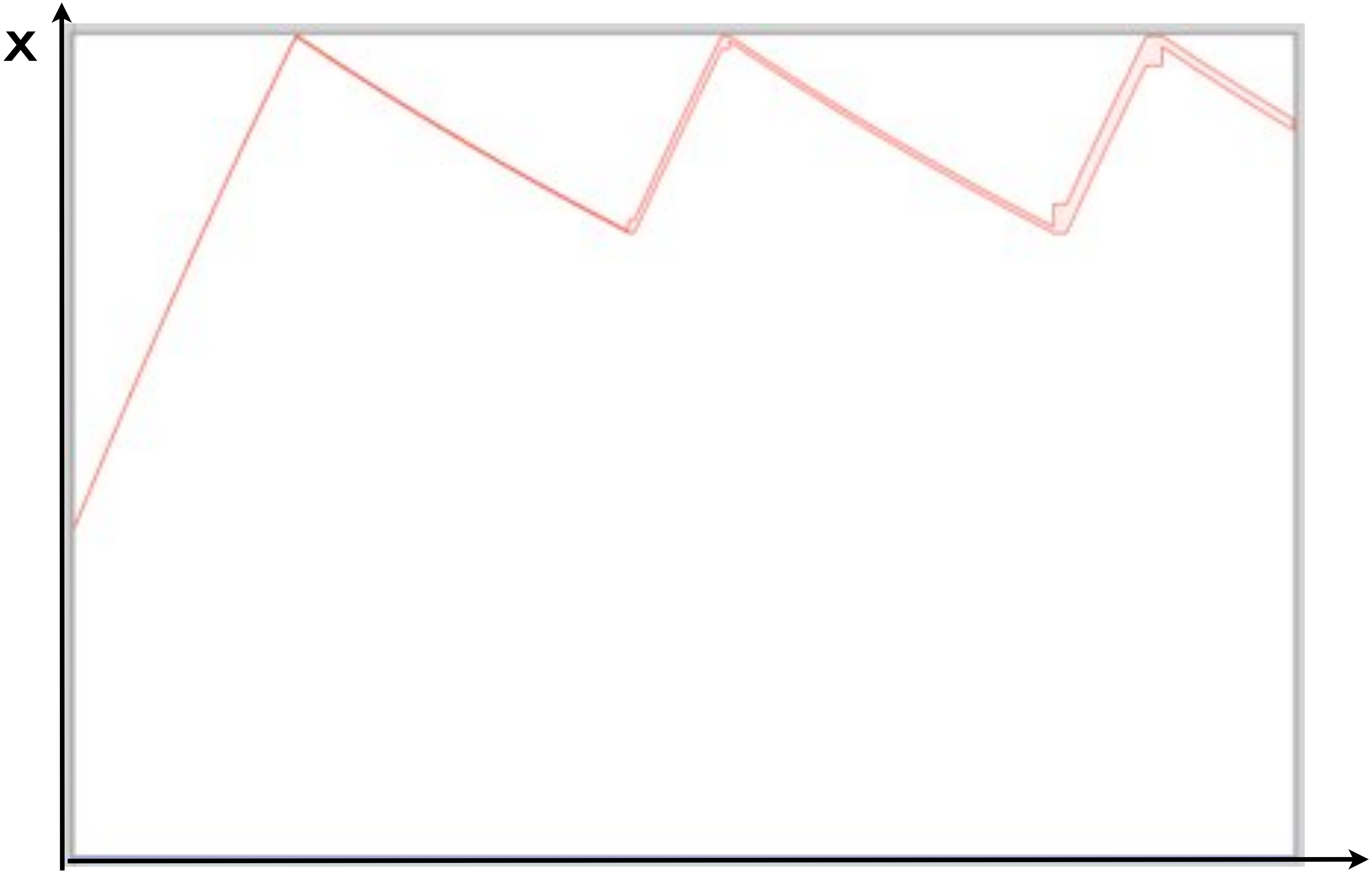
- Assume worst case behavior
- Note: Still need to know it was only *one* event that occurred in that interval



Example

```
class Main (simulator)
  private
    mode := "on"; x := 10; x' := 0;
  end
  switch mode
    case "on" require x <= 25
      if x == 25
        mode := "off"
      end;
      x' = 100 - x;
    case "off" require x >= 19
      if x == 19
        mode := "on"
      end;
      x' = - x;
  end;
  simulator.endTime := 1;
  simulator.minSolverStep := 0.01;
  simulator.minLocalizationStep := 0.001;
  simulator.minComputationImprovement := 0;
end
```

Example

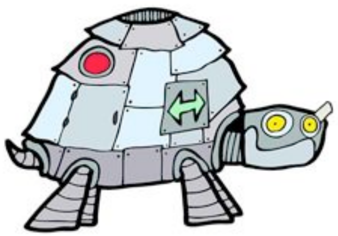


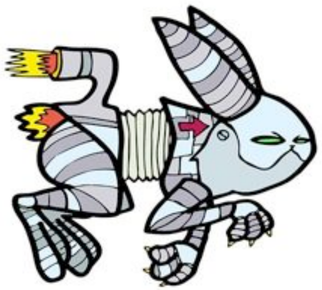
A bouncing ball

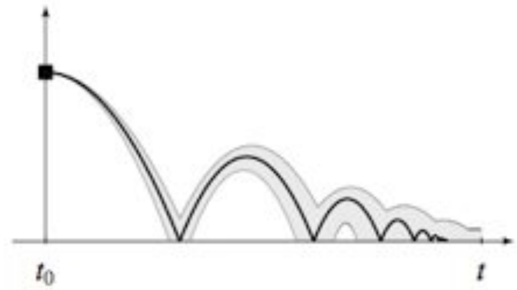
```
class Main(simulator)
  private
    mode := "Fly";
    x    := 5;
    x'   := 0;
    x''  := 0;
  end
  switch mode
    case "Fly"
      if x == 0 && x' <= 0
        x' := -0.5*x';
        mode := "Fly";
      end;
      x'' = -10;
  end;
  simulator.endTime := 4.5;
  simulator.minSolverStep := 0.01;
  simulator.minLocalizationStep := 0.01;
  simulator.minComputationImprovement := 0.001;
end
```

Zeno Behavior

- A real problem for rigid body dynamics *with impacts*



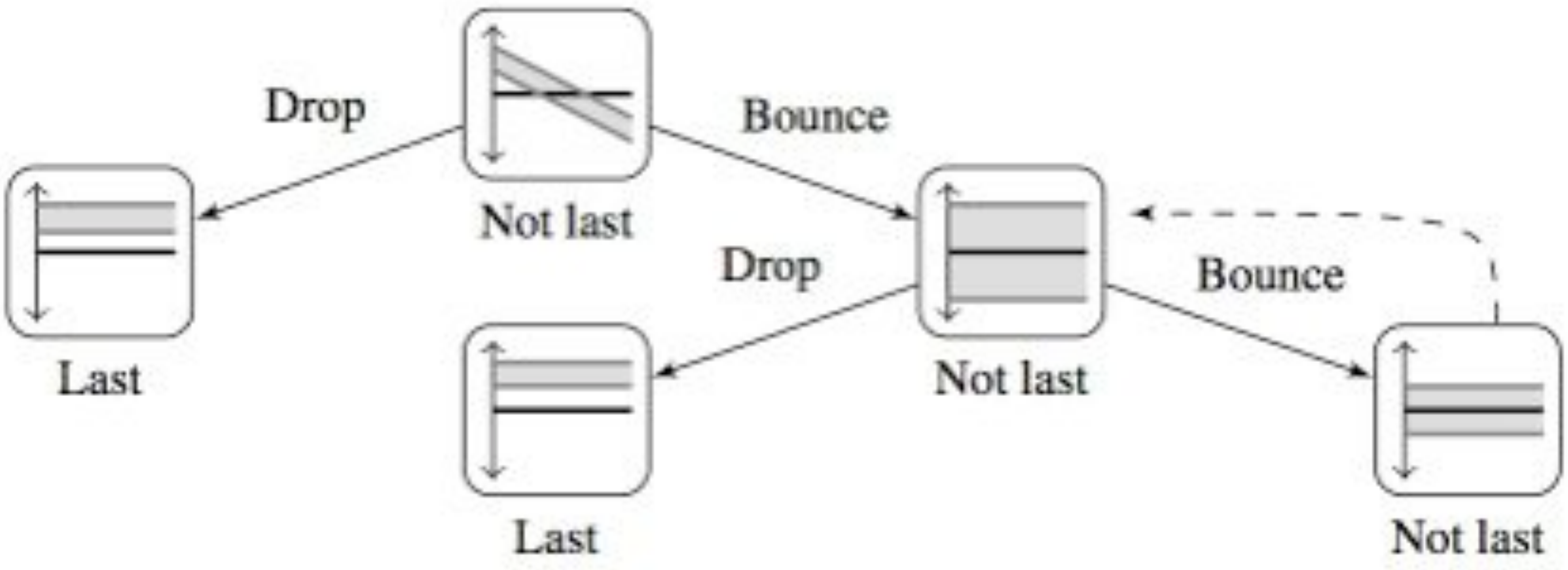




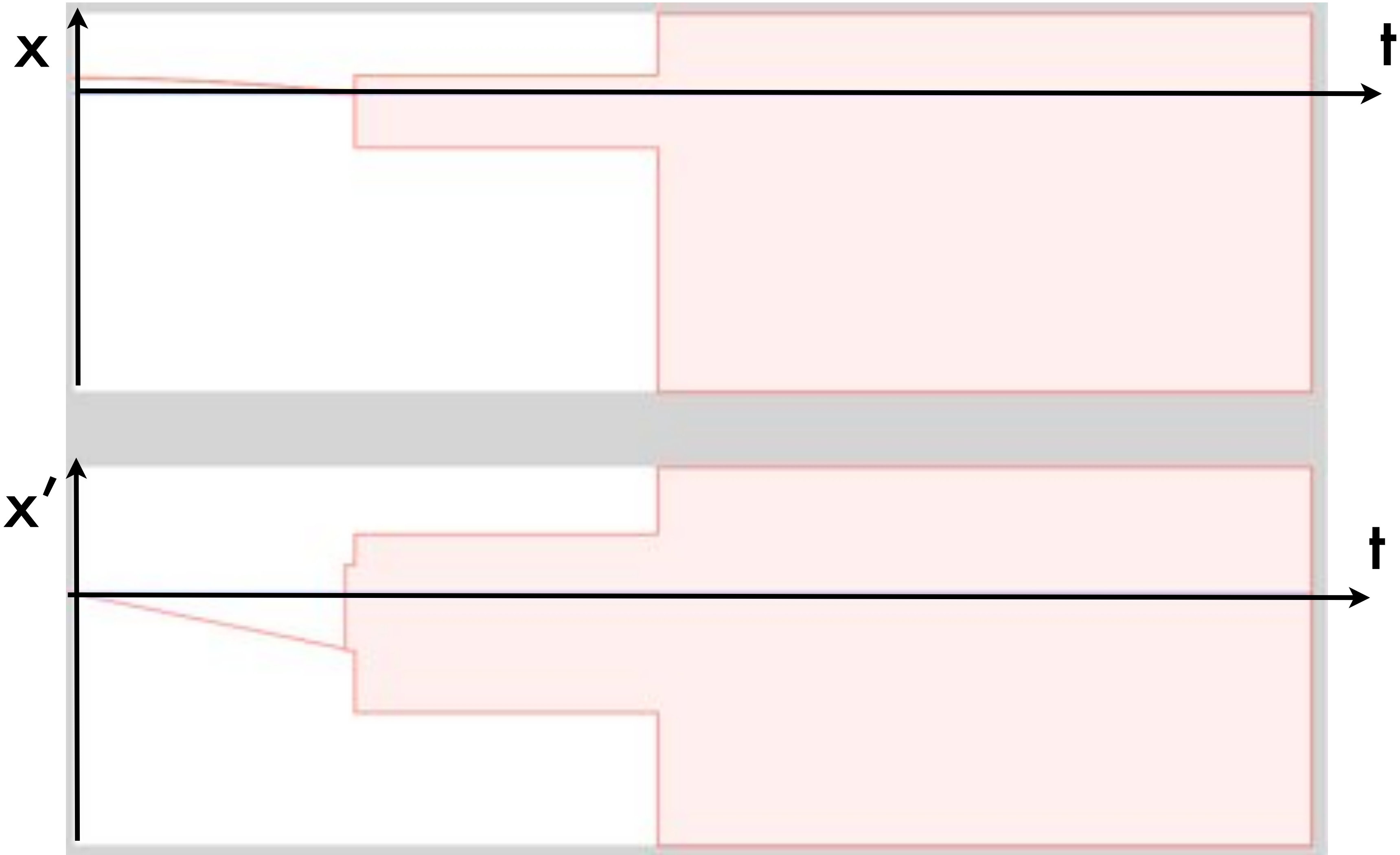
- A bouncing ball comes to rest in finite time, but it does so with an *infinite* number of bounce events!

Enclosing Zeno

- Idea: We can actually relax that requirement if we know that a repeat event does NOT enlarge the enclosure we start with



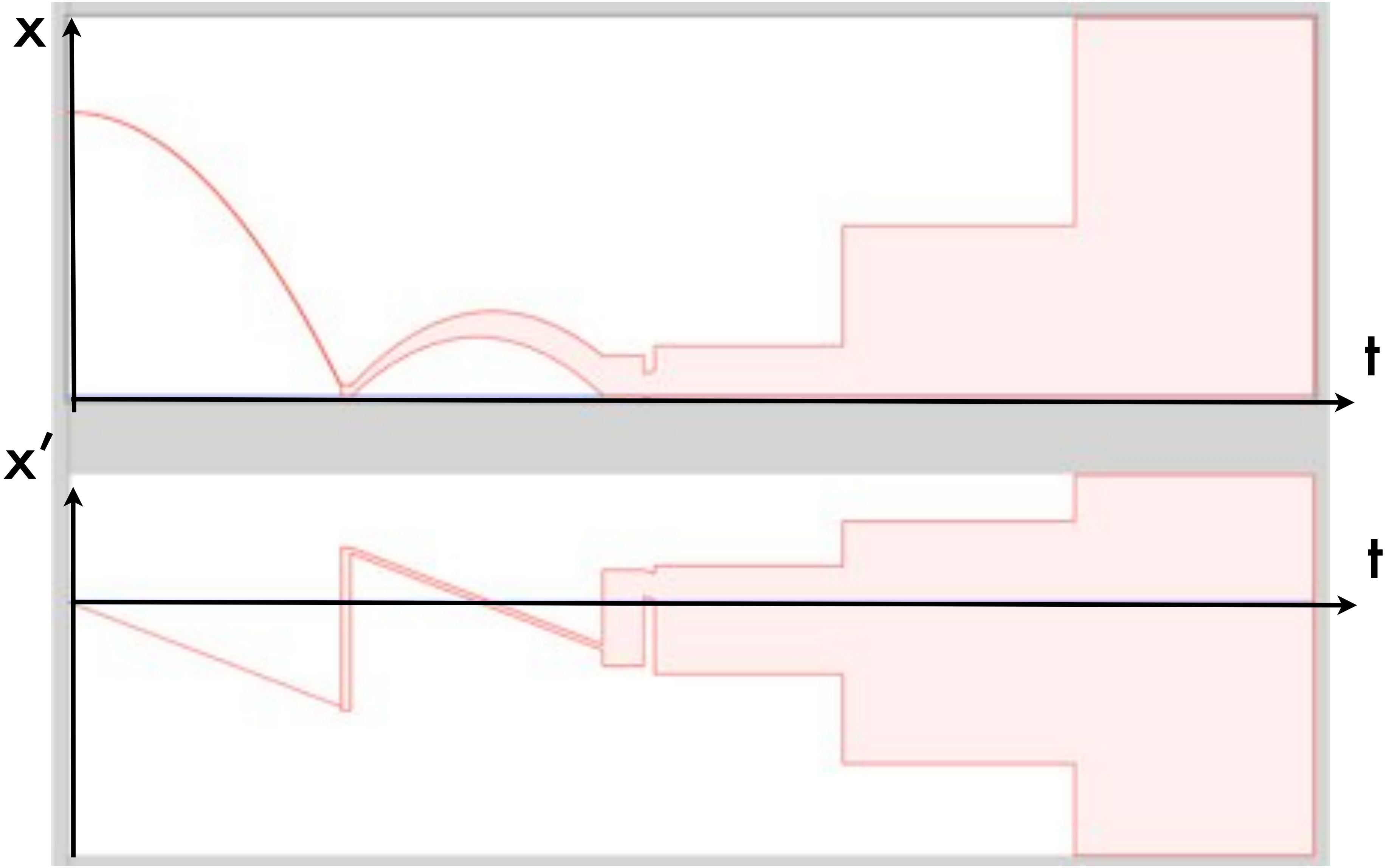
Enclosing Zeno, Take I



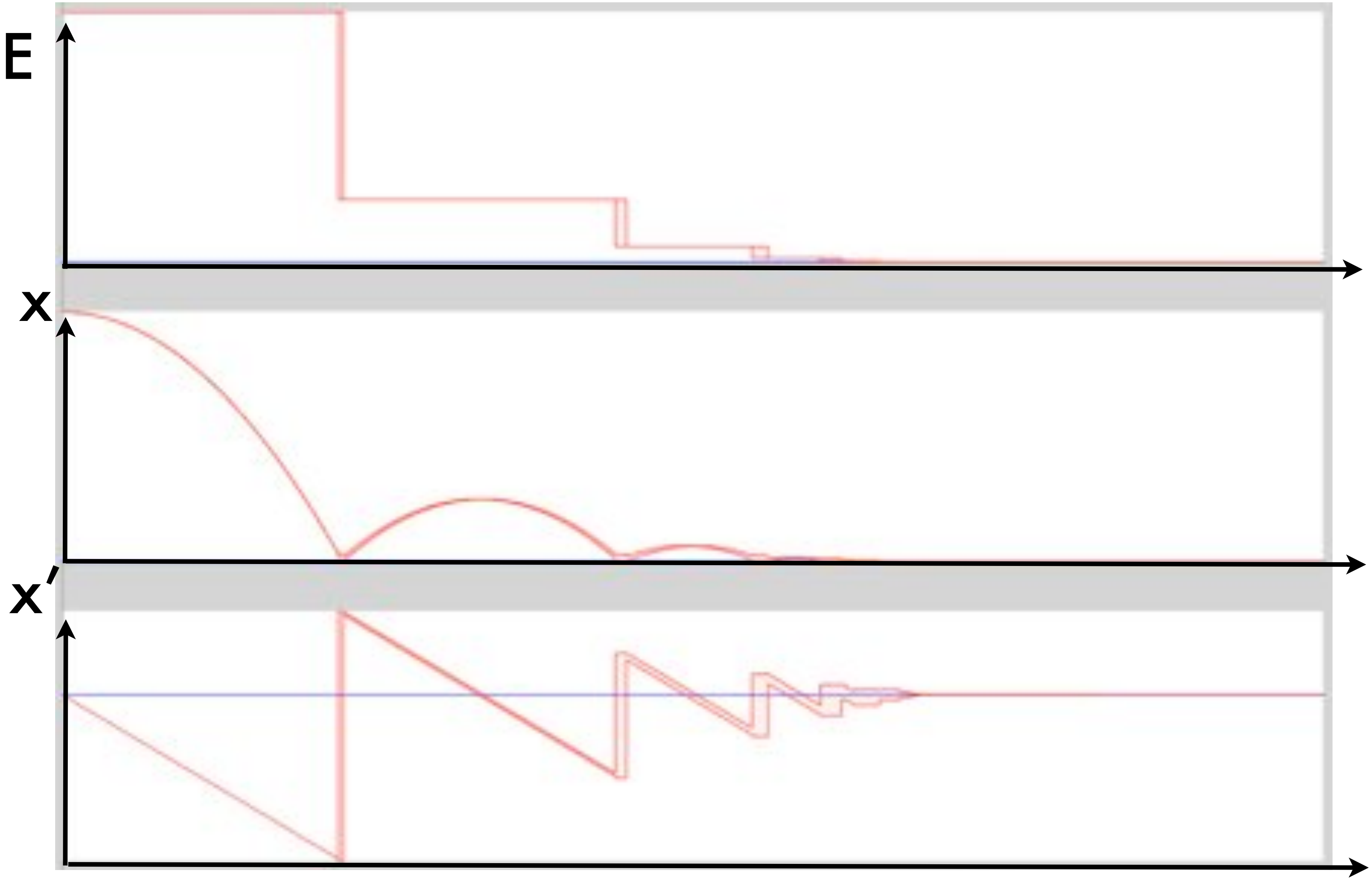
Fix: Over-constraining

- Enforce domain constraints (intersect)
  - Example: x >= 0
- Constraining speed based on explicit energy
  - Example: A notion of energy

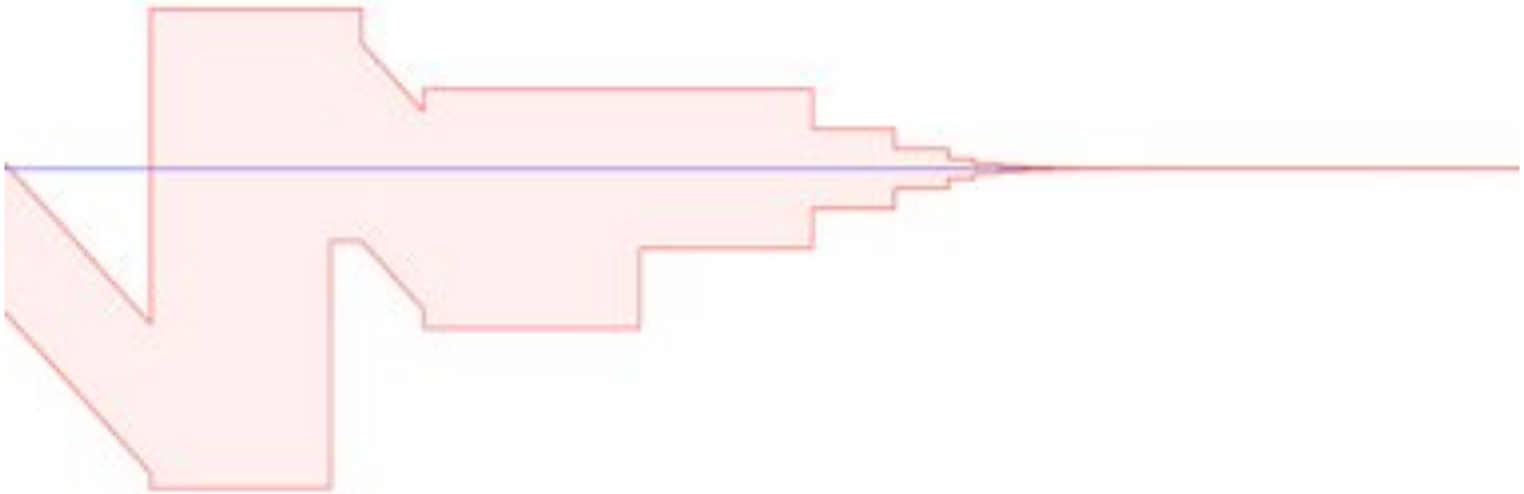
Enclosing Zeno, Take II



Enclosing Zeno, Take III



Empire State Building



CPSNA 2013
The 1st IEEE International Conference on Cyber-Physical Systems, Networks, and Applications

Taipei, Taiwan **August 19 - 20, 2013**

Enclosing the Behavior of a Hybrid System up to and Beyond a Zeno Point

Michal Konečný[*], Walid Taha[†‡], Jan Duracz[†], Adam Duracz[†] and Aaro
[*]School of Engineering and Applied Science, Aston University, Birmingham, UK, Email:
[†]Halmstad University, Halmstad, Sweden, Email: Walid.Taha@hh.se, Jan.Duracz@hh.s
[‡]Computer Science Department, Rice University, Texas, USA
[§]Department of Electrical & Computer Engineering, Texas A&M University, USA,

Abstract—Even simple hybrid systems like the classic bouncing ball can exhibit Zeno behaviors. The existence of this type of behavior has so far forced simulators to either ignore some events or risk looping indefinitely. This in turn forces modelers to either insert ad hoc restrictions to circumvent Zeno behavior or to abandon hybrid modeling. To address this problem, we take a fresh look at event detection and localization. A key insight that emerges from this investigation is that an *enclosure* for a given time interval can be valid independently of the occurrence of a given event. Such an event can then even occur an unbounded number of times, thus making it possible to handle certain types of Zeno behavior.

systems over a decade
interesting patholog
lar behavior turns
between tradition
its existence can
tions and/or enteri
consideration in de
systems simulator i

This phenomenon
systems. It can oc
with impact constr

Technical results

- Proper interval Picard converges
- Event detection is sound
- Zeno method is sound

Next Generation Testing

Uncertainty-aware design

Activity in NG-Test

- Analysis of ISO 26262-3
- Defining high-level models of test scenarios
  - Vehicles, controls, sensors
- Using enclosures to establish bounds on severity of collisions
- Gradual model refinement is key

Conclusions

- Using enclosures
  - ensures that any answer produced is correct
  - simplifies correct event detection
  - admits an elegant way of handling certain classes of Zeno behavior
  - benefits from over-constraining

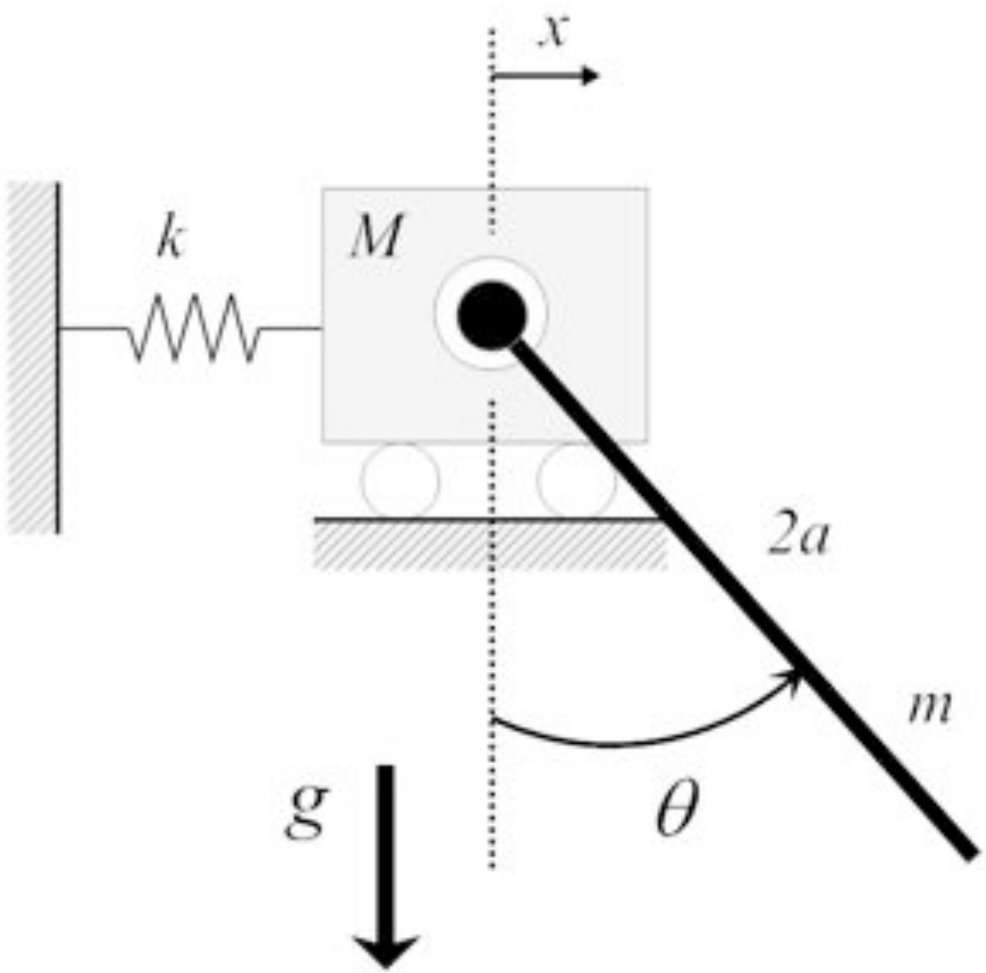
Future work

- Algorithmic complexity
- Performance on larger models (mainly drawn from the robotics domain)
- Methods to limit loss of precision during continuous segments
- Extending language expressivity

Whence BTA?

- Quick demo of tool
- Current implementation uses only subset, supporting only “directed equation”
- Extending expressivity requires adding:
  - “Static” partial derivatives
  - Converting DAEs to basic subset

Equational models



Equational models

$$q = [x, \theta] \quad a = 1 \quad m = 2 \quad M = 5$$

$$g = \frac{49}{5} \quad k = 2 \quad I = \frac{4}{3} m a^2$$

$$T = \frac{1}{2} (M + m) \dot{x}^2 + m a \dot{x} \dot{\theta} \cos(\theta) + \frac{2}{3} m a^2 \dot{\theta}^2$$

$$V = \frac{1}{2} k x^2 + m g a (1 - \cos(\theta)) \quad L = T - V$$

$$\forall i \in \dim(q) \quad \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = 0$$

(d) Pendulum/Mass

Equational models

$$\boxed{q} = \boxed{[x,\theta]} \quad a = 1 \quad m = 2 \quad M = 5$$

$$g = \frac{49}{5} \quad k = 2 \quad I = \frac{4}{3} m a^2$$

$$T = \frac{1}{2}(M+m)\,\dot{x}^2 + m a \dot{x}\dot{\theta}\cos(\theta) + \frac{2}{3} m a^2 \dot{\theta}^2$$

$$V = \frac{1}{2}kx^2 + mga\,(1-\cos(\theta)) \quad L = T - V$$

$$\boxed{\forall i \in \dim(q)} \quad \frac{d}{dt}\left(\frac{\partial L}{\partial \boxed{\dot{q}_i}}\right) - \frac{\partial L}{\partial \boxed{q_i}} = 0$$

(d) Pendulum/Mass

From Acumen to DAEs

$$q = [x, \theta] \quad a = 1 \quad m = 2 \quad M = 5 \quad g = 9.8 \quad k = 2$$

$$I = \frac{4}{3}ma^2 \quad T = \frac{1}{2}(M + m)\dot{x}^2 + ma\dot{x}\dot{\theta}\cos(\theta) + \frac{2}{3}ma^2\dot{\theta}^2$$

$$V = \frac{1}{2}kx^2 + mga(1 - \cos(\theta)) \quad L = T - V$$

$$\forall i \in \dim(q) \quad \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0$$

(a) Acumen Source for Pendulum/Mass Example

Defined: $q := [x, \theta] \quad a := 1 \ \ldots \quad I := \frac{4}{3}ma^2 \ \ldots$

Computed: $\forall i \in \dim(q) \quad \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0$

(b) After Defined Variable Analysis

From Acumen to DAEs

Defined: $q := [x, \theta] \quad a := 1 \ \ldots \quad I := \frac{4}{3} m a^2 \ \ldots$

Computed: $\forall i \in \dim(q) \quad \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0$

(b) After Defined Variable Analysis

Defined: $q := \boxed{[\boxed{x}, \boxed{\theta}]} \quad a := \boxed{1} \ \ldots \quad I := \boxed{\frac{4}{3} m a^2} \ \ldots$

Computed: $\boxed{\forall i \in \dim(q) \quad \boxed{\frac{d}{dt}\left(\frac{\partial L}{\partial \boxed{\dot{q}_i}}\right) - \frac{\partial L}{\partial \boxed{q_i}} = 0}}$

(c) After Binding-Time Analysis (BTA)

Whence BTA?

- Current activity:
  - Defining BTA rules
  - Defining BTA constraint language
  - Defining constraint generation
  - Defining constraint solving
  - Proving the soundness of it all

Thank you!

- Checkout
  - effective-modeling.org
  - acumen-language.org