In situ model-based learning in Pamela

Dynamic Object Language Labs Inc.

paulr@dollabs.com, *tmarble@dollabs.com*
Speakers: Paul Robertson and Prakash Manghwani and Tom Marble

2nd December 2016

Outline

- Introduction
  - the team
  - what is Pamela
  - architecture
  - philosophy
- Part 1: Learning about transitions
  - Hidden State - the biased coin problem
  - Temporal bounds
  - (noisy) State Transitions
- Part 2: Practical Considerations
  - Plant Interface
  - Short Demonstration
  - Pamela is open source; what you need to know

The Team

- Dynamic Object Language Labs:
  - Dr Paul Robertson
  - Dr Andreas Hofmann
  - Prakash Manghwani
  - Dan Cerys
- Consultants:
  - Tom Marble

A brief History and Acknowledgement

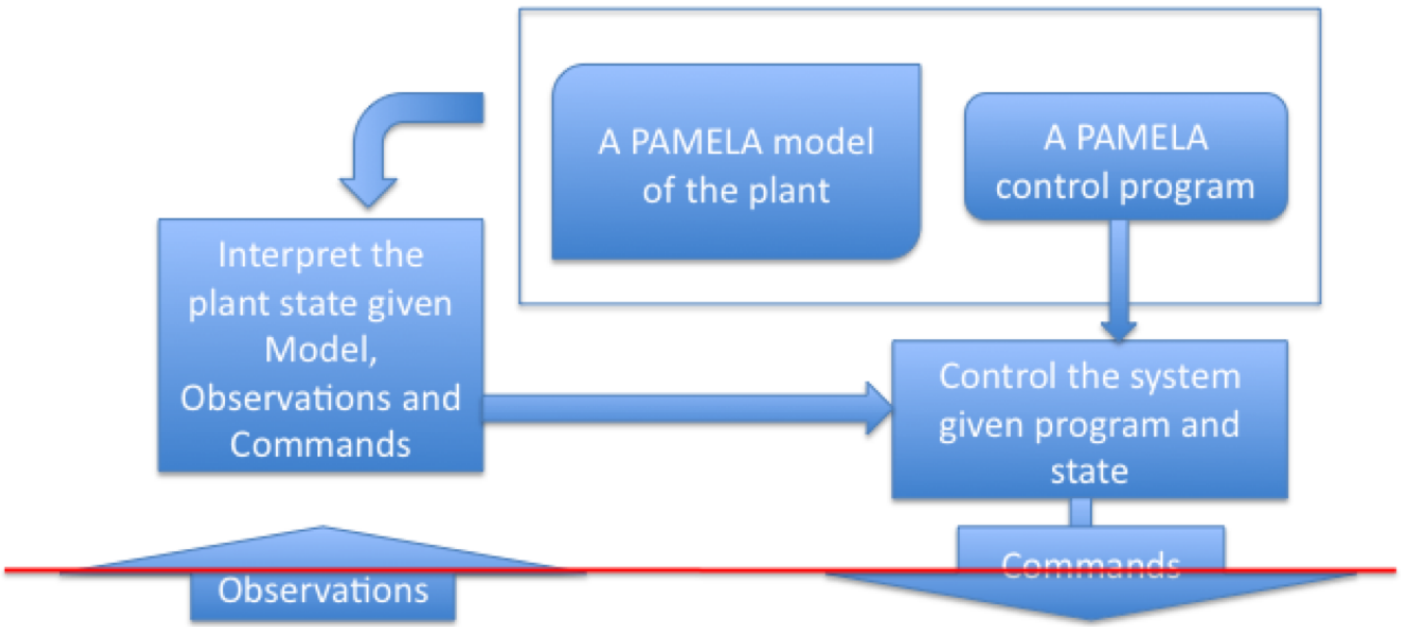
- Language antecedants: RMPL, MPL, Esterel, ESL, and PDDL.
- Lead innovators: Xerox Parc, NASA, MIT (Williams)
- Deployments: Deep Space 1 in 2001.
- Funding: DARPA:

This material is based upon work supported by the Army Contracting and DARPA under contract No. W911NF-15-C-0005. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Army Contracting Command and DARPA.
This work was supported by Contract FA8650-11-C-7191 with the US Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

What is Pamela

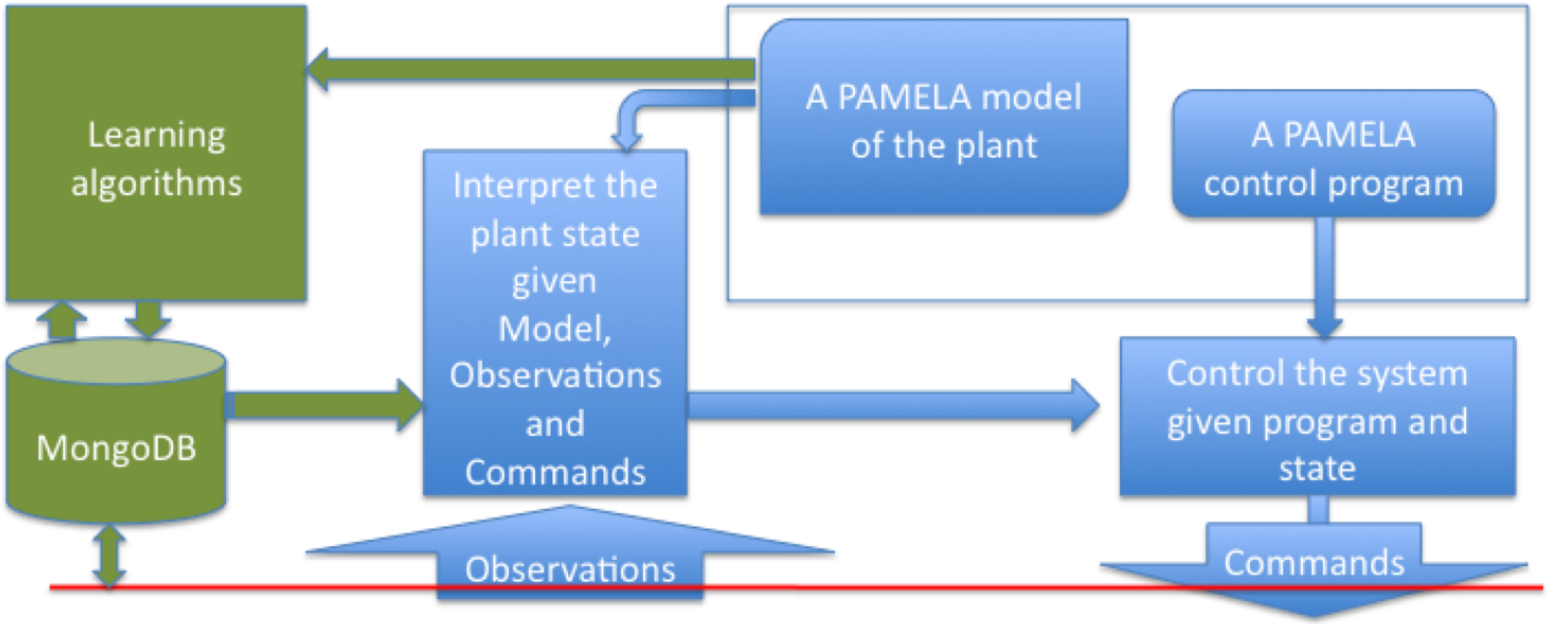
- An open source modeling language and toolset.
- Support for machine learning algorithms (probabilistic variables).
- Philosophy: A modeling language to make integration of complex algorithms intuitive.
- Cyber Physical Systems
- Cyber Cyber Systems
- Learning Algorithms
  - **Hidden Markov Models (HMM)**
  - Markov Decision Processes (MDP)
  - Partially Observable Markov Decision Processes (POMDP)
  - Bayesian Networks
  - Deep Learning
  - Linear and Nonlinear constraint solvers
  - Constraint solvers in general especially including uncertainty.

A model Based Program



The plant: A cyber physical system,
a robot,
a software system,
a controllable device,
etc, ...

Pamela Learning Architecture



The plant: A cyber physical system,
a robot,
a software system,
a controllable device,
etc, ...

Flipping a coin is a simple instantiation of a random value

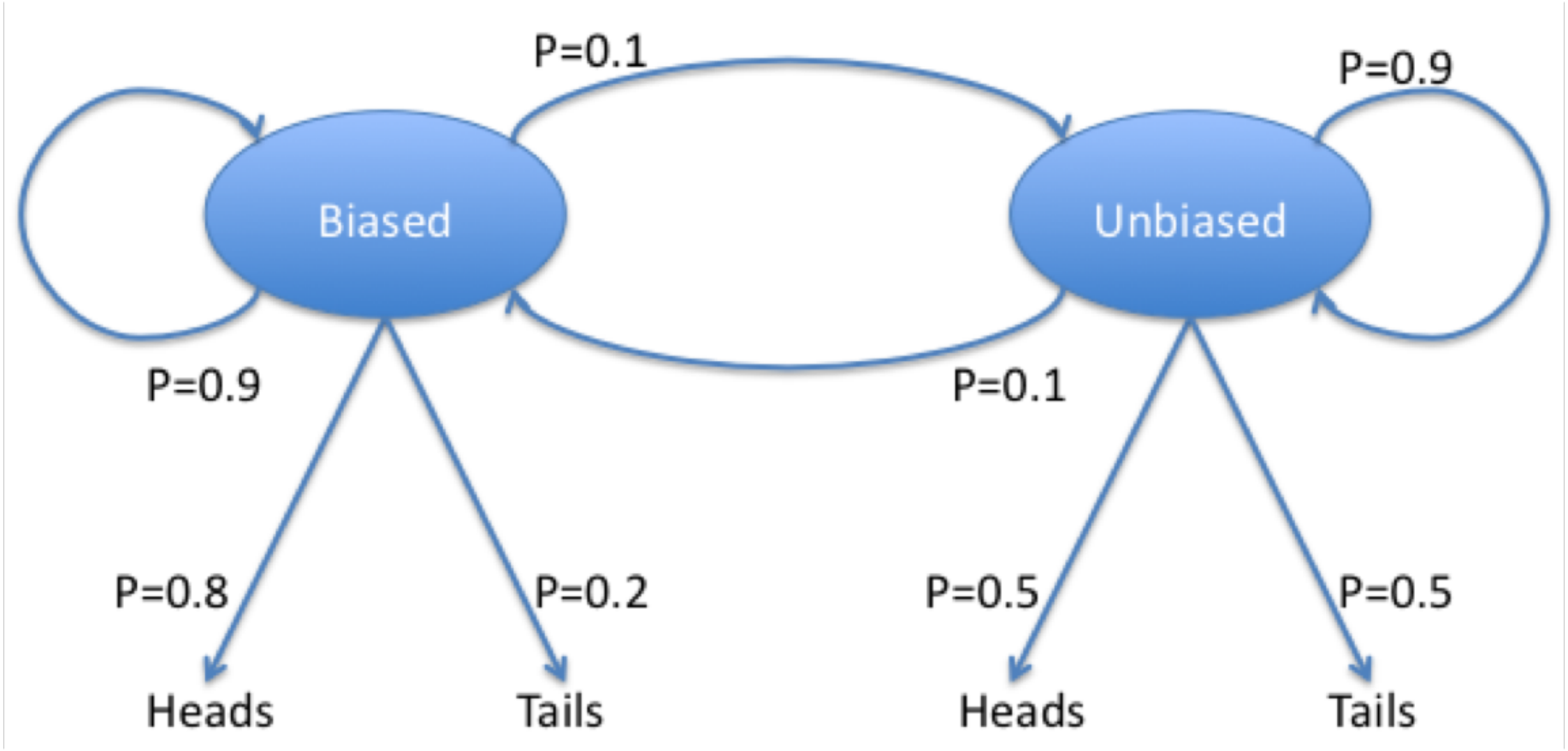
The coin may be biased towards heads or tails

A traditional example machine learning problem

- A coin tosser has two coins, one honest and one biased.
- When tossing the honest coin the tosser will periodicaly switch to using the biased coin.
- When tossing the biased coin the tosser will periodically switch back to the honest coin.
- the honest coin produces heads and tails approximately equally.
- P(heads)=P(tails) and P(heads)+P(tails)=1
- the biased coin produces heads and tails with different probabilities.
- P(heads)≠P(tails) and P(heads)+P(tails)=1

Our job is to model the problem in Pamela and to learn (automatically) the probabilities involved that can later be used to estimate the hidden state (which coin the tosser is flipping at any point in time).

Biased Coin Problem Automata Diagram



Modeling the Biased Coin in Pamela

We model this problem as a coin that is either biased or unbiased which randomly changes from biased to unbiased and which when tossed yields heads or tails based on emission probabilities.

```
(defpclass face-up []
  :meta {:doc "Observed value of the coin after flip"}
  :modes [:head :tail])

(defpclass coin []
  :modes [:biased :unbiased]

  :fields {:observed-face
           (face-up :observable true :initial :head)}

...Continued on next slide...
```

Modeling the Biased Coin in Pamela-2

```
:transitions
  {:biased:unbiased
     {:pre :biased :post :unbiased
      :probability (lvar "tBU" 0.15)}
   :unbiased:biased
     {:pre :unbiased :post :biased
      :probability (lvar "tUB" 0.2)}
   :biased:biased
     {:pre :biased :post :biased
      :probability (lvar "tBB" 0.85)}
   :unbiased:unbiased
     {:pre :unbiased :post :unbiased
      :probability (lvar "tUU" 0.8)}}
```

...Continued on next slide...

Modeling the Biased Coin in Pamela-3

```
:methods [(defpmethod flip {:primitive true} []
            (choose
              (choice :guard :biased
                (choose
                  (choice :probability (lvar "eBH" 0.82)
                    (tell :head))
                  (choice :probability (lvar "eBT" 0.18)
                    (tell :tail))))
              (choice :guard :unbiased
                (choose
                  (choice :probability (lvar "eUH" 0.52)
                    (tell :head))
                  (choice :probability (lvar "eUT" 0.48)
                    (tell :tail))))))])
```

...Continued on next slide...

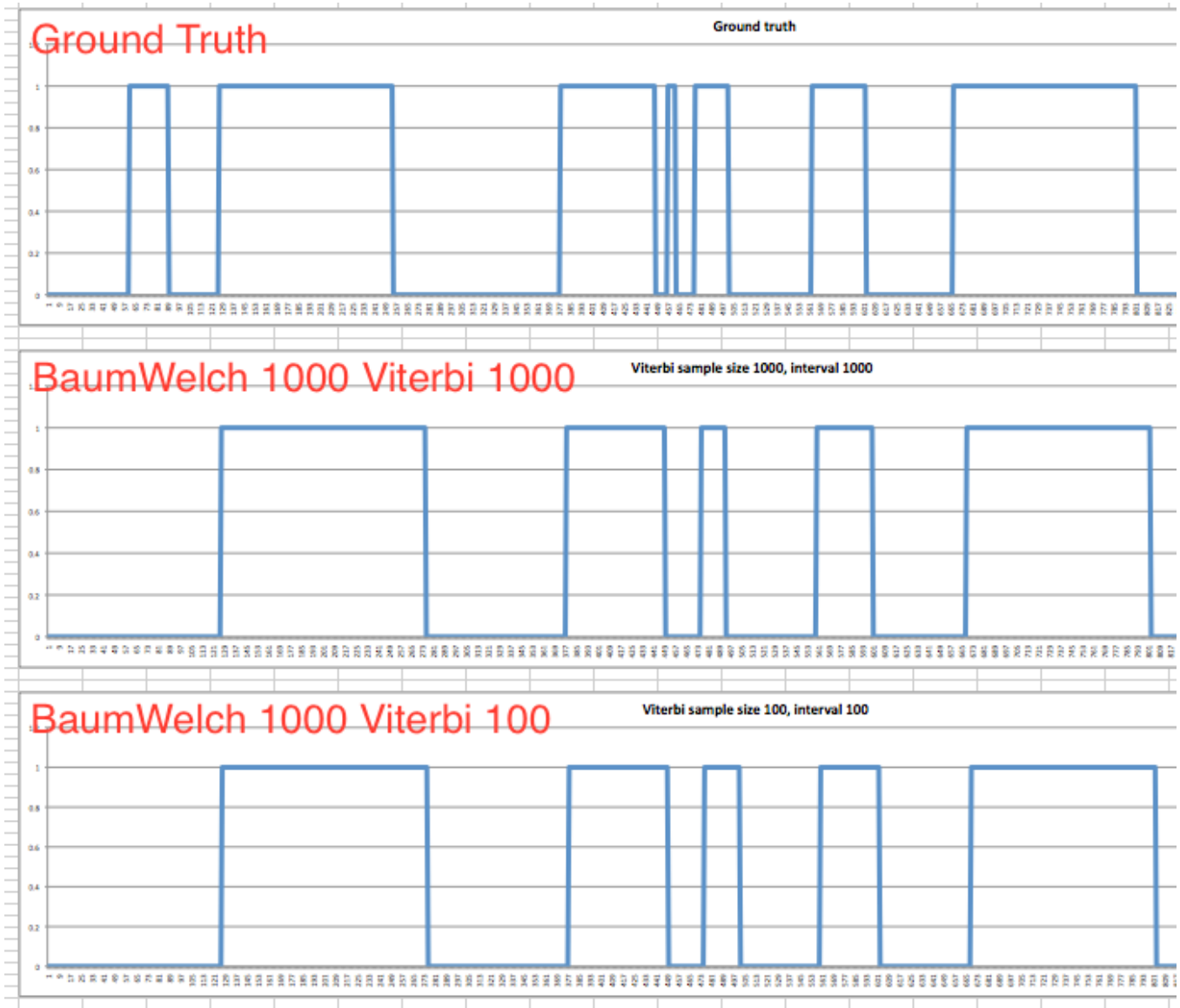
Modeling the Biased Coin in Pamela-4

The main program instantiates the coin and includes a method that can be used to generate training data. The plant implements this model by implementing the 'flip' method. The probabilities are known by the plant but our goal is to learn them.

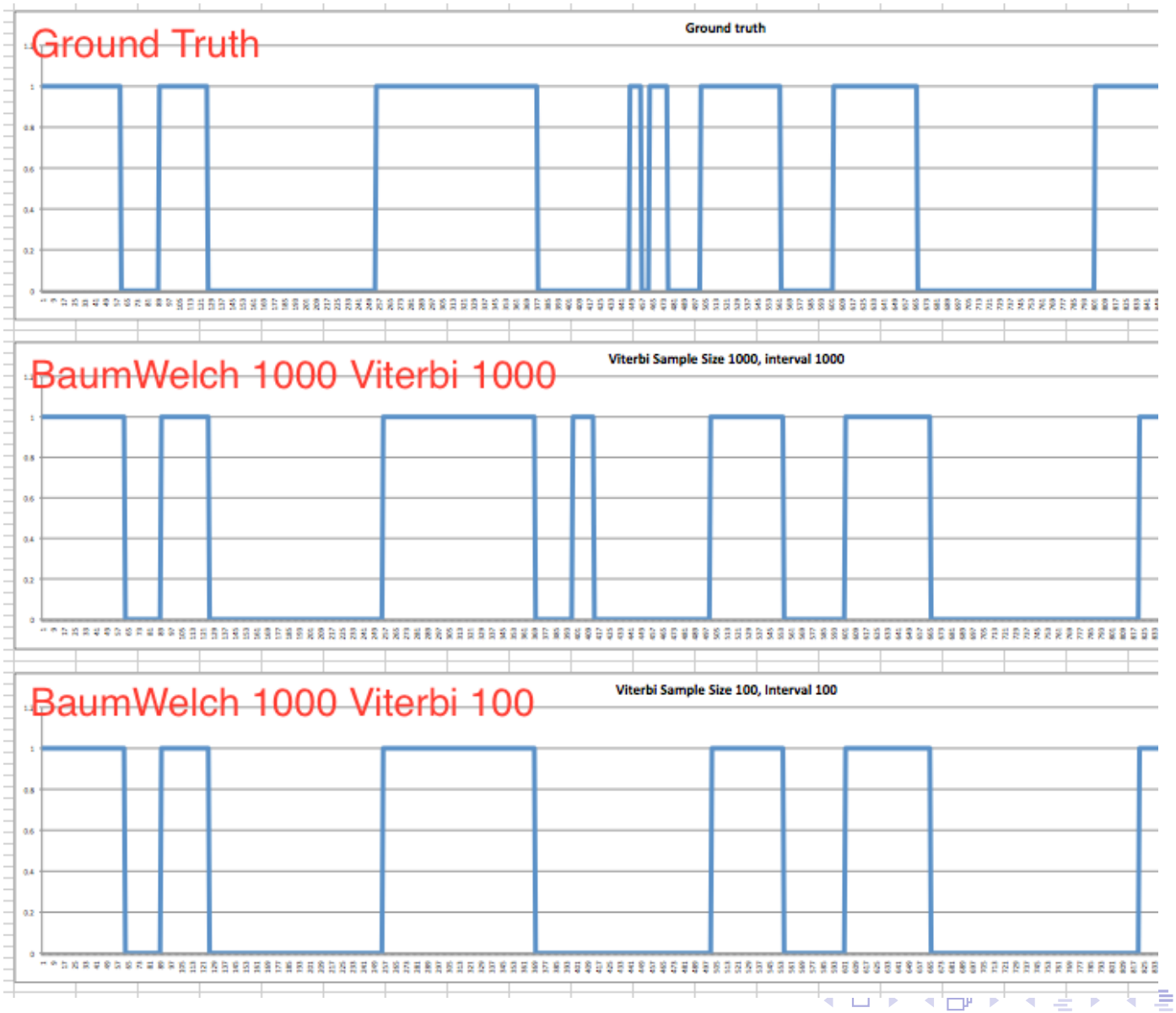
```
(defpclass main []
  :fields {:coin (coin :id "coin-plant" :plant-part "coin1")}
  :methods [(defpmethod flip-sequence
              {:doc "Series of coin flips"}[]
              (dotimes 1000
                (coin.flip)))
            ])
```

Running the flip-sequence method causes the plant to be invoked 1000 times to perform the fip operation. Observations of the flips (heads or tails) are automatically stored in the DB. The learning algorithm is automatically selected and invoked on the data.

Baum-Welch - Viterbi, eBH = 0.9 eBT = 0.1



Baum-Welch - Viterbi, eBH = 0.8 eBT = 0.2



Temporal Bounds

- In cyber-physical systems everything takes time, For example:
  - Move forwards three inches.
  - Heat the catalyst bed to ignition temperature.
- Even in software only systems, most things are not instantaneous:
  - Send a data file to a server.
  - Establish a connection with a server.
- How long things take is important!

```
(defpclass psw [gnd pwr]
  ...
  :modes {:on (= :pwr :high)
          :off (= :pwr :none)
          :fail true }
  :methods [(defpmethod turn-on
              {:pre :off :post :on :bounds [1 3]
               :doc "turns on the power supply"} []) ... ])
```

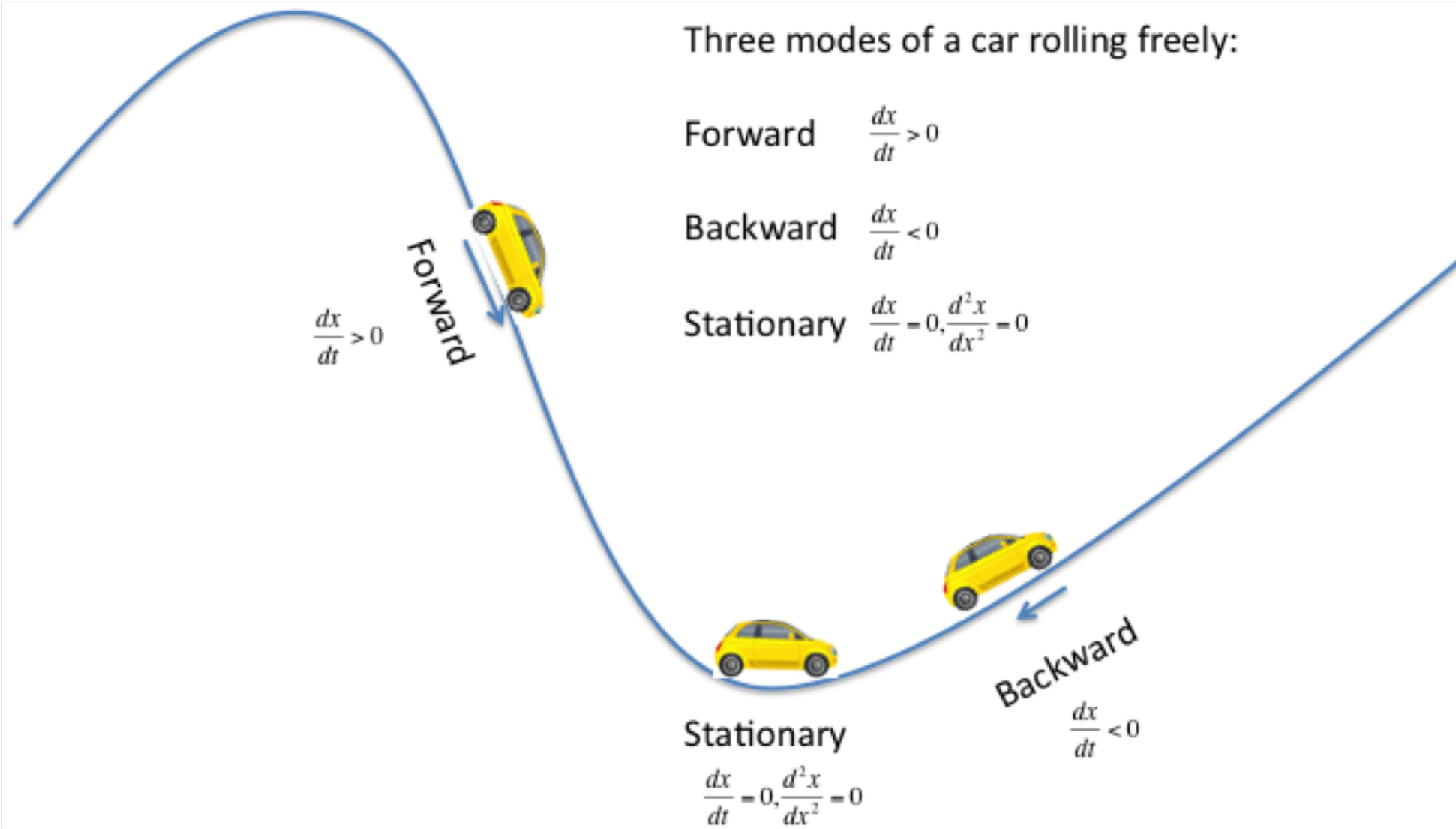
In order to reason accurately about the system behavior, we need to know the system parameters accurately as well.
The time to turn on the power was estimated by the engineer as being between 1 and 3 (milliseconds).
We can learn them, in-situ, for the real system, continuously throughout the lifetime of the system. The parameters may change with age.

```
(defpclass psw [gnd pwr]
  ...
  :modes {:on (= :pwr :high)
          :off (= :pwr :none)
          :fail true }
  :methods [(defpmethod turn-on
              {:pre :off :post :on
               :bounds [(lvar "lbTO" 1) (lvar "ubTO" 3)]
               :doc "turns on the power supply"} []) ... ])
```

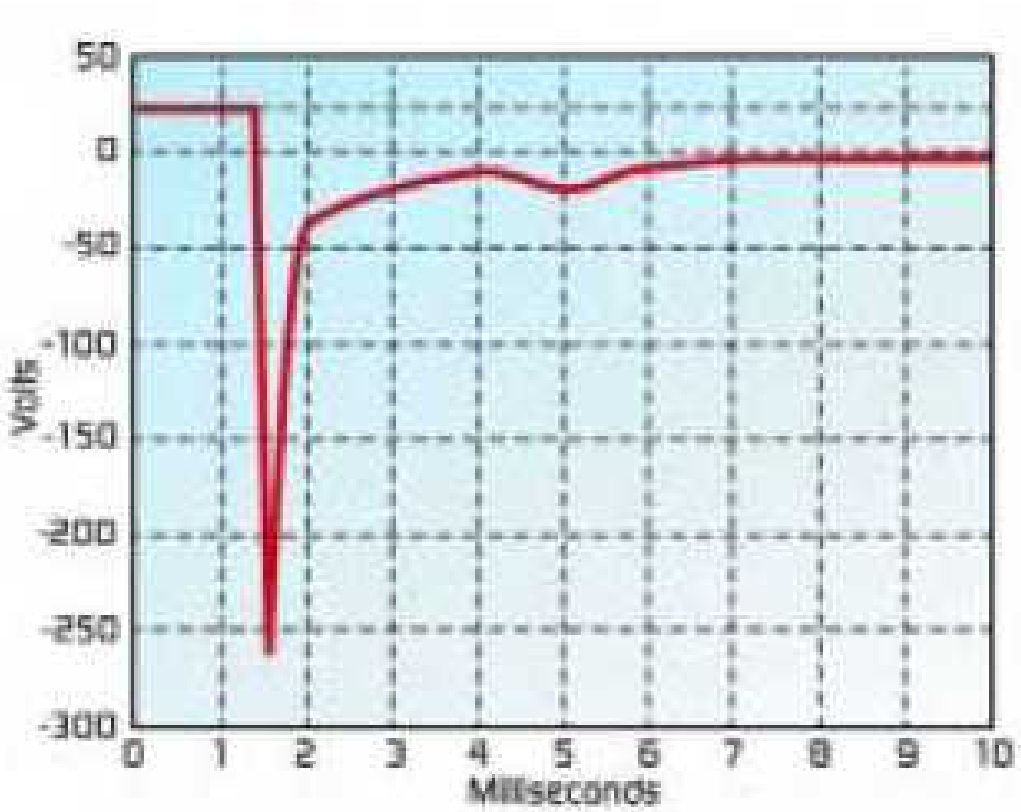
Here the two learning variables lbTO and ubTO for the lower and upper time bounds for the turn on activity have been added along with the engineers estimate of what they should be.
Learning algorithms often work better if you can give a reasonable ball-park estimate of the values.

Transitions in cyber physical systems

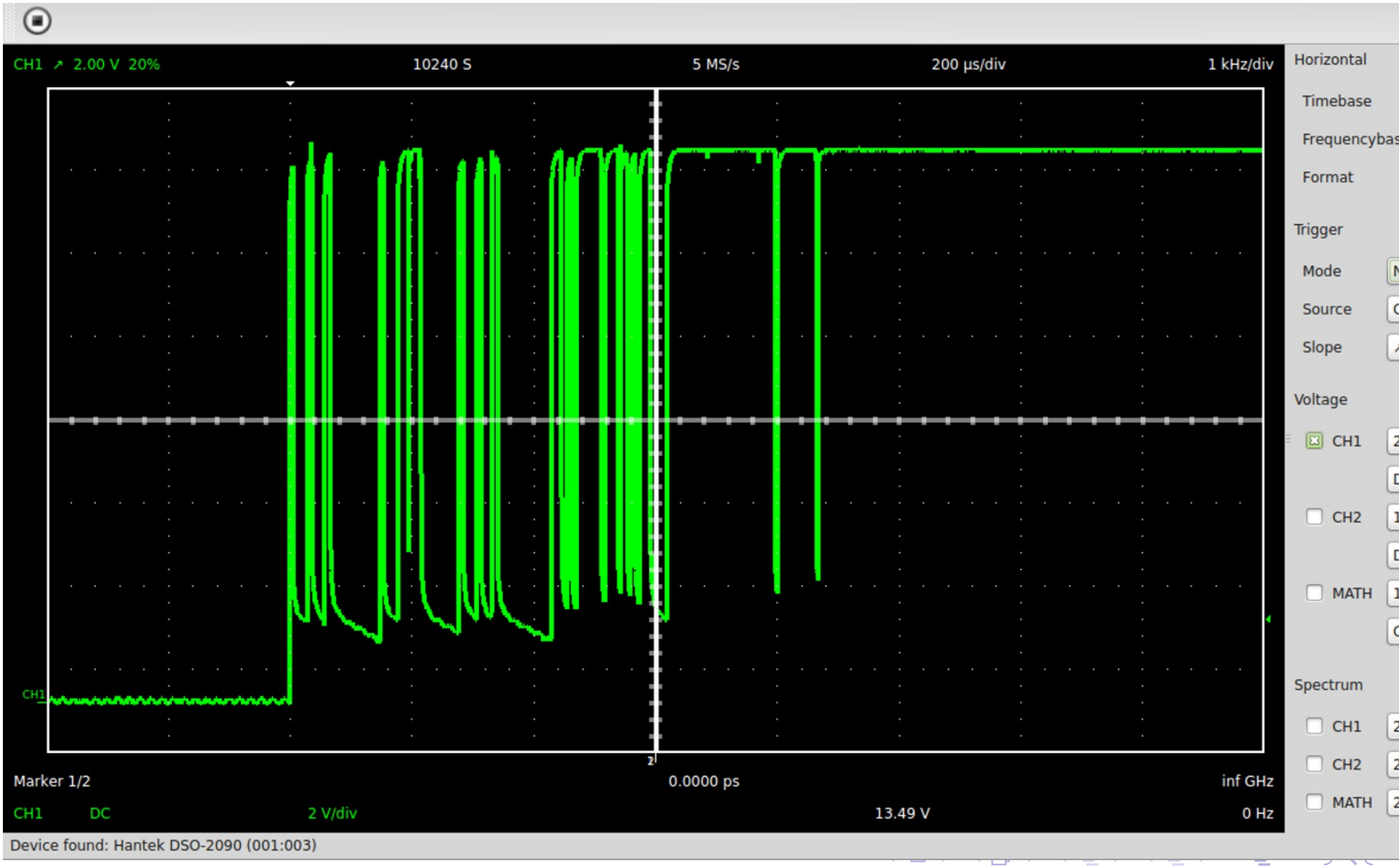
Rolling Down a Hill



Back EMF



Bouncy Switch



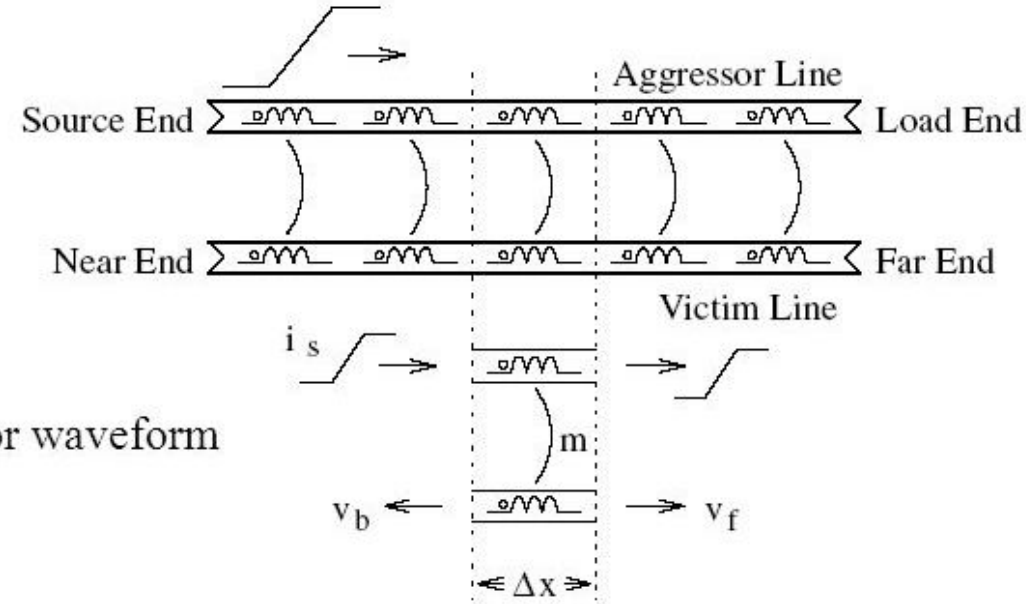
Crosstalk

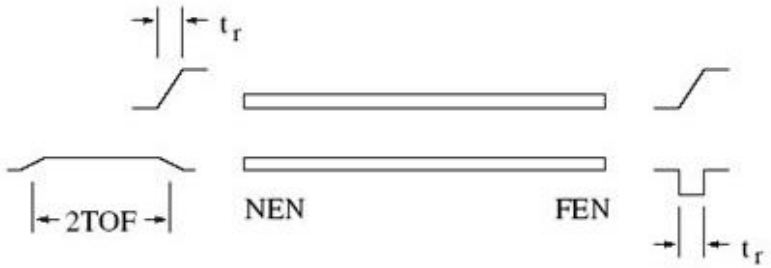
Inductive Crosstalk

$$v_{FE} = -\frac{1}{2}\frac{m}{Z_o} d \frac{dv_s}{dt}$$

$$v_{NE} = \frac{1}{4}\frac{m}{\ell} v_o.$$

assuming a triangular edge on the aggressor waveform





Part 1 Conclusions

- Transition data can be recorded automatically into a database during normal operation of the system and learning of transition and emission probabilities can take place periodically to update its self model.
- Timing information is automatically collected and can be used to automatically learn time bounds for uncontrollable activities.
- Learning transition signatures can be learned using deep learning and used for:
  - Avoiding misinterpretation during transitions
  - Early detection of transitions from signatures
  - Early warnings of declining health of system parts.

The Plant Interface

- Defines the protocol for initiating and ending an activity in the plant.
- Defines the protocol whereby the plant can asynchronously report changes in its state.
- A set of simple messages between smart components (Planner, Dispatcher) and real world (Robots, Home appliances, Quad Copters) objects

Plant messages

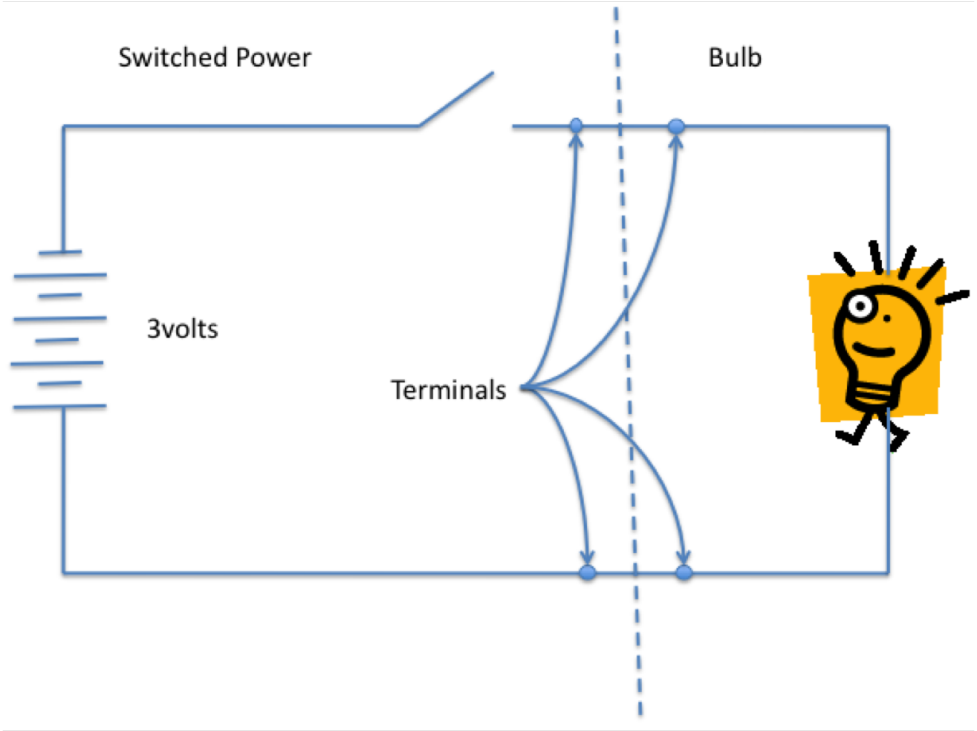
- start, started, finished, failed
- cancel, cancelled
- status-update, observations

Plant Interface in Clojure

```
(defprotocol plantI
 (start [plant-connection plant-id id function-name args ...
 (started [plant-connection plant-id id timestamp])
 (status-update [plant-connection plant-id id ...
 (finished [plant-connection plant-id id reason timestamp])
 (failed [plant-connection plant-id id reason timestamp])
 (cancel [plant-connection plant-id id timestamp])
 (cancelled [plant-connection plant-id id reason timestamp])
 (observations [plant-connection plant-id id observations time
```

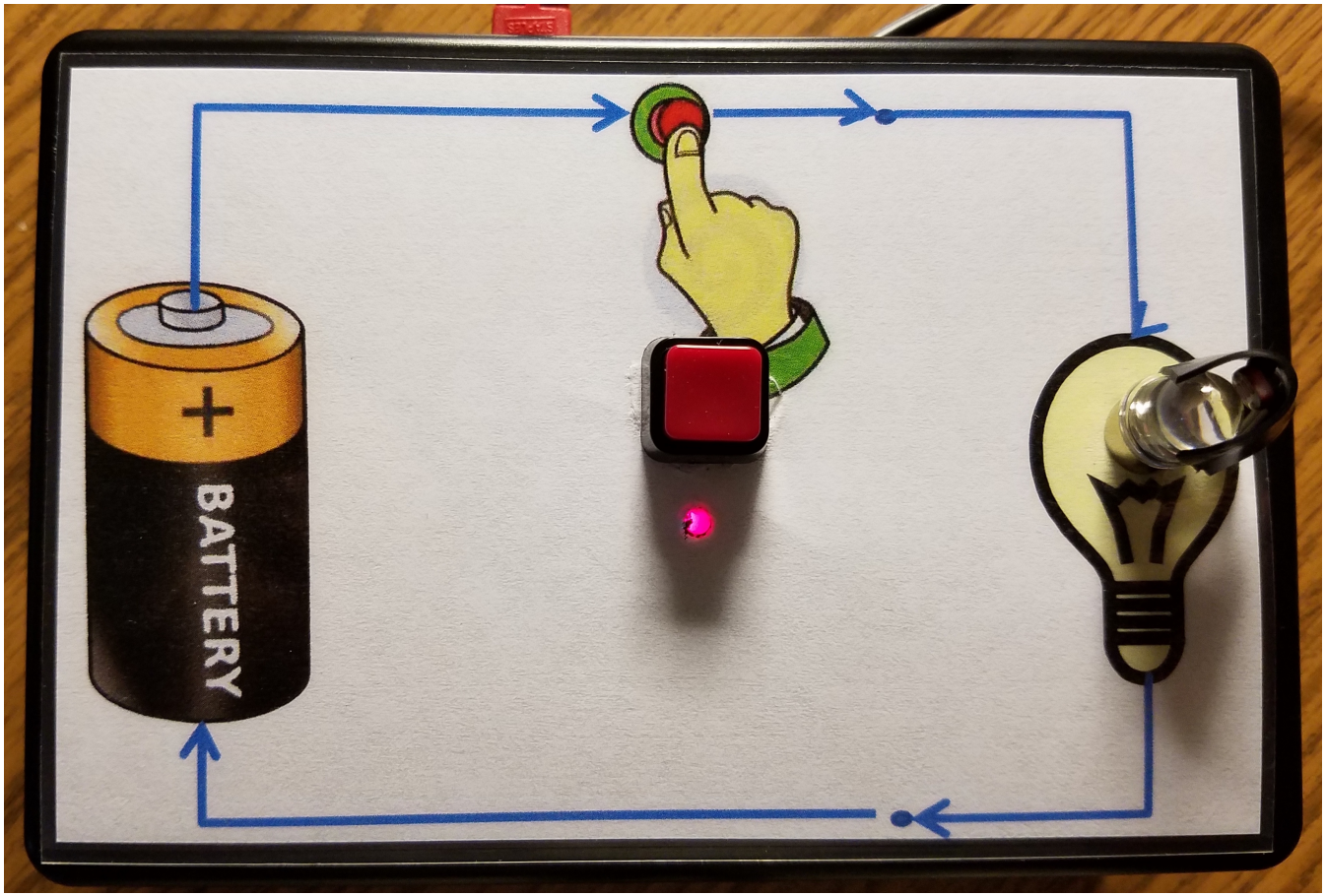
Switch and Bulb Example

At Clojure West we described a simple circuit involving two components, a controllable power switch and a lamp with a sensor:



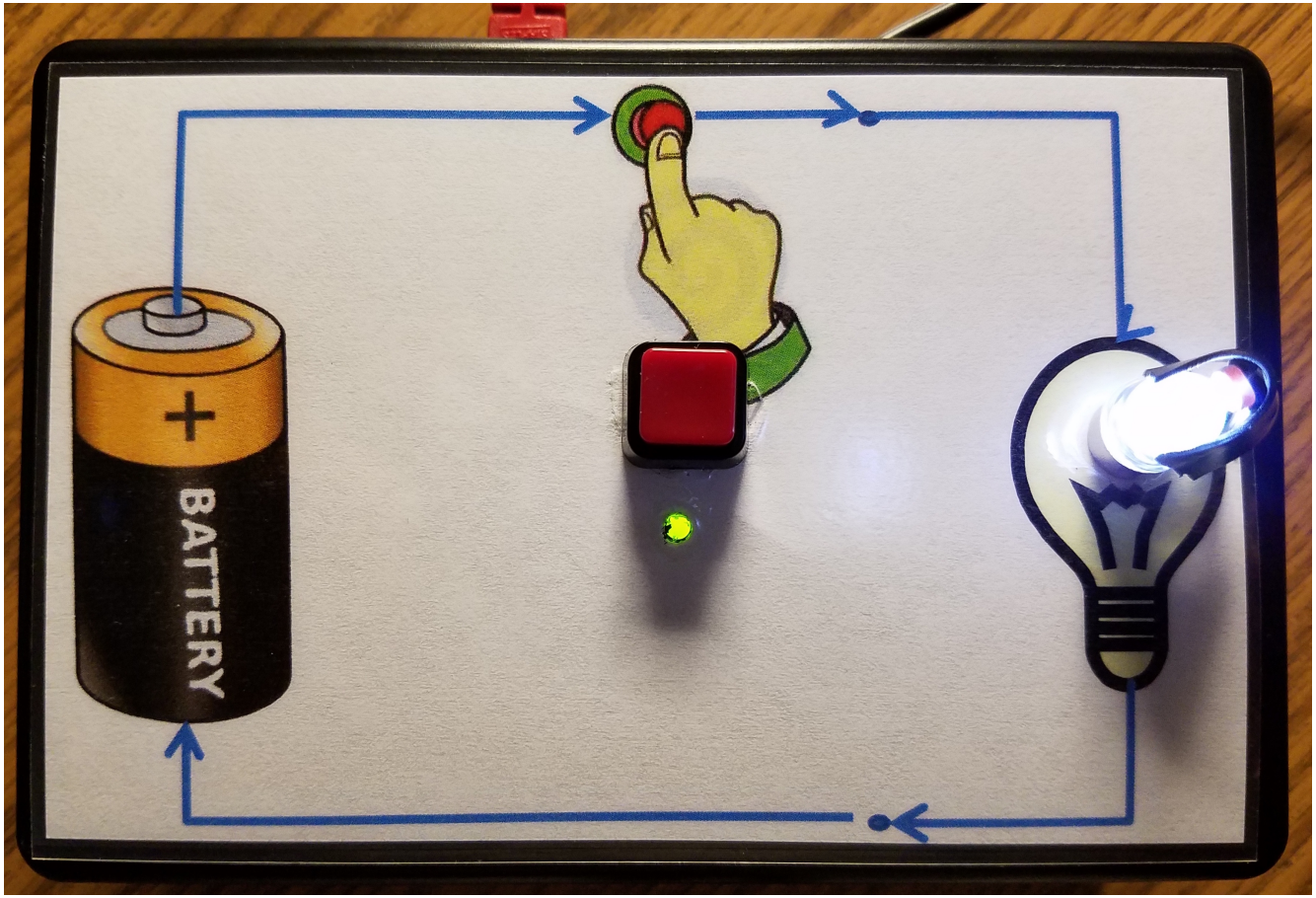
The power switch can be turned on or off by command or by physical intervention. The bulb can report whether the light is on or not.

Box Switch And Bulb Off



We built one as a testbed for Pamela using a Raspberry Pi 3.

Box Switch And Bulb On



Lets see it demonstrate the plant interface! **RUN THE DEMONSTRATION NOW**

- Release 1.0 planned for Q1 2017
- Work in the open on github/dollabs/pamela (under AL2, no CLA)
- Changes since Clojure/west
  - Expanded Pamela grammar (HTN representation, "macros")
  - Parser re-implemented with Instaparse
  - Planviz improved UI settings, can export TPN/HTN to SVG
- Features planned for release
  - Plant interface
  - Monte Carlo temporal planner
  - Belief state estimation module
  - Machine Learning
- Talk to us for details and contact us on slack #pamela

Questions?