Verification *by*, *for*, and *of* Humans: Formal Methods for Cyber-Physical Systems and Beyond

Sanjit A. Seshia

UC Berkeley

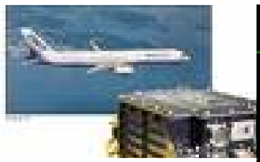
Joint work with:

Garvit Juniwal, Dorsa Sadigh, Alex Donze, Wenchao Li,
Jeff. C. Jensen, Xiaoqing Jin, Jyo Deshmukh,
Edward Lee, Shankar Sastry

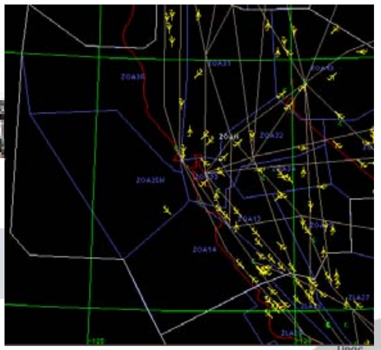
UC Berkeley, NI, Toyota

Illinois ECE Colloquium
March 19, 2015

Cyber-Physical Systems (CPS):
Tight integration of networked computation with physical systems



Avionics



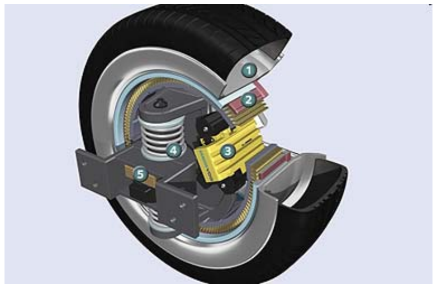
Transportation (Air traffic control at SFO)

Building Systems

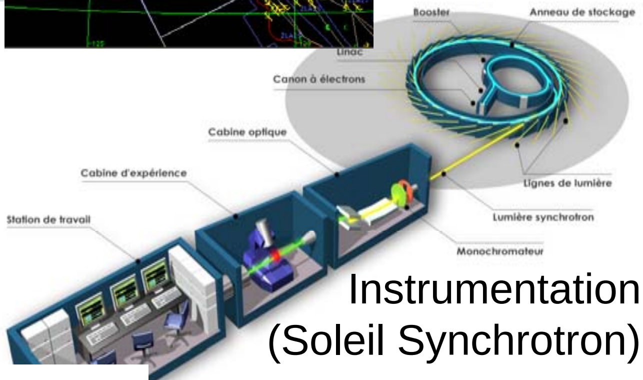
Elevator
Switch
Server room
Fire and security monitoring
Electricity

Telecommunications

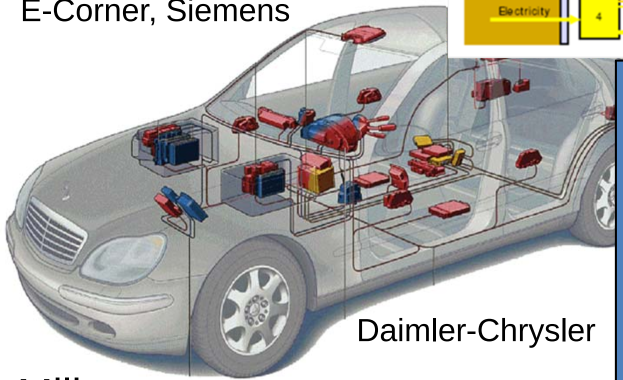
Automotive



E-Corner, Siemens



Instrumentation (Soleil Synchrotron)

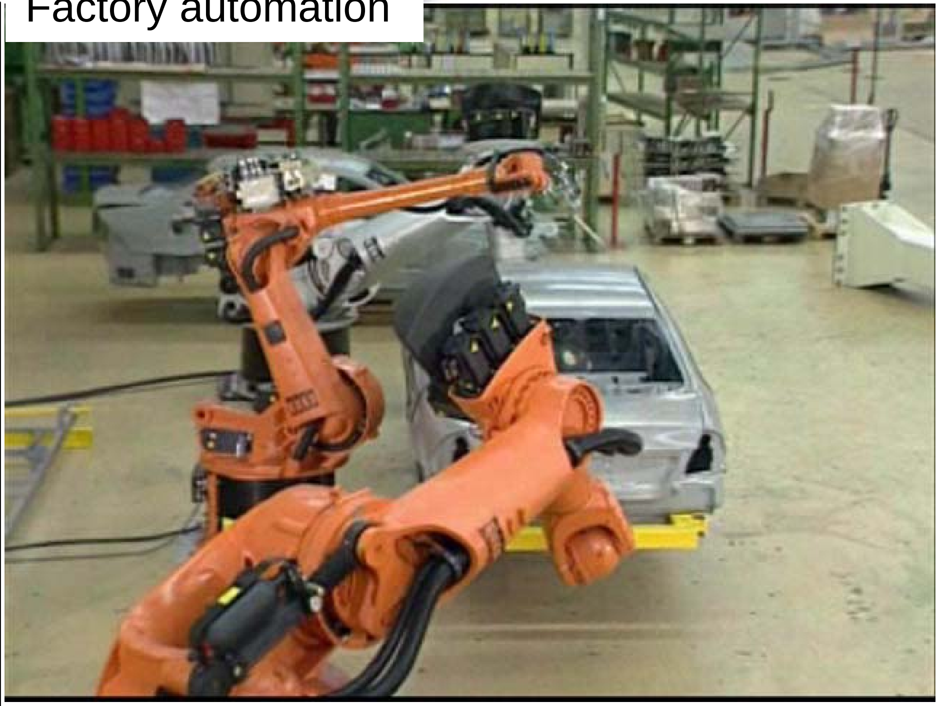


Daimler-Chrysler

Power generation and distribution

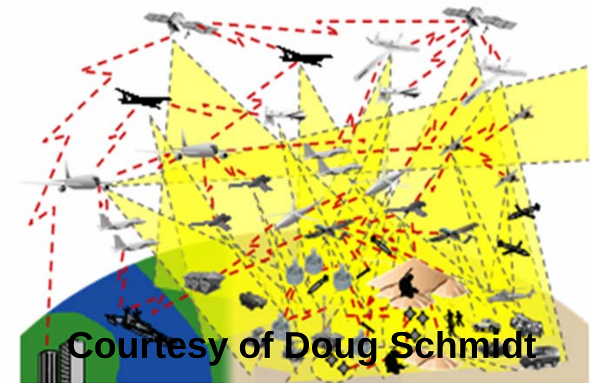
Courtesy of General Electric

Factory automation



Courtesy of Kuka Robotics Corp.

Military systems:



Courtesy of Doug Schmidt

[E. A. Lee]

Formal Methods ≈ Computational Proof Methods

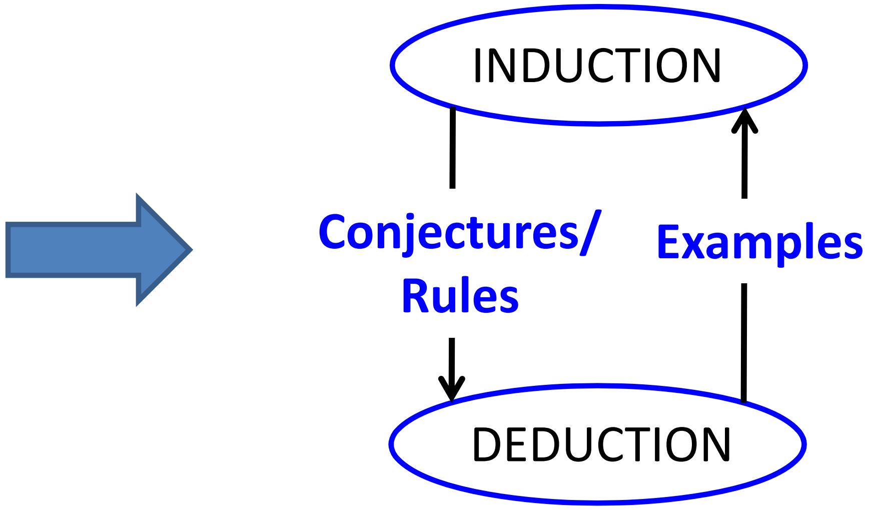
- Formal Methods ≈ Provable Guarantees
  - Specification/Modeling ≈ Statement of Conjecture/Theorem
  - Verification ≈ Proving/Disproving the Conjecture
  - Synthesis ≈ Generating (parts of) Conjecture/Proof
- Formal Methods ≈ Computational Proof methods
  - SAT / SMT solvers, model checkers, theorem provers, …

"brute force"

Formal Methods ≈ Computational Proof Methods

- Formal Methods ≈ Provable Guarantees ≈ Computational Proof methods
  - Specification, Modeling, Verification, Synthesis, …
  - SAT / SMT solvers, model checkers, theorem provers, …
- **Efficient Proof Strategy = Induction + Deduction**
  - Induction (learning from examples) + Deduction (logical inference and constraint solving)

"brute force"

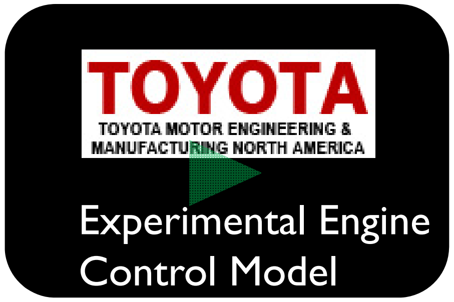


[Seshia, DAC 2012]

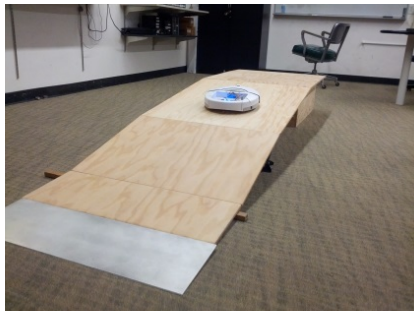
Verification by, for, of Humans

- **By** ≈ Make formal methods easier to use by engineers
- **For** ≈ Use formal methods to help users in other domains in their own work
- **Of** ≈ Use formal methods for design and analysis of human-in-the-loop systems

Three Stories: Verification by, for, of Humans


TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA
Experimental Engine Control Model

By:
Requirements Mining and Verification for Closed-Loop Control Models (Automotive focus)



For:
Virtual Lab / Automatic Grading System for Massive Open Online Course in CPS

Of:
Design and Verification of Human Cyber-Physical Systems (semi-autonomous driving)

Automobiles: A Challenging Domain for Verification of Cyber-Physical Systems

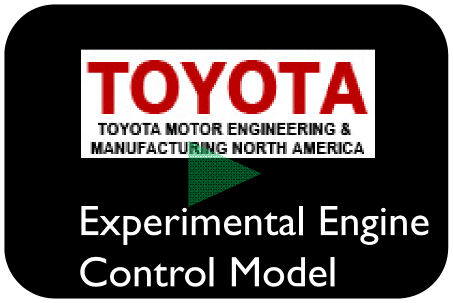
Today's automobiles "run on software" in a "networked world"

- **Nearly 100 million lines of code**
    - **cf. ~ 6.5 million lines of code for Boeing 787**
- **Running on 70 to 100 networked microprocessor-based electronic control units (ECUs)**

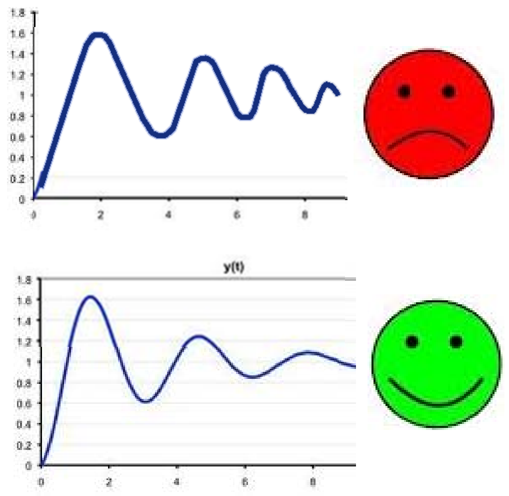
[IEEE Spectrum, Feb. 2009]

Challenges for Verification of Control Systems

- Closed-loop setting very complex
  - software + physical artifacts
  - nonlinear dynamics
  - large look-up tables
  - large amounts of switching


TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA
Experimental Engine Control Model

- Requirements Incomplete/Informal
  - Specifications often created concurrently with the design!
  - Designers often only have informal intuition about what is “good behavior”
    - “shape recognition”



Industry problem: Legacy Code → Models

Our Solution: Requirements Mining

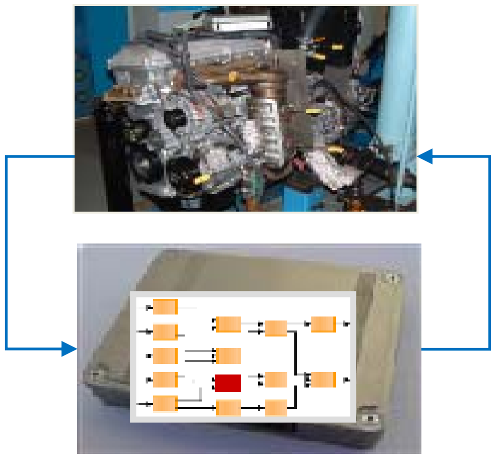
Value added by mining:

- Mined Requirements become useful documentation
- Use for code maintenance and revision
- Use during tuning and testing

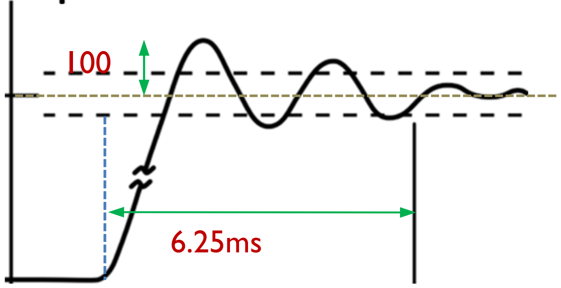
It's working, but I don't understand why!

Control Designer’s Viewpoint of Our Solution

- Tool extracts properties of closed-loop design



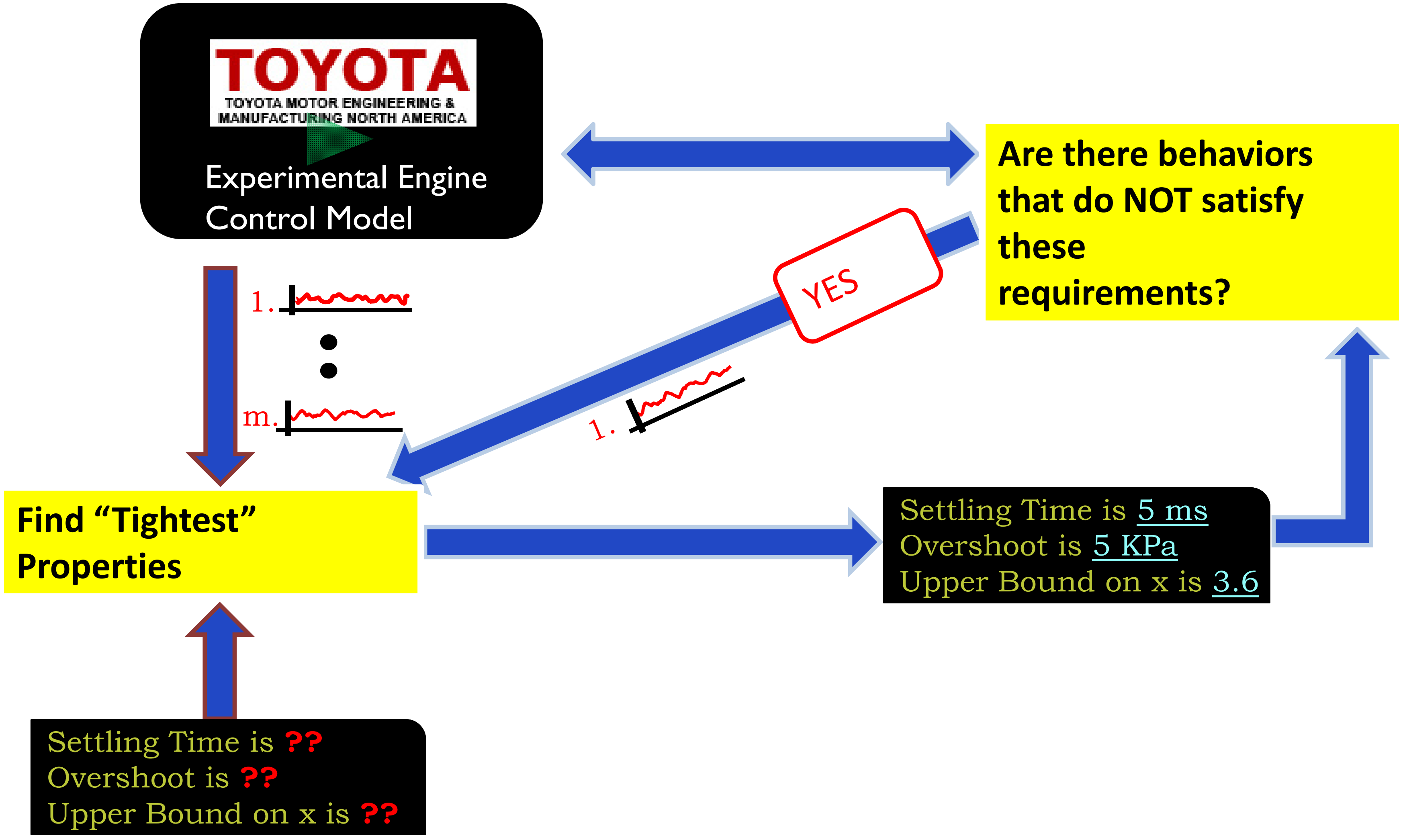
- Designer reviews mined requirements
  - “Settling time is 6.25 ms”
  - “Overshoot is 100 units”
  - Expressed in Signal Temporal Logic [Maler & Nickovic, ‘04]



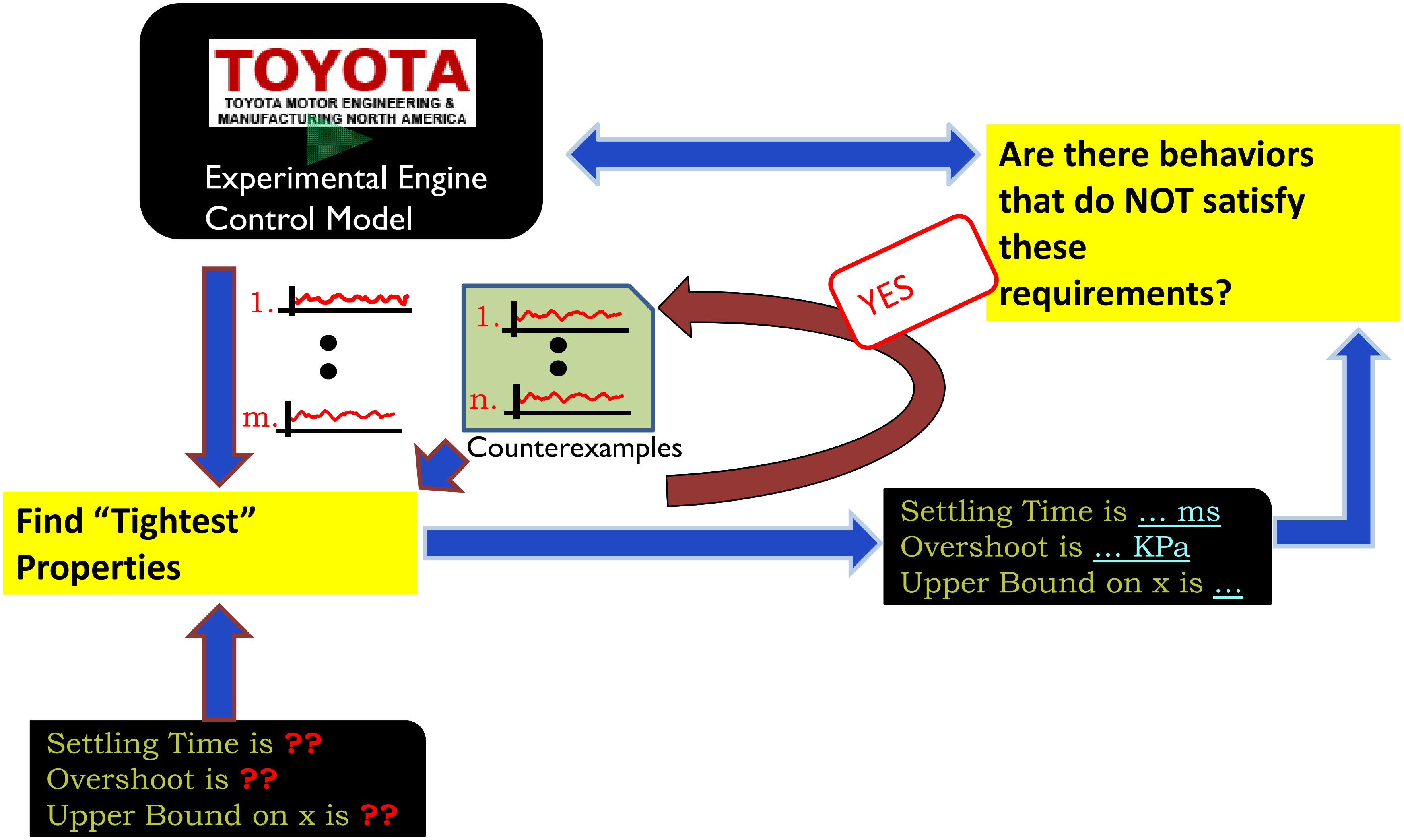
OKAY, LET'S START BY DOCUMENTING YOUR MARKET REQUIREMENTS.
NO, LET'S START BY YOU TELLING ME ALL THE THINGS YOU CAN DESIGN. THEN I'LL TELL YOU WHICH ONE I LIKE.
S. Adams E-mail: SCOTTADAMS@AOL.COM

CounterExample Guided Inductive Synthesis

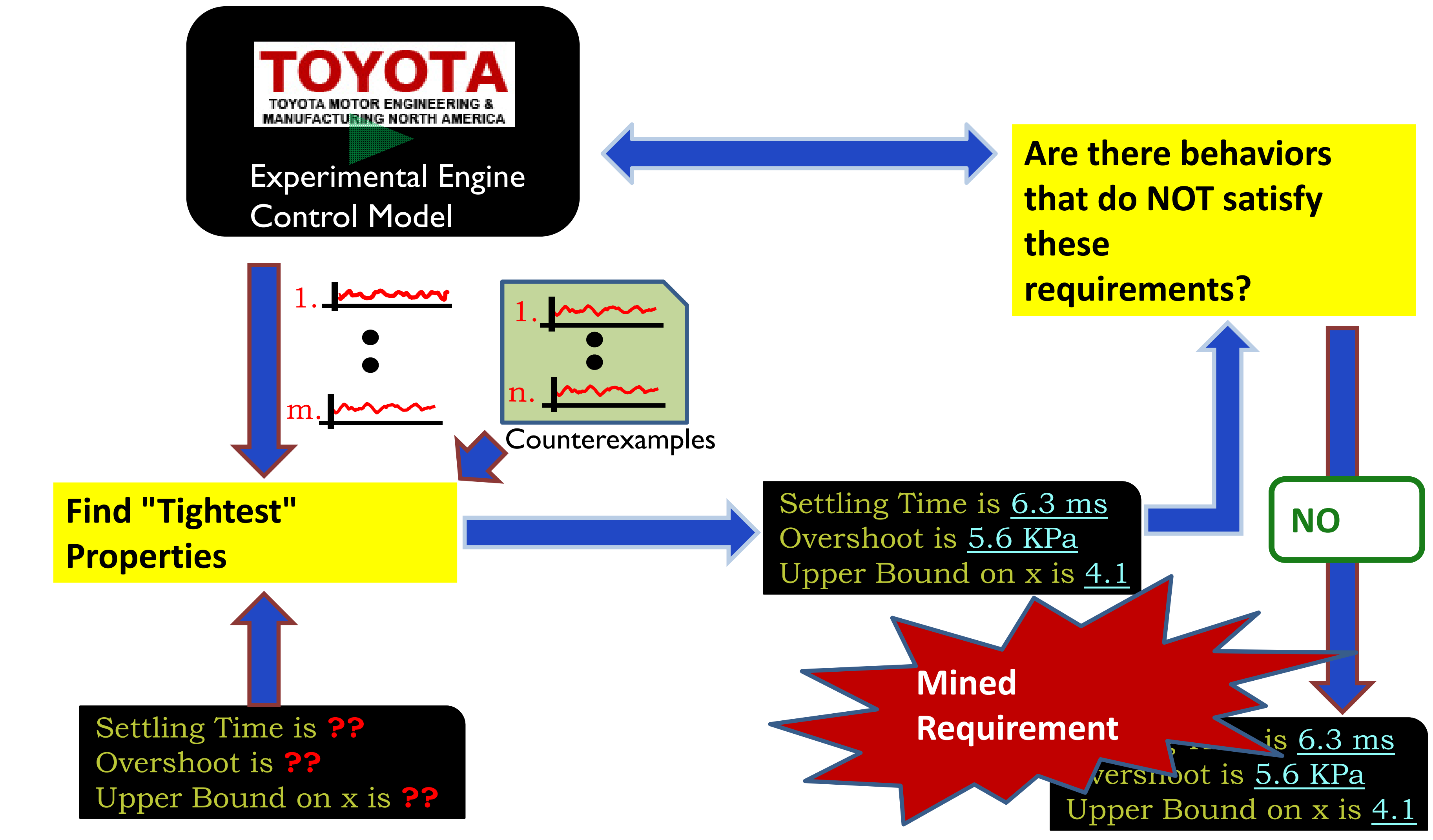
[Jin, Donze, Deshmukh, Seshia, HSCC'13; TCAD'15]


TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA
Experimental Engine Control Model
1.
m.
Are there behaviors that do NOT satisfy these requirements?
YES
1.
Find "Tightest" Properties
Settling Time is 5 ms
Overshoot is 5 KPa
Upper Bound on x is 3.6
Settling Time is ??
Overshoot is ??
Upper Bound on x is ??

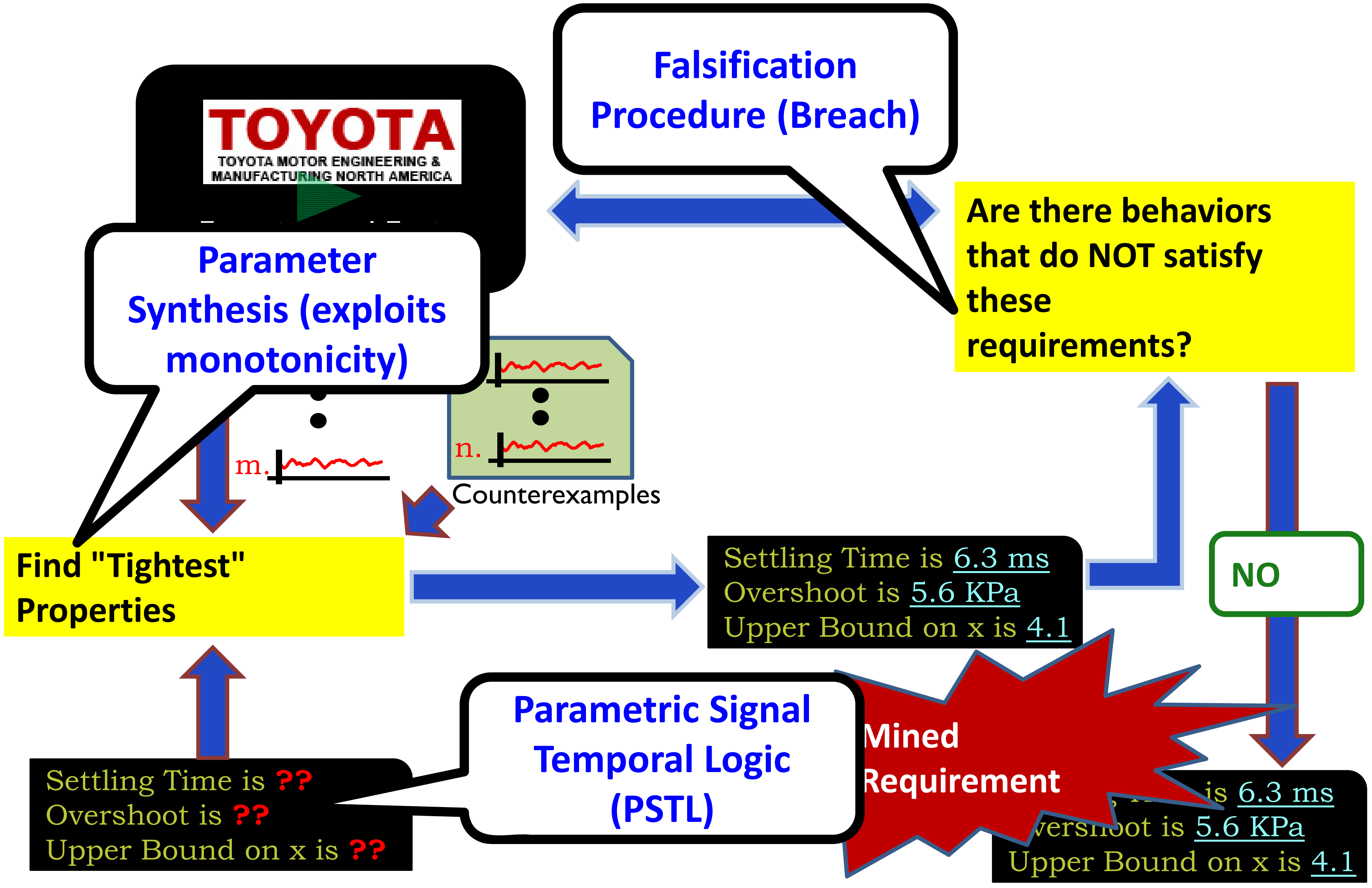
CounterExample Guided Inductive Synthesis (CEGIS)


TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA
Experimental Engine Control Model
Are there behaviors that do NOT satisfy these requirements?
YES
1.
m.
n.
Counterexamples
Find "Tightest" Properties
Settling Time is … ms
Overshoot is … KPa
Upper Bound on x is …
Settling Time is ??
Overshoot is ??
Upper Bound on x is ??

CounterExample Guided Inductive Synthesis



CounterExample Guided Inductive Synthesis

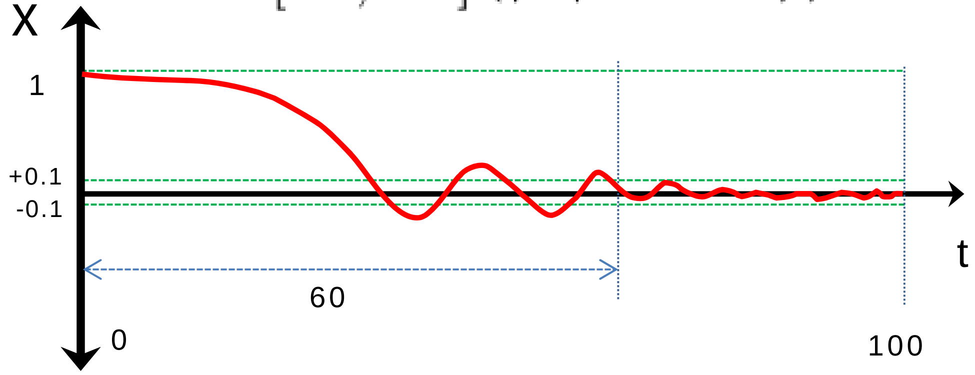


TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA
Falsification Procedure (Breach)
Are there behaviors that do NOT satisfy these requirements?
Parameter Synthesis (exploits monotonicity)
m.
n.
Counterexamples
Find "Tightest" Properties
Settling Time is 6.3 ms
Overshoot is 5.6 KPa
Upper Bound on x is 4.1
NO
Parametric Signal Temporal Logic (PSTL)
Mined Requirement
Settling Time is ??
Overshoot is ??
Upper Bound on x is ??
is 6.3 ms
vershoot is 5.6 KPa
Upper Bound on x is 4.1

Signal Temporal Logic (STL)

- Extension of Linear Temporal Logic (LTL) and Metric Temporal Logic (MTL)
  - Quantitative semantics: satisfaction of a property over a trace given real-valued interpretation
  - Greater value → more easily satisfied
  - Non-negative satisfaction value ≡ Boolean satisfaction
- Example: *"For all time points between 60 and 100, the absolute value of* x *is below 0.1"*

$$\Box_{[\mathbf{60},100]}(|x| < 0.1))$$



Parametric Signal Temporal Logic (PSTL)

- Constants in STL formula replaced with parameters
  - Scale parameters
  - Time parameters
- Examples:

$$\varphi(\tau, \pi) \doteq \Box_{[\tau,10]}(x > \pi)$$

Between some time τ and 10 seconds, x remains greater than some value π

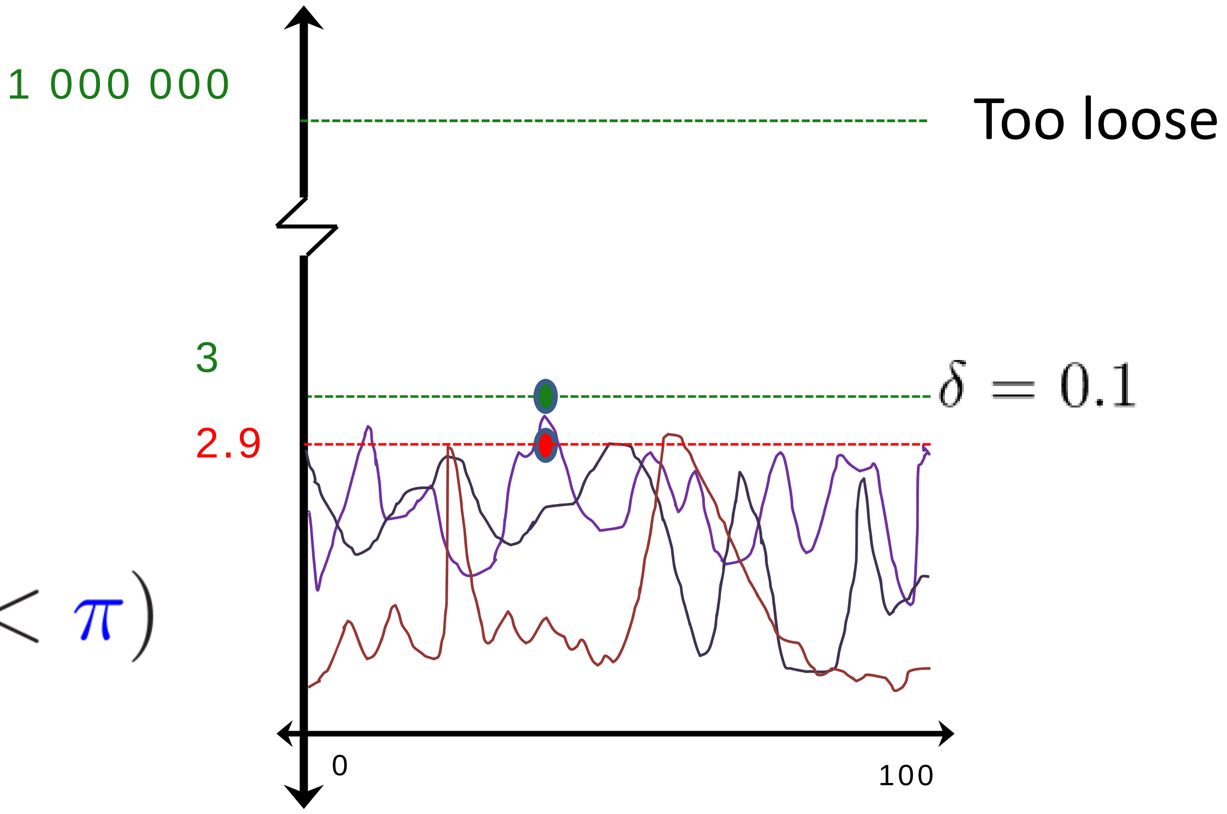
$$\varphi(\tau) \doteq \Box \left(\begin{array}{l} (gear \neq 2) \wedge \\ \Diamond_{[0,0.001]}(gear = 2) \end{array} \right) \Rightarrow \Box_{[0,\tau]}(gear = 2)$$

After transmission shifts to gear 2, it remains in gear 2 for at least τ secs

Find "Tightest" Properties

Parameter Synthesis = Find δ-tight values

$\Box(x < \pi)$



Parameter Synthesis

Find "Tightest" Properties

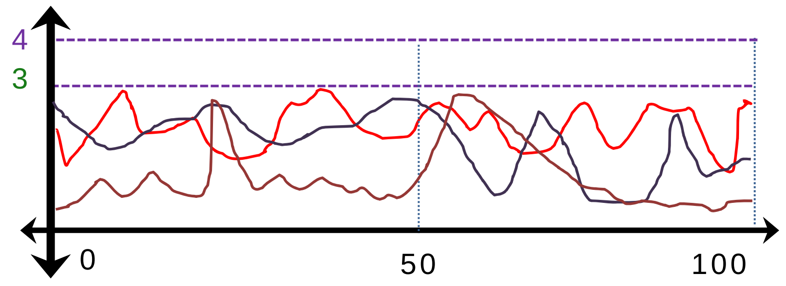
- Non-linear optimization problem
- Naïve approach:
  - grid parameter space
  - evaluate satisfaction value at each point
  - pick valuation with smallest satisfaction value
- Problems:
  - Exponential number of grid points (in #parameters)
  - Could miss optimal values due to wrong gridding

Satisfaction Monotonicity

Find "Tightest" Properties

- Satisfaction value monotonic in parameter value

$\square(x < \pi)$

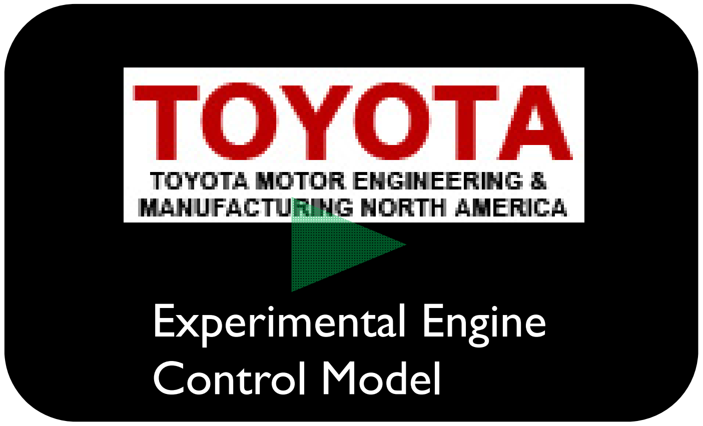


If upper bound of all signals is 3, any number > 3 is also an upper bound

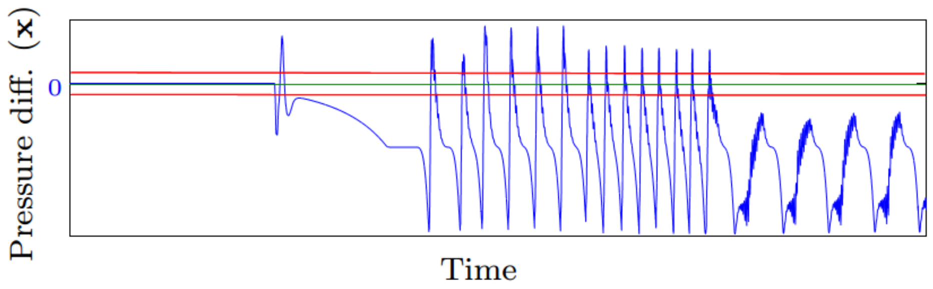
- Theorem: Deciding monotonicity of a PSTL formula is undecidable
- Use an encoding to SMT (undecidable logic)
- If monotonic, use binary search, otherwise exhaustive search

Experimental Results on Industrial Airpath Controller

[Jin, Donze, Deshmukh, Seshia, HSCC 2013]


TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA
Experimental Engine Control Model

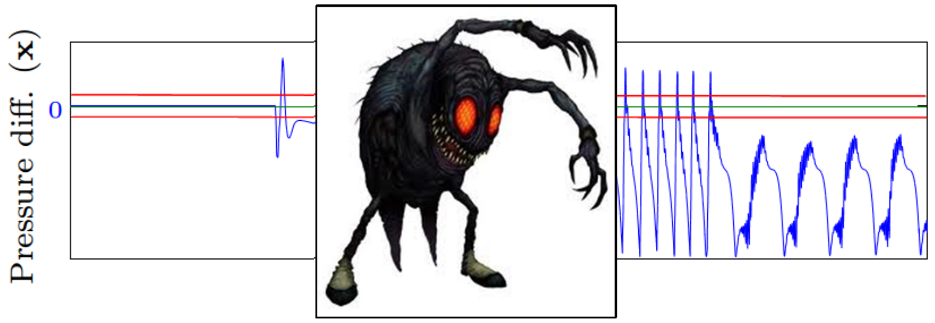
- Found max overshoot with 7000+ simulations in 13 hours
- Attempt to mine maximum observed settling time:
  - stops after 4 iterations
  - gives answer t_{settle} = simulation time horizon (shown in trace below)



Mining can expose deep bugs

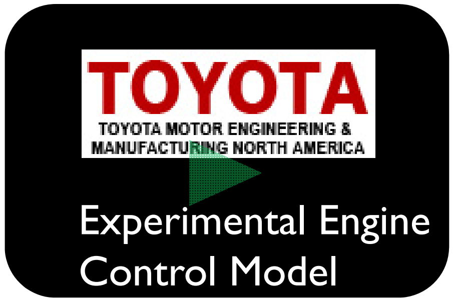
TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA

Experimental Engine Control Model

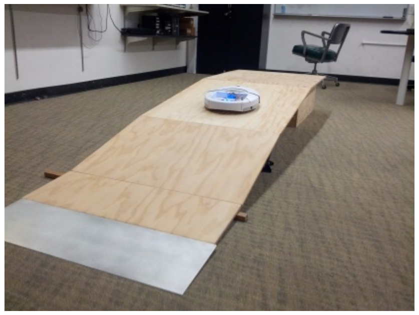


- Uncovered a tricky bug
  - Discussion with control designer revealed it to be a real bug
  - Root cause identified as wrong value in a look-up table, bug was fixed
- Why mining could be useful for bug-finding:
  - Can uncover subtle relations that should not hold
  - Looking for bugs ≈ Mine for negation of bug

Three Stories: Verification by, for, of Humans


TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA
Experimental Engine Control Model

By:
Requirements Mining and Verification for Closed-Loop Control Models (Automotive focus)



For:
Virtual Lab / Automatic Grading System for Massive Open Online Course in CPS

Of:
Design and Verification of Human Cyber-Physical Systems (semi-autonomous driving)

Massive Open Online Courses (MOOCs)

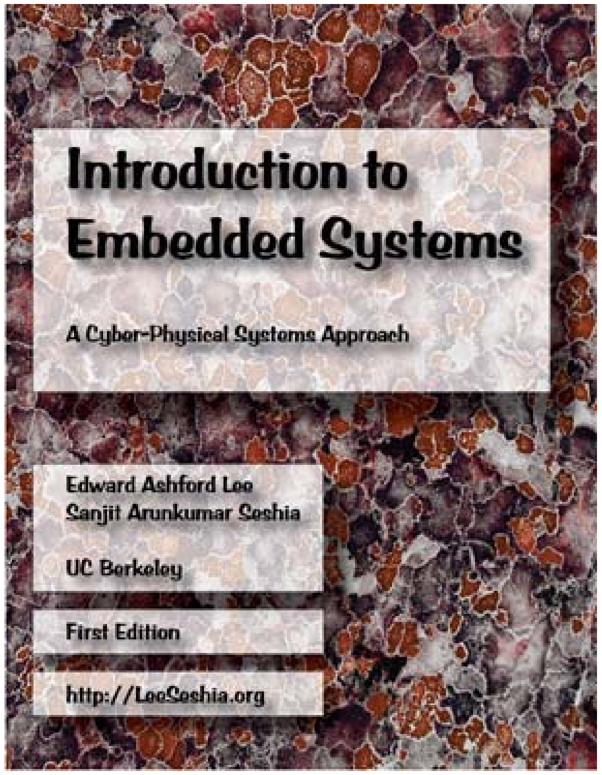
edX coursera UDACITY

Courses from universities world-wide available
to any one with an Internet connection

EECS 149: Introduction to Embedded Systems

UC Berkeley

This course introduces the *modeling, design and analysis* of *computational systems that interact with physical processes*.



Introduction to Embedded Systems

A Cyber-Physical Systems Approach

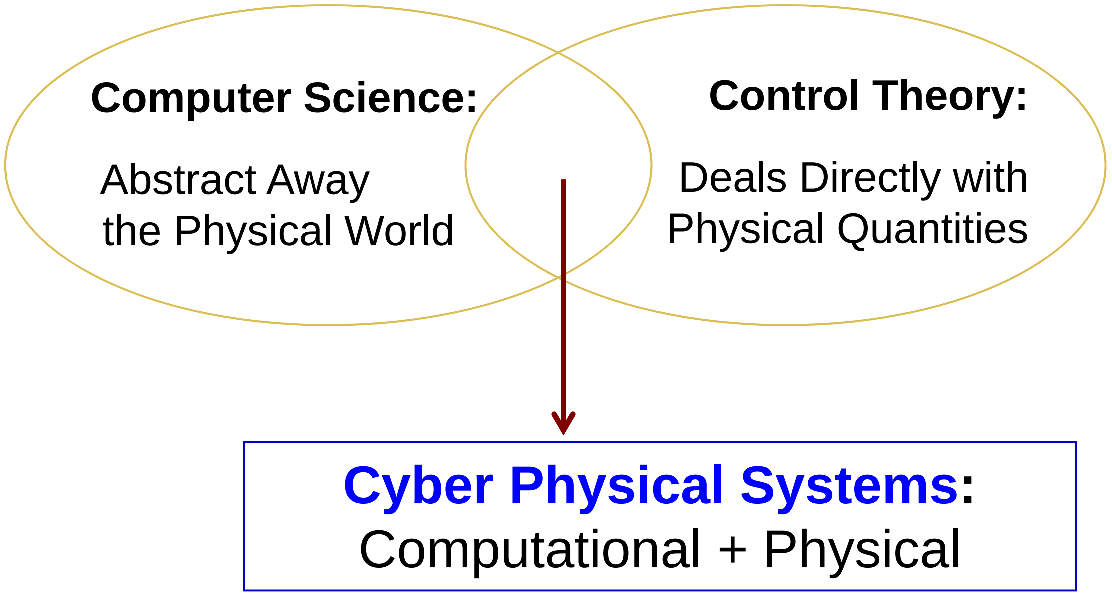
Edward Ashford Lee
Sanjit Arunkumar Seshia

UC Berkeley

First Edition

http://LeeSeshia.org

http://leeseshia.org/

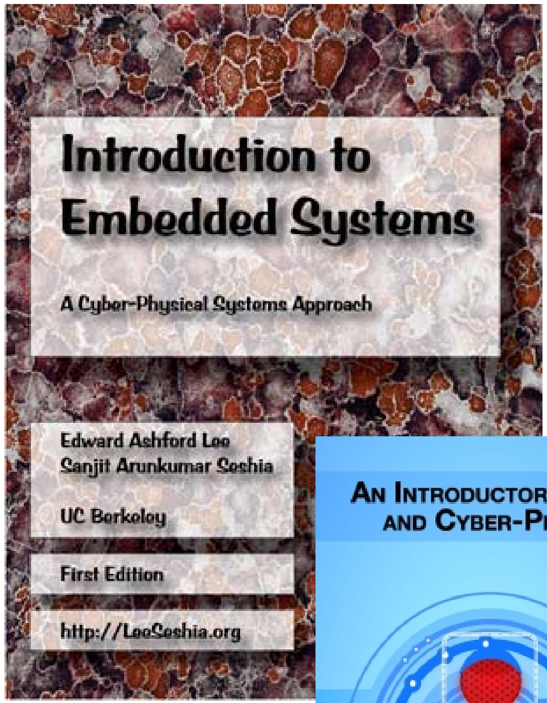

Computer Science:
Abstract Away the Physical World
Control Theory:
Deals Directly with Physical Quantities
Cyber Physical Systems:
Computational + Physical

On-campus course gets somewhat diverse enrollment (EE/CS, ME, CE, …)

The Core Learning Experience: Exercises and Labs

- Textbook Exercises:
  - High-level modeling with FSMs, ODEs, temporal logic, etc.
  - Programming in various languages (C, LabVIEW, etc.)
  - Algorithm design and analysis (scheduling, verification, etc.)
- Laboratory (6 weeks)
- Capstone design project (12 weeks)

➢ **How to extend this experience to a MOOC version of EECS 149?**


Introduction to Embedded Systems

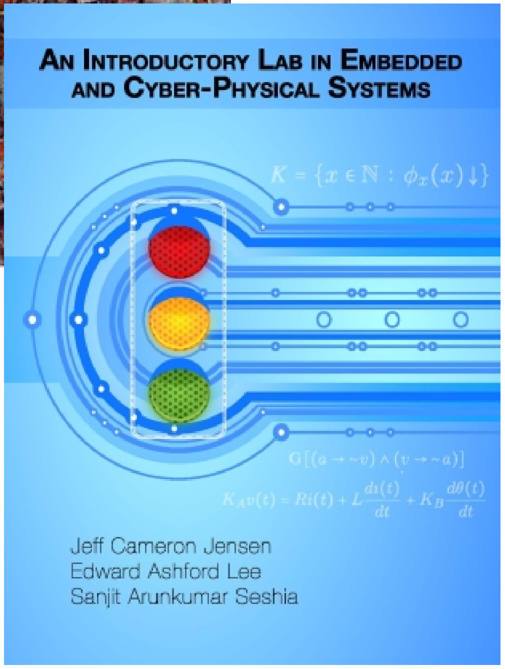
A Cyber-Physical Systems Approach

Edward Ashford Lee
Sanjit Arunkumar Seshia

UC Berkeley

First Edition

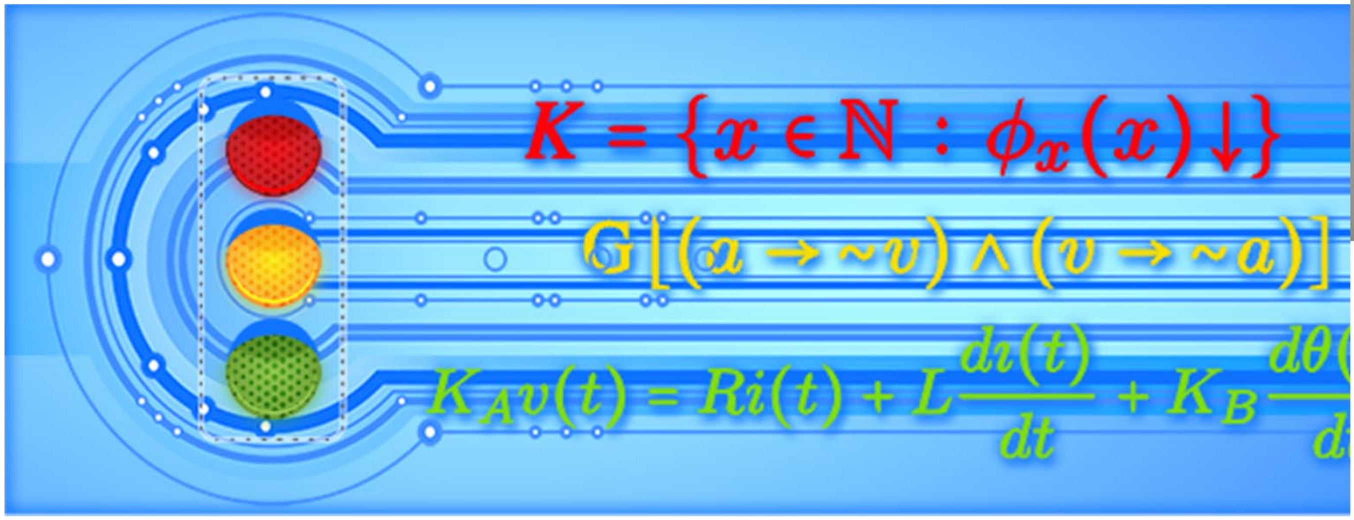
http://LeeSeshia.org


AN INTRODUCTORY LAB IN EMBEDDED AND CYBER-PHYSICAL SYSTEMS

Jeff Cameron Jensen
Edward Ashford Lee
Sanjit Arunkumar Seshia

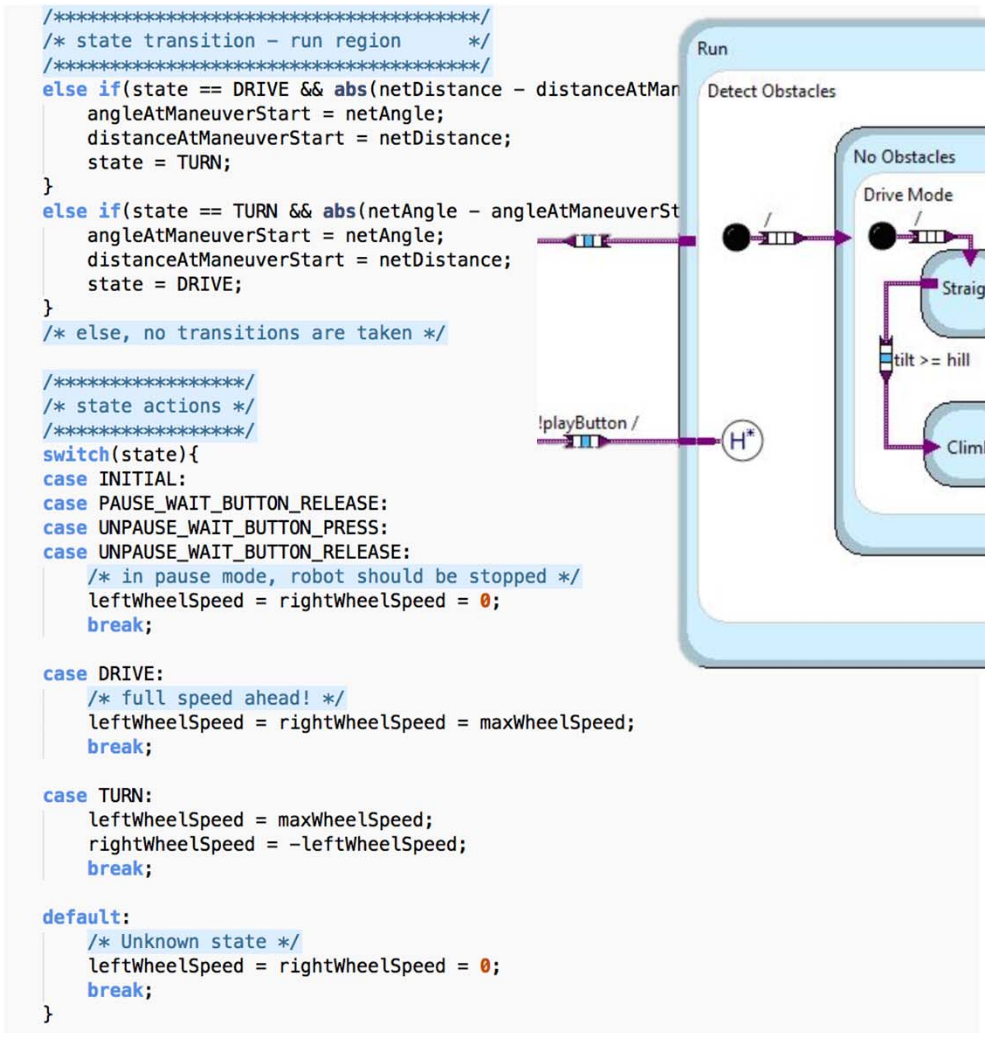
EECS149.1x: Cyber-Physical Systems

- MOOC offering on edX: May 6 to June 24, 2014
- Berkeley-NI collaboration
- Virtual lab technology for CPS: NI LabVIEW Robotics Simulator
- First course to employ *formal verification* in auto-grader

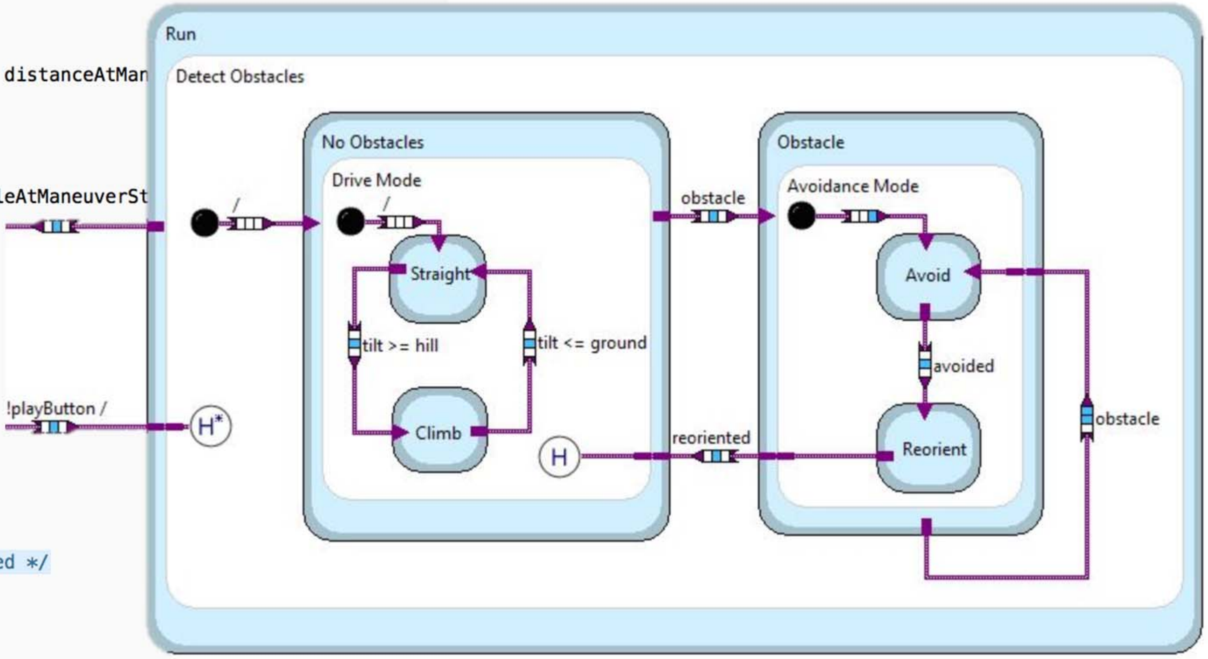

$K = \{x \in \mathbb{N} : \phi_x(x)\downarrow\}$
$G[(a \rightarrow \sim v) \wedge (v \rightarrow \sim a)]$
$K_A v(t) = Ri(t) + L\frac{di(t)}{dt} + K_B\frac{d\theta}{d}$

On-Campus Lab Assignment: The “Hill-Climbing” Robot

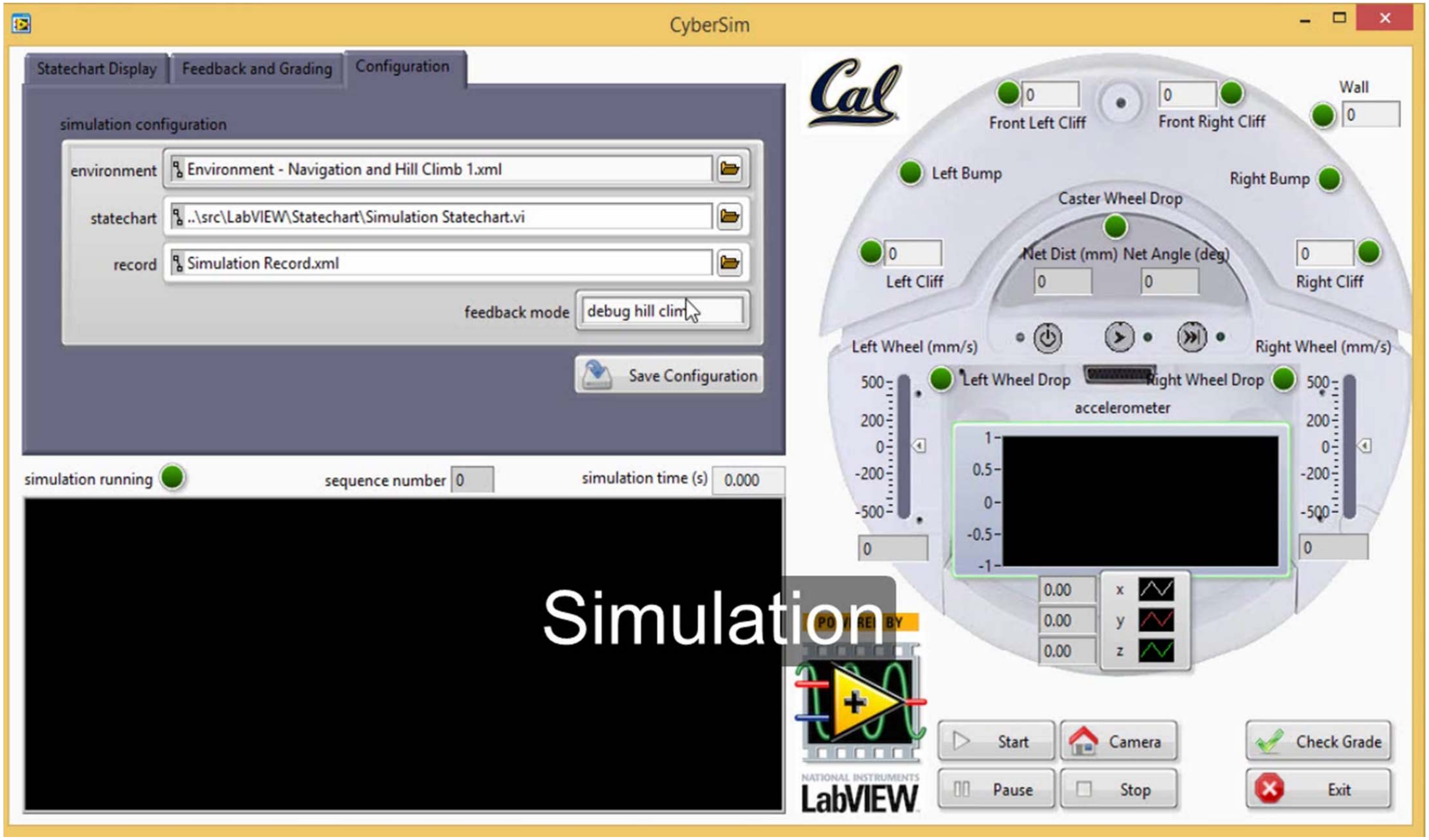
Controller: Programming in C, then LabVIEW



```
/****************************************/
/* state transition - run region      */
/****************************************/
else if(state == DRIVE && abs(netDistance - distanceAtMan
    angleAtManeuverStart = netAngle;
    distanceAtManeuverStart = netDistance;
    state = TURN;
}
else if(state == TURN && abs(netAngle - angleAtManeuverSt
    angleAtManeuverStart = netAngle;
    distanceAtManeuverStart = netDistance;
    state = DRIVE;
}
/* else, no transitions are taken */

/*****************/
/* state actions */
/*****************/
switch(state){
case INITIAL:
case PAUSE_WAIT_BUTTON_RELEASE:
case UNPAUSE_WAIT_BUTTON_PRESS:
case UNPAUSE_WAIT_BUTTON_RELEASE:
    /* in pause mode, robot should be stopped */
    leftWheelSpeed = rightWheelSpeed = 0;
    break;

case DRIVE:
    /* full speed ahead! */
    leftWheelSpeed = rightWheelSpeed = maxWheelSpeed;
    break;

case TURN:
    leftWheelSpeed = maxWheelSpeed;
    rightWheelSpeed = -leftWheelSpeed;
    break;

default:
    /* Unknown state */
    leftWheelSpeed = rightWheelSpeed = 0;
    break;
}
```



Virtual Lab Demo: NI Robotics Simulator + UC Berkeley CPSGrader Software



simulation configuration

environment: Environment - Navigation and Hill Climb 1.xml

statechart: ..\src\LabVIEW\Statechart\Simulation Statechart.vi

record: Simulation Record.xml

feedback mode: debug hill clim

simulation running

sequence number: 0

simulation time (s): 0.000

CPSGrader: Auto-Grading and Feedback Generation

[Juniwal, Donze, Jensen, Seshia, EMSOFT 2014]

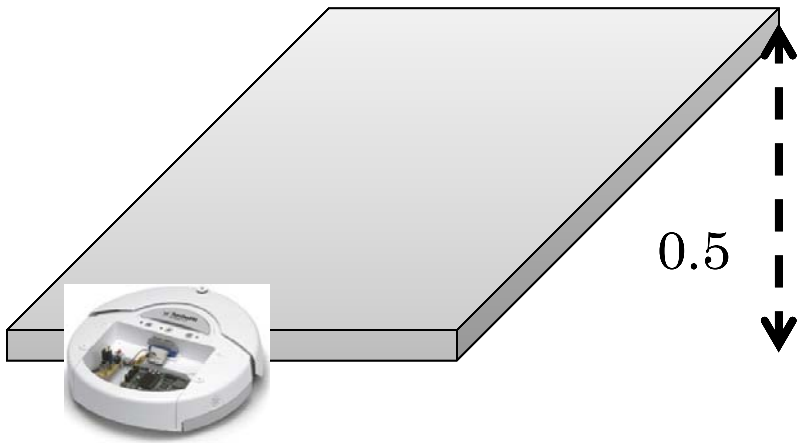
- *Auto-grading* = verification + debugging

- Employ *Simulation-based (run-time) verification*
  - get simulation trace
  - monitor *signal temporal logic* properties
  - localize *faulty* behavior

Fault Detection

- *Environment*: Arena composed of obstacles and hills
- *Monitor*: Signal Temporal Logic formula that captures presence of fault in a trace
- *Test*: Environment + Monitor

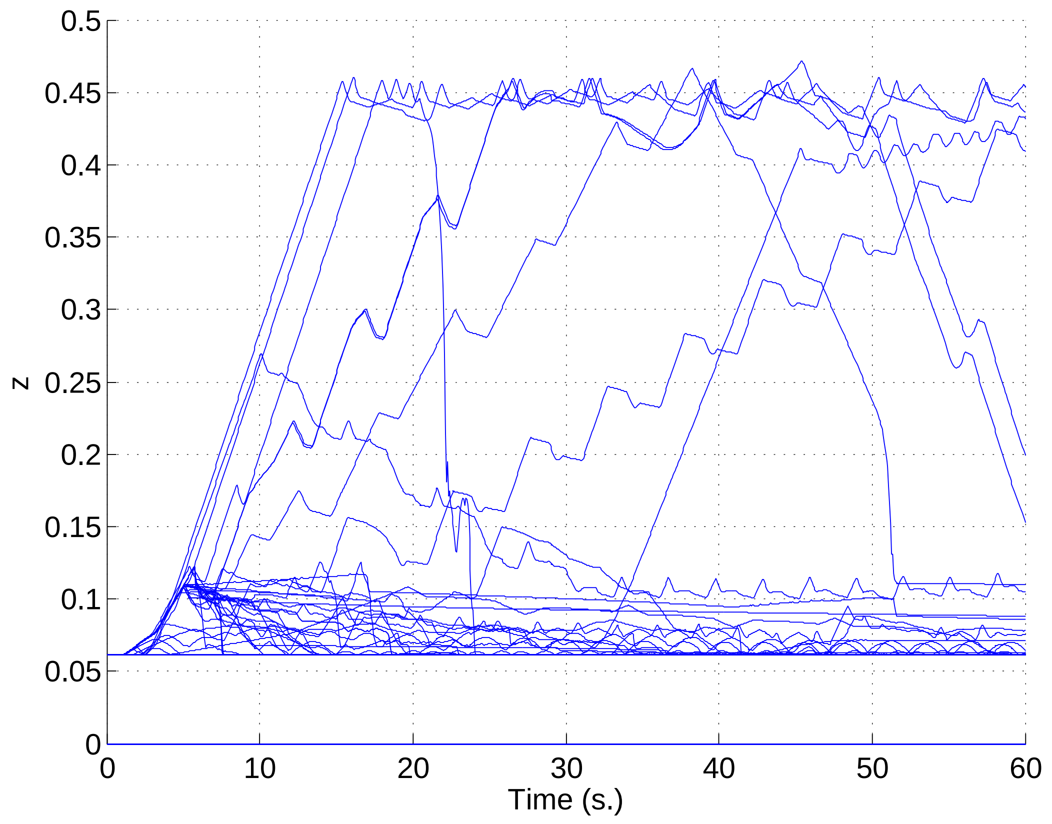
A test is "triggered" by a controller if the fault property holds on the simulation trace in the environment.



+ $\square_{[0,60]}\,\text{pos}.z < 0.4$

Technical Challenge

- Grading should be robust to variations in environment and student solutions.
  - Obstacle placement; hill incline & height
  - Different wheel speeds; strategies.



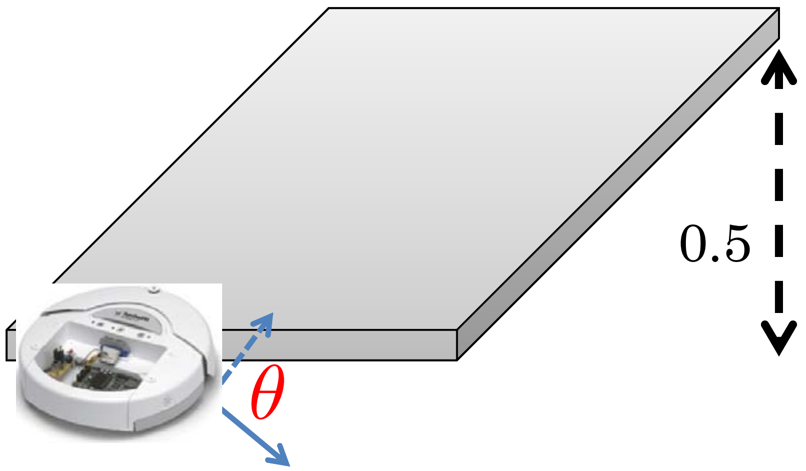
Technical Challenge

- Grading should be robust to variations in environment and student solutions.
  - Obstacle placement; hill incline & height
  - Different wheel speeds; strategies.

- Introduce parameters in environment and STL formula.
- Creating *temporal logic test benches* = solving a *parameter synthesis* problem.

Parameterization

- Generate a collection of tests (*parameter space*)



Test Bench

+

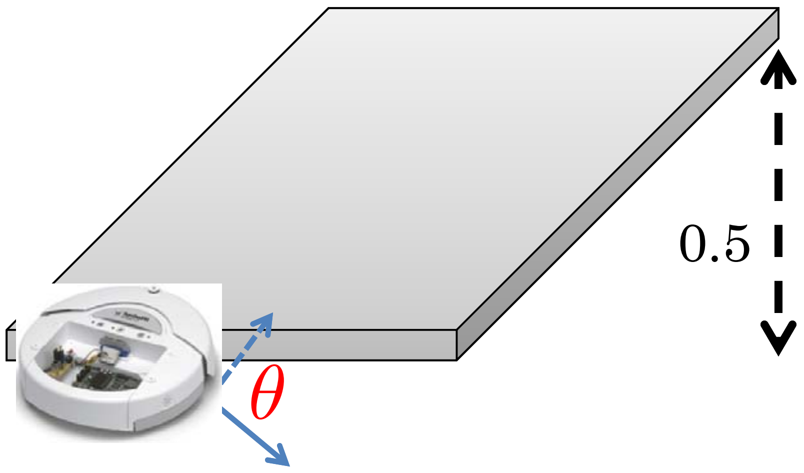
$$\Box_{[0,\tau]}\,\mathsf{pos}.z < h$$

+

$$[\theta \to (-180, 180), \tau \in \mathbb{R}^+,$$
$$h \in \mathbb{R}^+]$$

Parameterization

- Generate a collection of tests (*parameter space*)
- Only a subset of this collection is indicative of the fault.
  (*fault subspace*)
- If *at least* one test from this subspace is triggered, we label the controller as faulty.



Test Bench

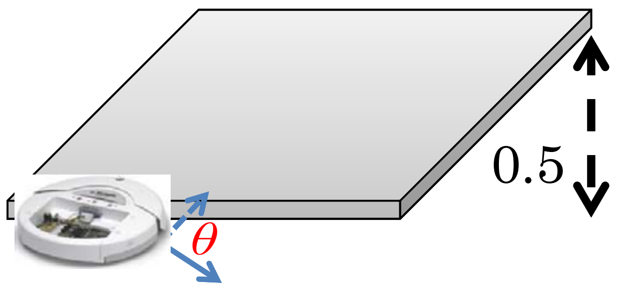
+

$$\Box_{[0,\tau]}\,\mathsf{pos}.z < h$$

+

$$[\theta \mapsto (-90, 90), \tau \mapsto (60, +\infty), h \mapsto (0, 0.4)]$$

Synthesis of Test Benches: Challenges



Test Bench

$\Box_{[0,\tau]}\mathsf{pos}.z < h$

+

+

$[\theta \mapsto (-90, 90), \tau \mapsto (60, +\infty),$
$h \mapsto (0, 0.4)]$

Challenge 1: Finding the fault subspace manually is tedious.

- have to try several variations and inspect traces carefully

Solution: Coming up with ***labeled** reference controllers* is relatively easy → synthesize from examples!

Challenge 2: Fault subspace can be very large

Solution: Minimal Adequate Test Sample

- small finite set of parameter evaluations

Synthesize Fault Subspace from Labeled Reference Controllers: Formal Problem Definition

Given
(1) a parameterized test $\mathbf{\Gamma}$ over a parameter space $\mathfrak{U}$,
(2) two sets of reference controllers:
$\mathcal{C}^{+}$(with fault), $\mathcal{C}^{-}$ (without fault)

Find a fault subspace $\boldsymbol{\rho} \subseteq \mathfrak{U}$, such that the test bench $(\mathbf{\Gamma}, \boldsymbol{\rho})$ correctly labels all controllers in $\mathcal{C}^{+}$ and $\mathcal{C}^{-}$

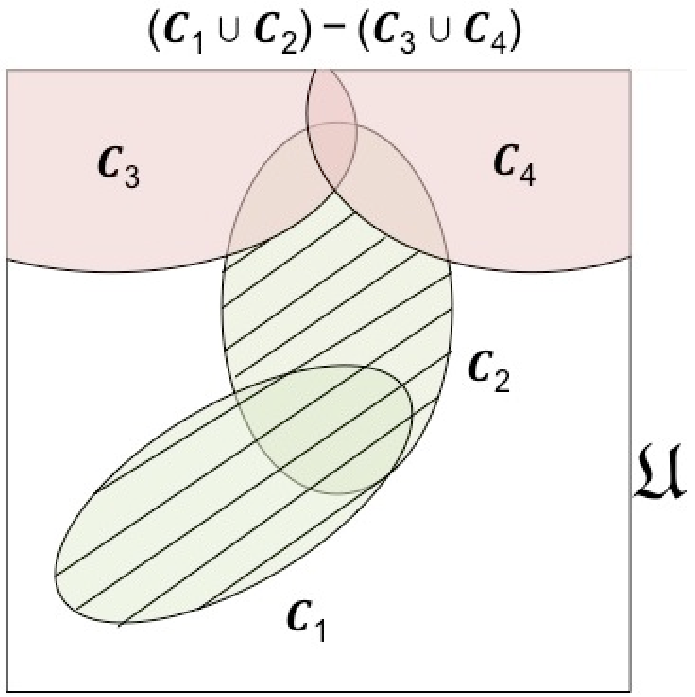
Problem: Given (1) a parameterized test $\mathbf{\Gamma}$ over a parameter space $\mathfrak{U}$, (2) two sets of reference controllers: $\mathcal{C}^{+}$(with fault), $\mathcal{C}^{-}$ (without fault). Find a fault subspace $\boldsymbol{\rho} \subseteq \mathfrak{U}$, such that the test bench $(\boldsymbol{\Gamma}, \boldsymbol{\rho})$ correctly labels all controllers in $\mathcal{C}^{+}$ and $\mathcal{C}^{-}$

Solution: Synthesize a region including every test which

– is triggered on ***at least one*** reference controller **with the fault**, but

– is NOT triggered on ***any*** reference controller **without the fault**.

"Lenient" grading



C_1, C_2 – controllers with the fault
C_3, C_4 – controllers without the fault

Relevant Aspects of Our Solution

- Exploits Monotonicity of Tests (PSTL + Env)

For some order $\lessdot$, a parameterized test Γ is monotonic in a parameter p if and only if
$\forall p_1 \lessdot p_2 \,.\, \Gamma(p_1)$ is triggered $\Rightarrow \Gamma(p_2)$ is triggered

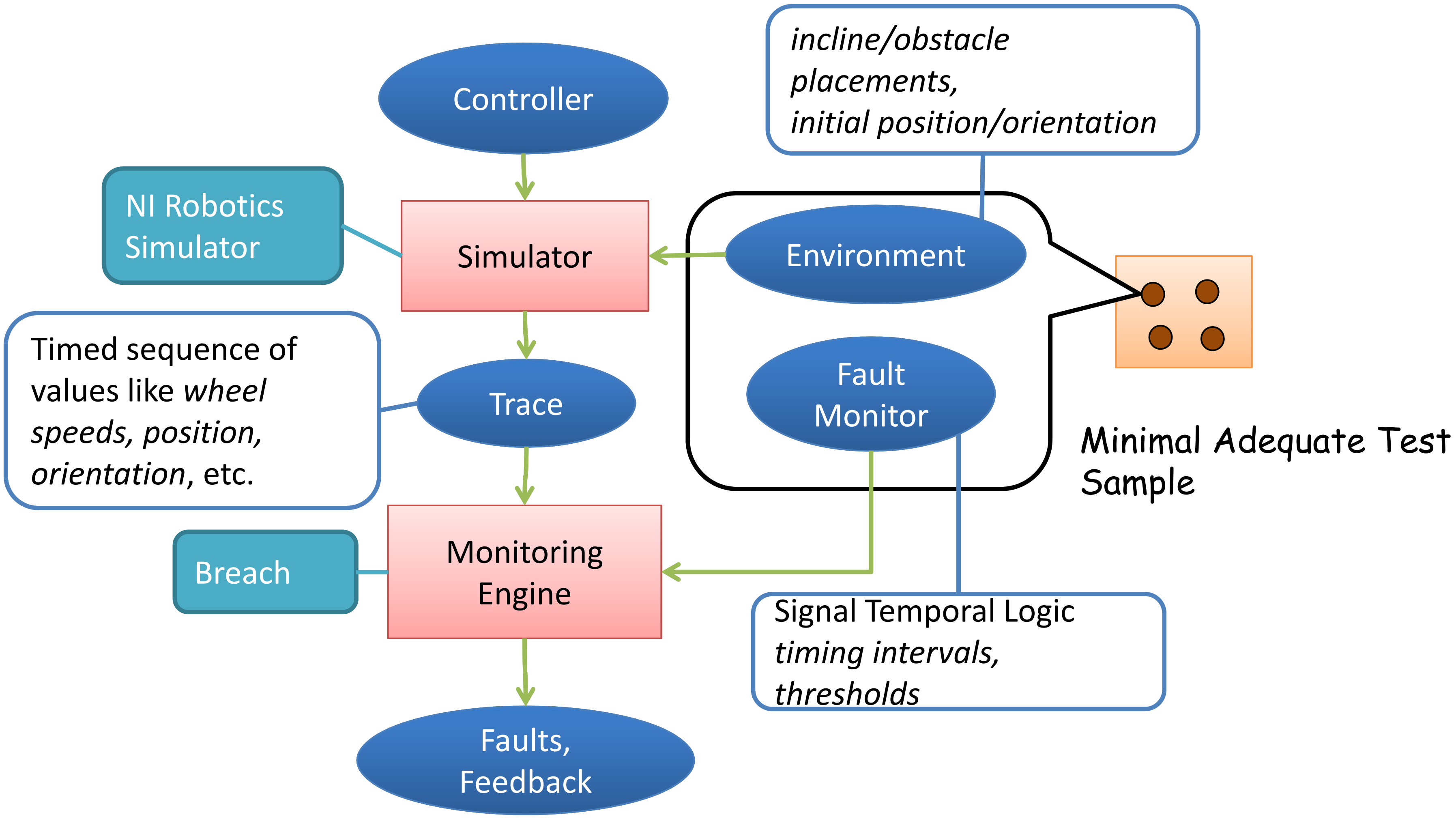
- For k monotonic parameters, efficient fault subspace (minimal test sample) computation:

$$k = 1 \; : O(\log N) \quad k = 2 : O(N) \quad k \geq 3 : O(N^{k-1})$$

  – This is also optimal

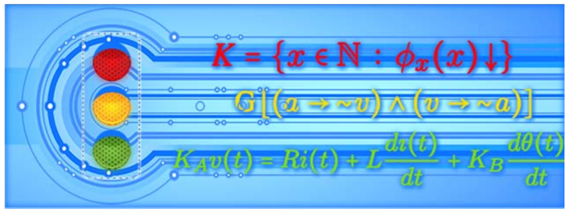
- Evaluation on on-campus dataset (details: EMSOFT'14 paper)
  – Grading accuracy: over 90% on avg
  – Efficiency: <50 sec per grade

Summary of Auto-Grading Flow



Controller
NI Robotics Simulator
Simulator
incline/obstacle placements, initial position/orientation
Environment
Fault Monitor
Minimal Adequate Test Sample
Timed sequence of values like *wheel speeds, position, orientation*, etc.
Trace
Breach
Monitoring Engine
Signal Temporal Logic *timing intervals, thresholds*
Faults, Feedback

EECS149.1x: Basic Statistics


$K = \{x \in \mathbb{N} : \phi_x(x)\downarrow\}$
$\mathbf{G}[(a \rightarrow \sim v) \wedge (v \rightarrow \sim a)]$
$K_A v(t) = Ri(t) + L\frac{di(t)}{dt} + K_B\frac{d\theta(t)}{dt}$

- edX offering: ran 6-7 weeks
- 49 lectures, 10 hours 50 minutes of video
- 6 weekly lab assignments

- Peak Enrollment: 8767
- Largest number submitting any lab: 2213
- Number scoring more than 0: 1543
- Number who passed: 342 (4% of peak enrollment)

edX Students Reporting Auto-grader Feedback as Useful:

86%

Hardware Track: When deploying to the real robot, did you modify your solution from the simulator?

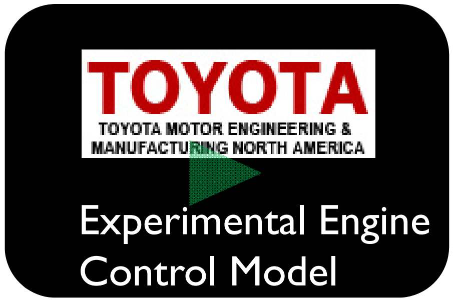
>90%

of controllers that passed the Virtual Lab auto-grader worked on the real robot with no or minor modifications

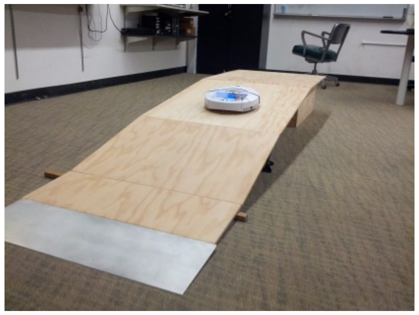
Summary

- CyberSim + CPSGrader
  - Virtual Lab Software + Automatic Grading System
  - http://CPSGrader.org
- Extension uses clustering-based active learning: reduces labeling burden on instructor [Juniwal et al., Learning@Scale'15]
- Exploring applications to other courses in Engineering: Mechatronics, Robotics, Circuits, etc.

Three Stories: Verification by, for, of Humans


TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA
Experimental Engine Control Model

By:
Requirements Mining and Verification for Closed-Loop Control Models (Automotive focus)



For:
Virtual Lab / Automatic Grading System for Massive Open Online Course in CPS

Of:
Design and Verification of Human Cyber-Physical Systems (semi-autonomous driving)

Human Cyber-Physical Systems

Driver Assistance in Cars

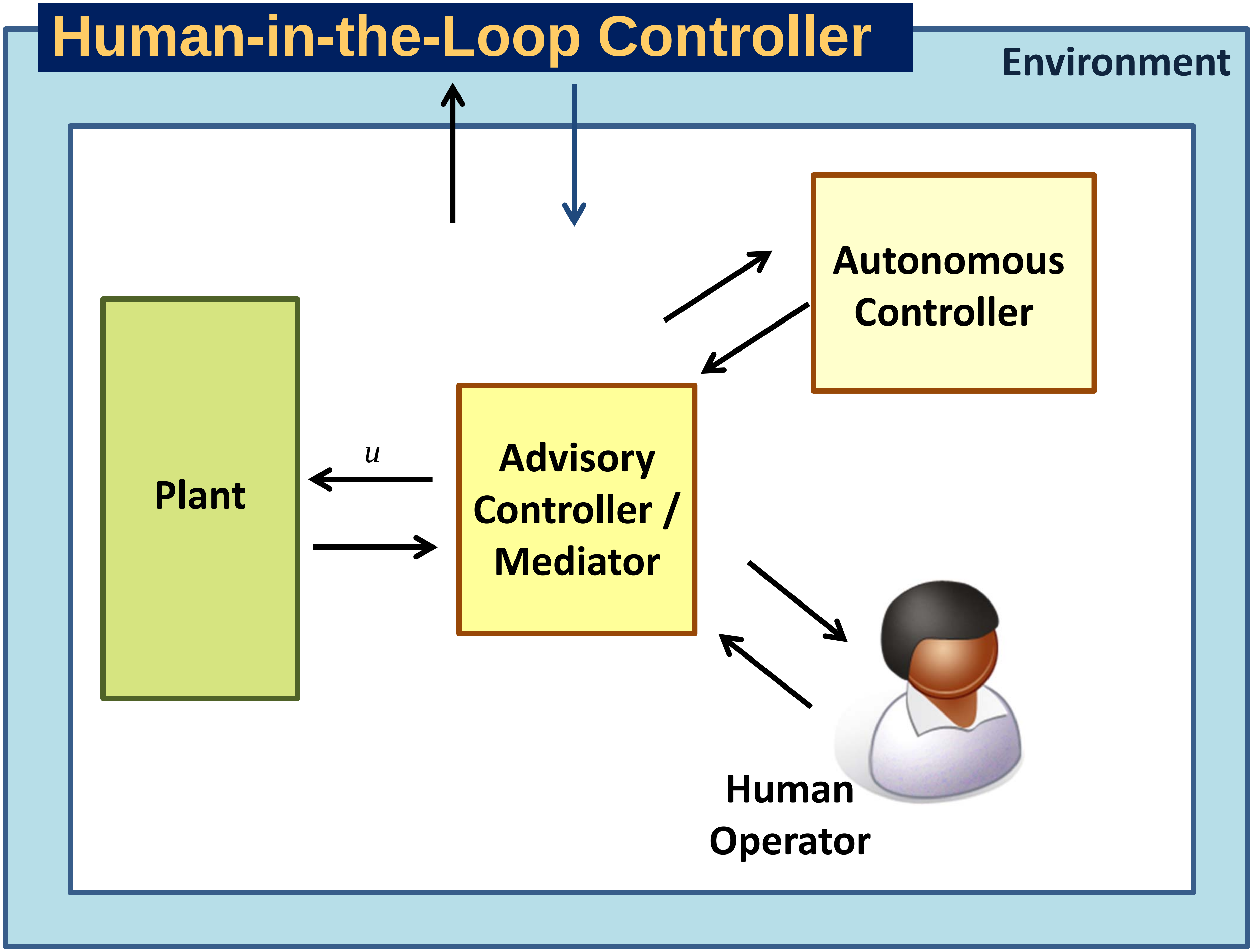
Fly-by-wire Cockpit Interfaces

Robotic Surgery & Medicine

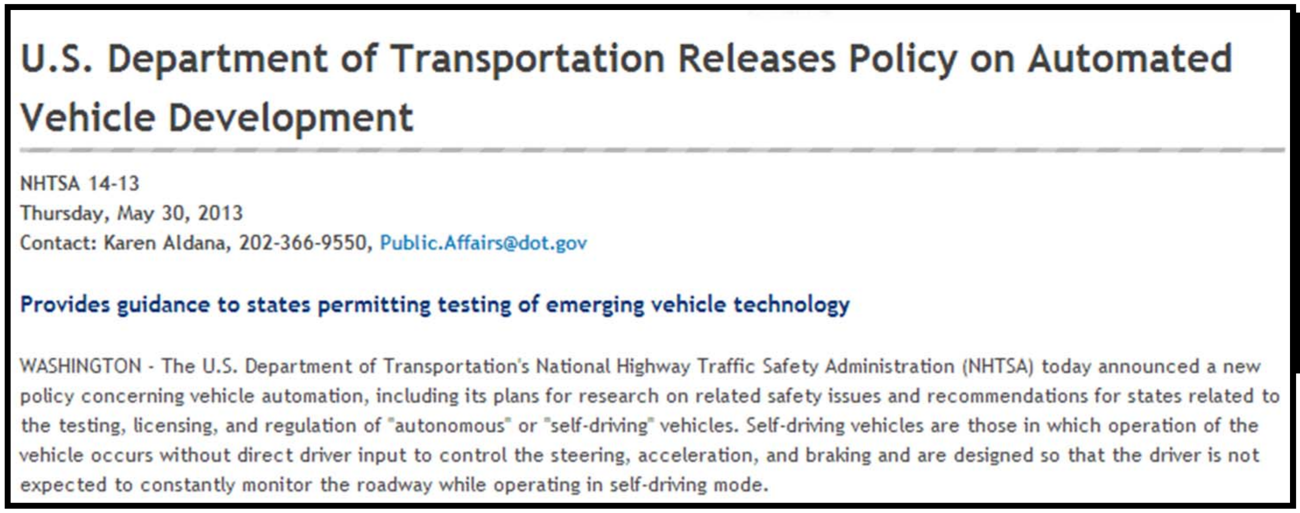
© Rethink Robotics

Semi-Autonomous Manufacturing

http://human-cps.eecs.berkeley.edu


Human-in-the-Loop Controller
Environment
Autonomous Controller
Plant
u
Advisory Controller / Mediator
Human Operator

NHTSA Preliminary Policy Statement, May 2013



U.S. Department of Transportation Releases Policy on Automated Vehicle Development

NHTSA 14-13
Thursday, May 30, 2013
Contact: Karen Aldana, 202-366-9550, Public.Affairs@dot.gov

Provides guidance to states permitting testing of emerging vehicle technology

WASHINGTON - The U.S. Department of Transportation's National Highway Traffic Safety Administration (NHTSA) today announced a new policy concerning vehicle automation, including its plans for research on related safety issues and recommendations for states related to the testing, licensing, and regulation of "autonomous" or "self-driving" vehicles. Self-driving vehicles are those in which operation of the vehicle occurs without direct driver input to control the steering, acceleration, and braking and are designed so that the driver is not expected to constantly monitor the roadway while operating in self-driving mode.

NHTSA
NATIONAL HIGHWAY TRAFFIC SAFETY ADMINISTRATION

- Levels of Automation in NHTSA document
  - Level 0: No Automation
  - Level 1: Function-Specific Automation
  - Level 2: Combined Function Automation
  - Level 3: Limited Self-Driving Automation
  - Level 4: Full Self-Driving Automation

Focus on Level 3: Limited Self-Driving Automation

"Vehicles at this level of automation enable the driver to cede full control of all safety-critical functions under certain traffic or environmental conditions and in those conditions to rely heavily on the vehicle to monitor for changes in those conditions requiring transition back to driver control. The driver is expected to be available for occasional control, but with sufficiently comfortable transition time. The vehicle is designed to ensure safe operation during the automated driving mode."

Specification for Level 3

[Li, Sadigh, Sastry, Seshia, TR 2013; TACAS'14]

4 Requirements common to all Level 3 systems:

- Effective Monitoring
  - Should be able to monitor traffic & environment conditions relevant for correct operation
- Conditional Correctness
  - Should guarantee correct (safe) operation under those conditions
- Prescient Switching
  - Should request driver to take over well in advance (*T* sec advance warning)
- Minimally Intervening
  - Should *rarely* request driver intervention (only when there is high probability of imminent failure)

Formal Specification for Level 3

[Li, Sadigh, Sastry, Seshia, TR 2013; TACAS'14]

4 Requirements common to all Level 3 systems:

- Effective Monitoring
  - Sufficient sensing
- Conditional Correctness
  - "Traditional" formal specification (e.g. temporal logic)
- Prescient Switching
  - Response Time Specification (bound *T*, or fine-grained model)
- Minimally Intervening
  - Cost function

Problem Formulation for Human-in-the-loop (HuIL) Synthesis

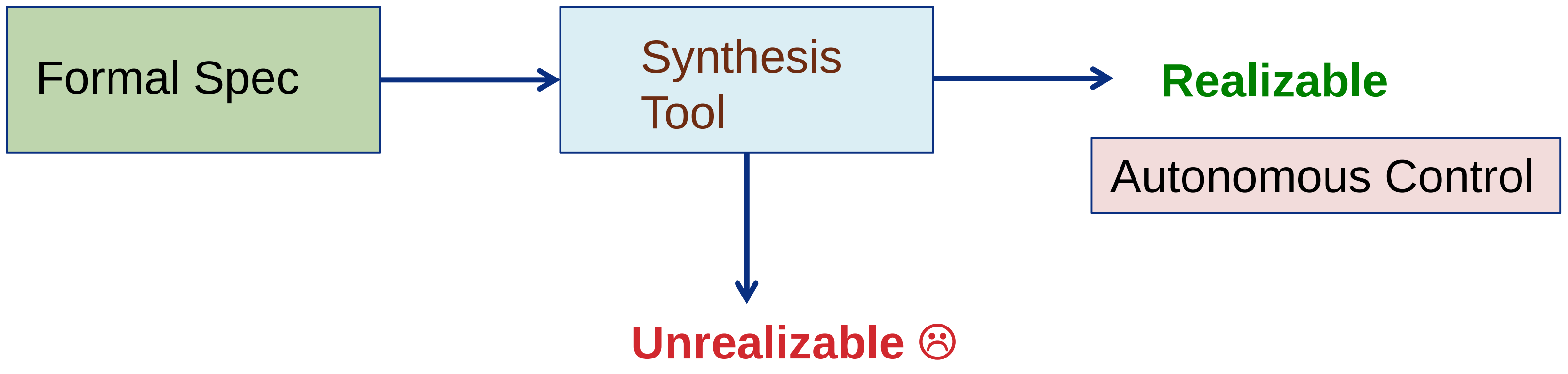
- Given **driver's response time** parameter T
- Given a **cost function** penalizing human's intervention
- Given a **formal specification** (e.g., LTL formula)

Synthesize a **fully autonomous controller** satisfying the specification, or
A Human in the Loop Controller (composition of auto-controller, human operator, advisory controller) that is:

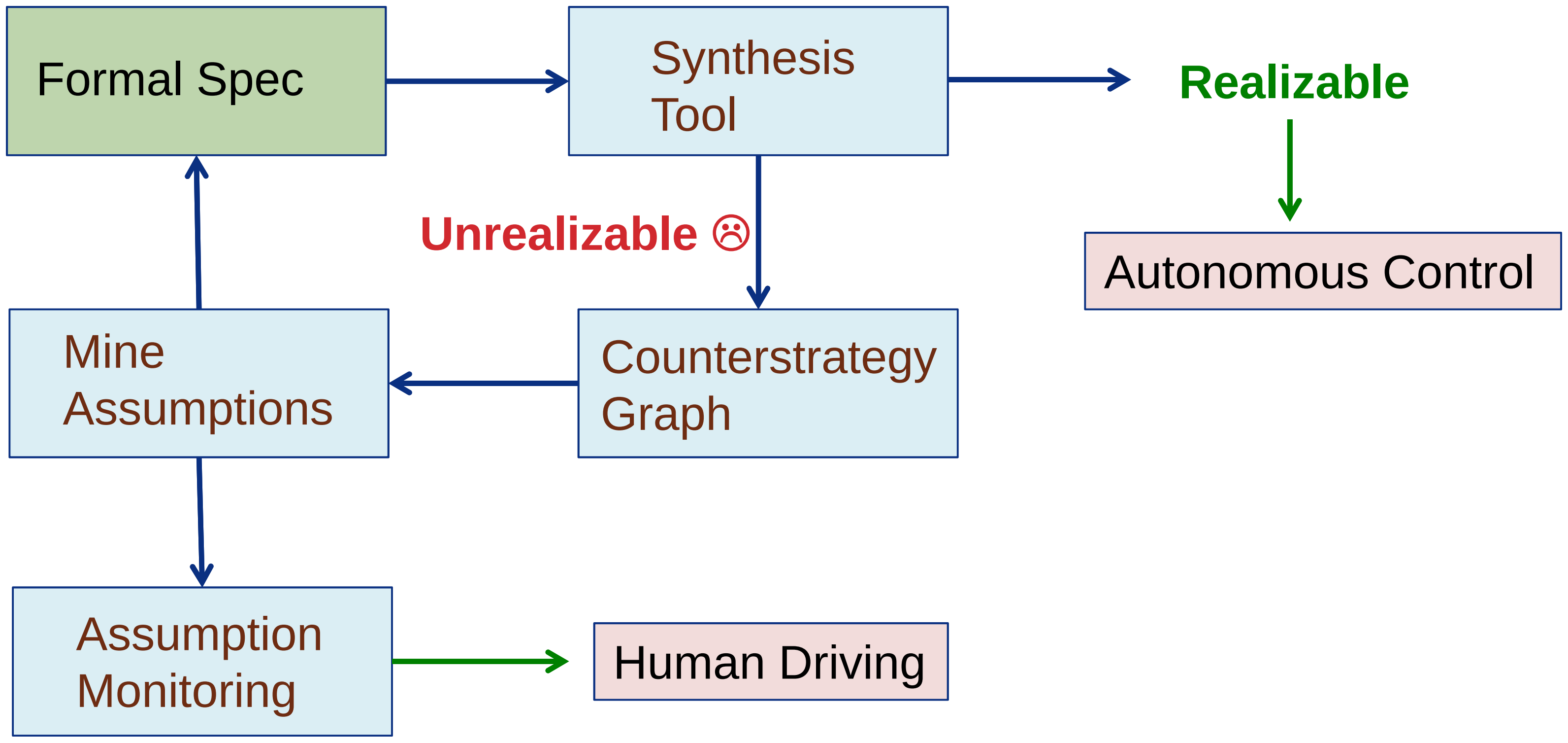
[Effectively Monitoring]
- Prescient (with parameter T)
- Minimally intervening
- Conditionally correct

Li, Sadigh, Sastry, Seshia, "Synthesis for Human-in-the-Loop Control Systems", TACAS 2014.

Approach: Counterstrategy-Guided Synthesis



Approach: Counterstrategy-Guided Synthesis

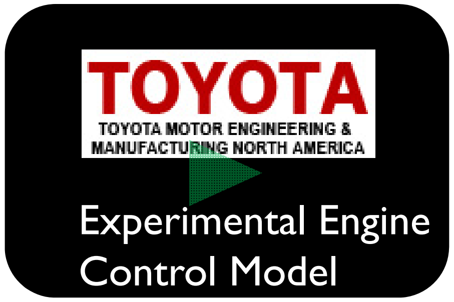


Li, Dworkin, Seshia, "Mining Assumptions for Synthesis", MEMOCODE 2011.
Li, Sadigh, Sastry, Seshia, "Synthesis for Human-in-the-Loop Control Systems", TACAS 2014.

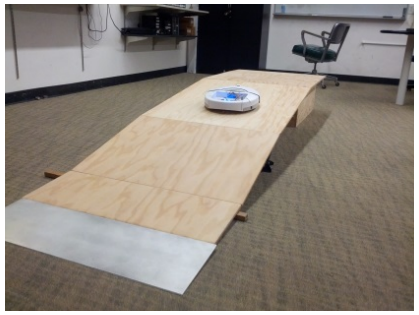
Lots of Future Directions

- Human Perception / Cognition Models
  - Data-driven modeling [Sadigh et al., AAAI Symp. ‘14]
- Extend HuIL Synthesis to broad range of control methods
  - Model-Predictive Control (MPC) variant in progress
- Probabilistic Modeling, Verification, Synthesis

In Conclusion: Verification by, for, of Humans


TOYOTA
TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA
Experimental Engine Control Model

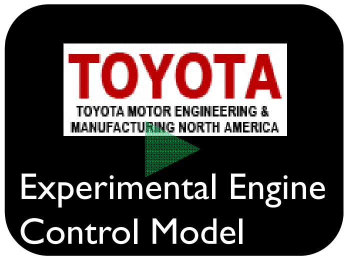
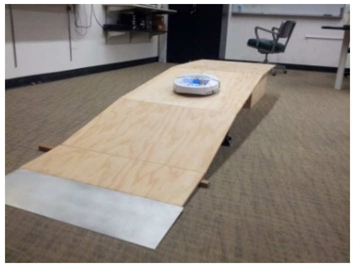
By:
Requirements Mining and Verification for Closed-Loop Control Models (Automotive focus)



For:
Virtual Lab / Automatic Grading System for Massive Open Online Course in CPS

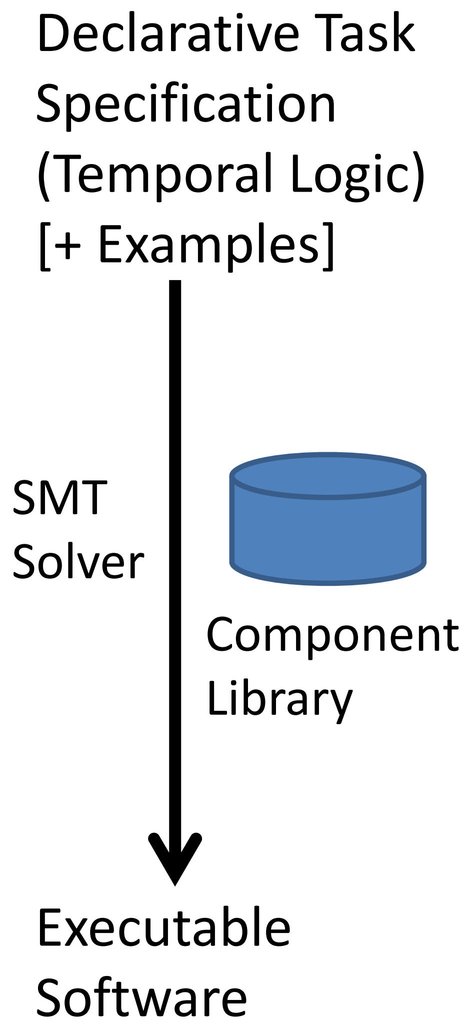
Of:
Design and Verification of Human Cyber-Physical Systems (semi-autonomous driving)

Common Thread: Induction + Deduction + Structure

Story	Induction	Deduction	Structure Hypothesis
 TOYOTA TOYOTA MOTOR ENGINEERING & MANUFACTURING NORTH AMERICA Experimental Engine Control Model	Learning from Counterexample Traces	STL Falsifier	Parametric STL Templates
	Learning from Controllers (reference solns)	STL Run-Time Verifier	Parametric STL Templates
	Learning from Counter-strategies	Automata-theoretic Synthesizer	Efficiently Monitorable "Safe LTL"

(see [Seshia, DAC'12] for other examples)

Multi-Robot Motion Planning from Temporal Logic: Software Synthesis for Robotics


Declarative Task Specification (Temporal Logic) [+ Examples]
SMT Solver
Component Library
Executable Software

Video of Demonstration on Quadrotors