Formal Methods, Machine Learning, and Cyber-Physical Systems

Sanjit A. Seshia

Professor

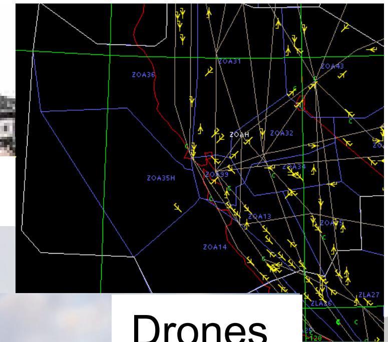
UC Berkeley

ATVA 2018 Tutorial
October 7, 2018

Cyber-Physical Systems (CPS):

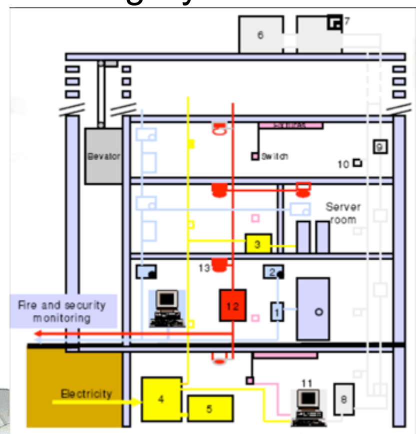
Integration of computation with physical processes, defined by both cyber & physical

Avionics

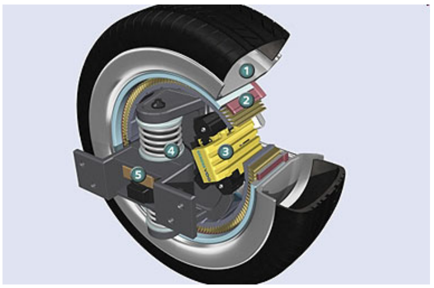


Transportation (Air traffic control at SFO)

Building Systems



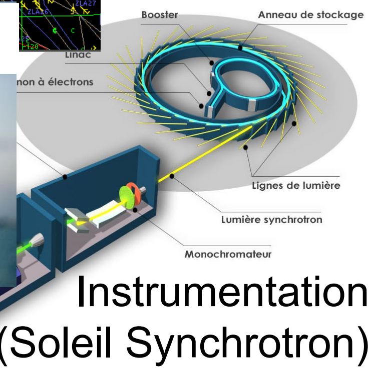
Automotive



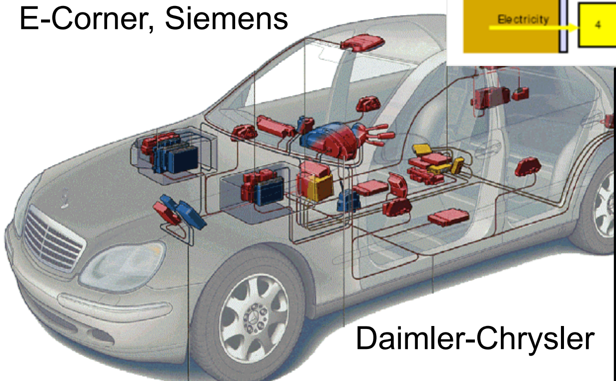
E-Corner, Siemens

Telecommunications

Drones

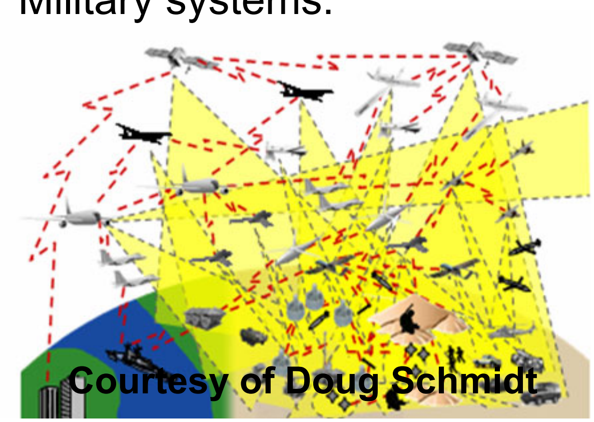


Instrumentation (Soleil Synchrotron)



Daimler-Chrysler

Military systems:

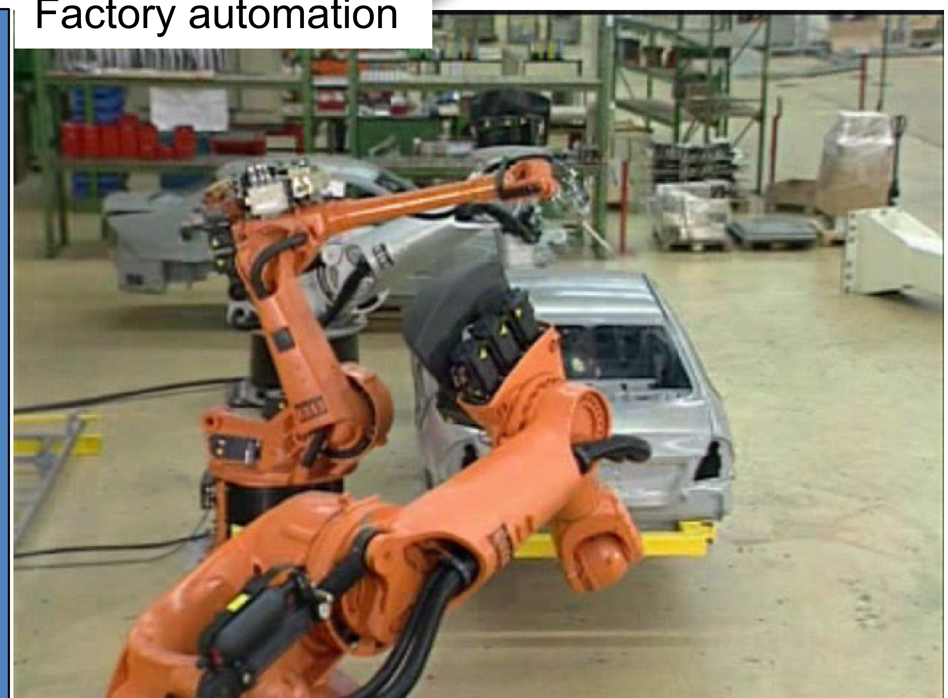


Courtesy of Doug Schmidt

Power generation and distribution

Courtesy of General Electric

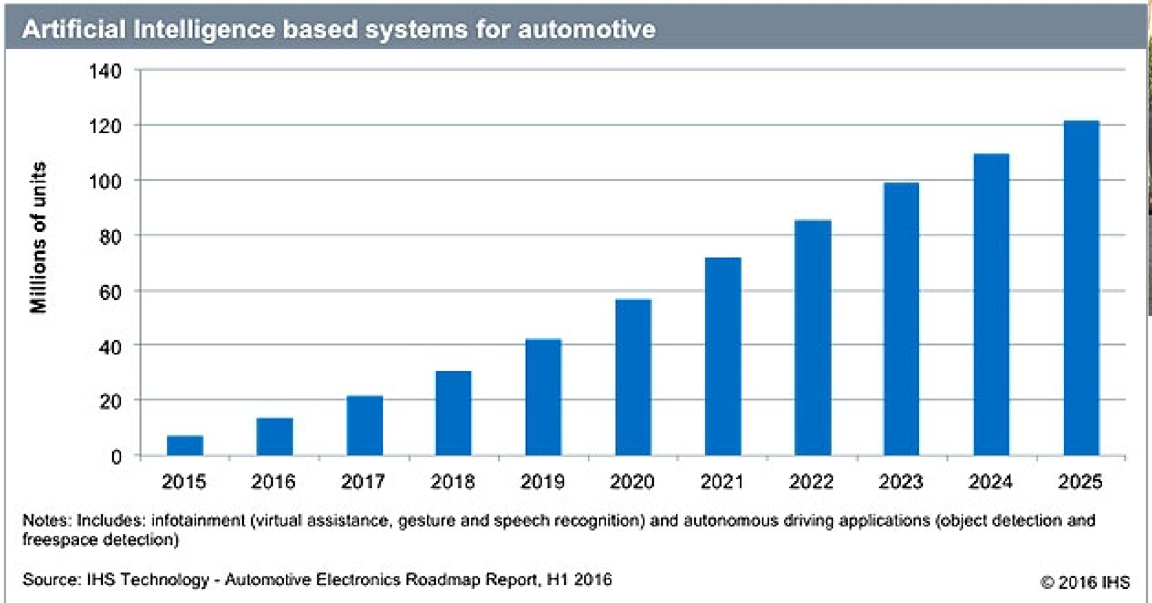
Factory automation

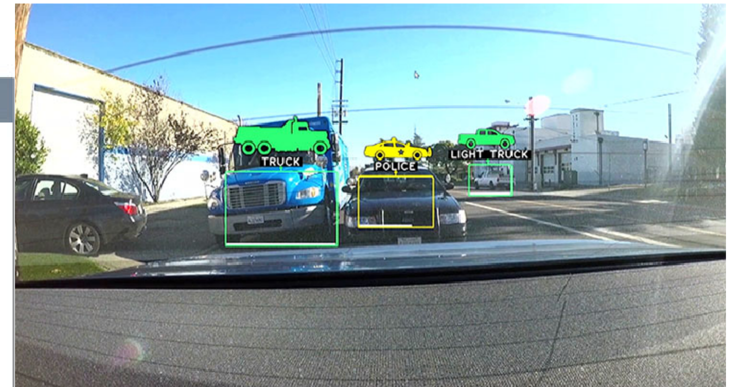


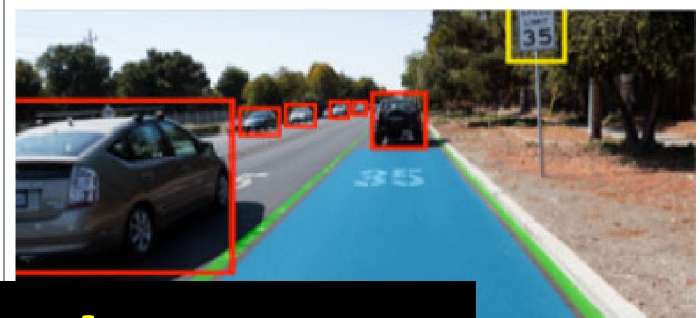
Courtesy of Kuka Robotics Corp.

[E. A. Lee]

Growing Use of Machine Learning/AI in Cyber-Physical Systems



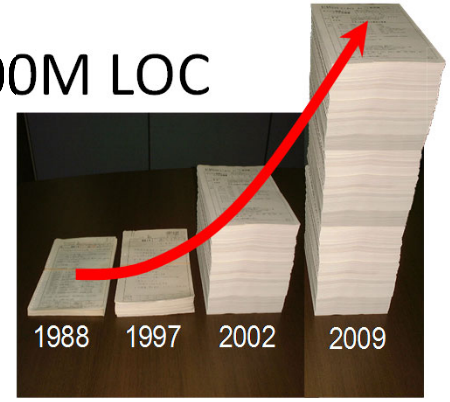




Many Safety-Critical Systems

Growing Features → Growing Costs

- 70 to 100 ECUs in modern luxury cars, close to 100M LOC
- Engine control: 1.7M LOC
  - F-22 raptor: 1.7M, Boeing 787: 6.5M
- Frost & Sullivan: 200M to 300M LOC
- Electronics & Software: 35-40% of luxury car cost



[from J. Deshmukh]

Charette, R., "This Car Runs on Code", IEEE spectrum,
`http://spectrum.ieee.org/transportation/systems/this-car-runs-on-code`

High Cost of Failures

- Safety-critical: human life at risk
- Recalls, production delays, lawsuits, etc.
- Toyota UA: $1.2B settlement with DoJ in 2014, lawsuits, …
- Tesla/Uber autopilot incidents…

…

Formal Methods to the Rescue?

- Formal methods = Mathematical, Algorithmic techniques for modeling, design, analysis
  - Specification: WHAT the system must/must not do
  - Verification: WHY it meets the spec (or not)
  - Synthesis: HOW it meets the spec (correct-by-construction design)
- Industry need for higher assurance → Increasing interest in Formal Methods
- Major success story: Digital circuit design
- *Can we address the challenges of CPS design?*

Formal Methods meets Machine Learning

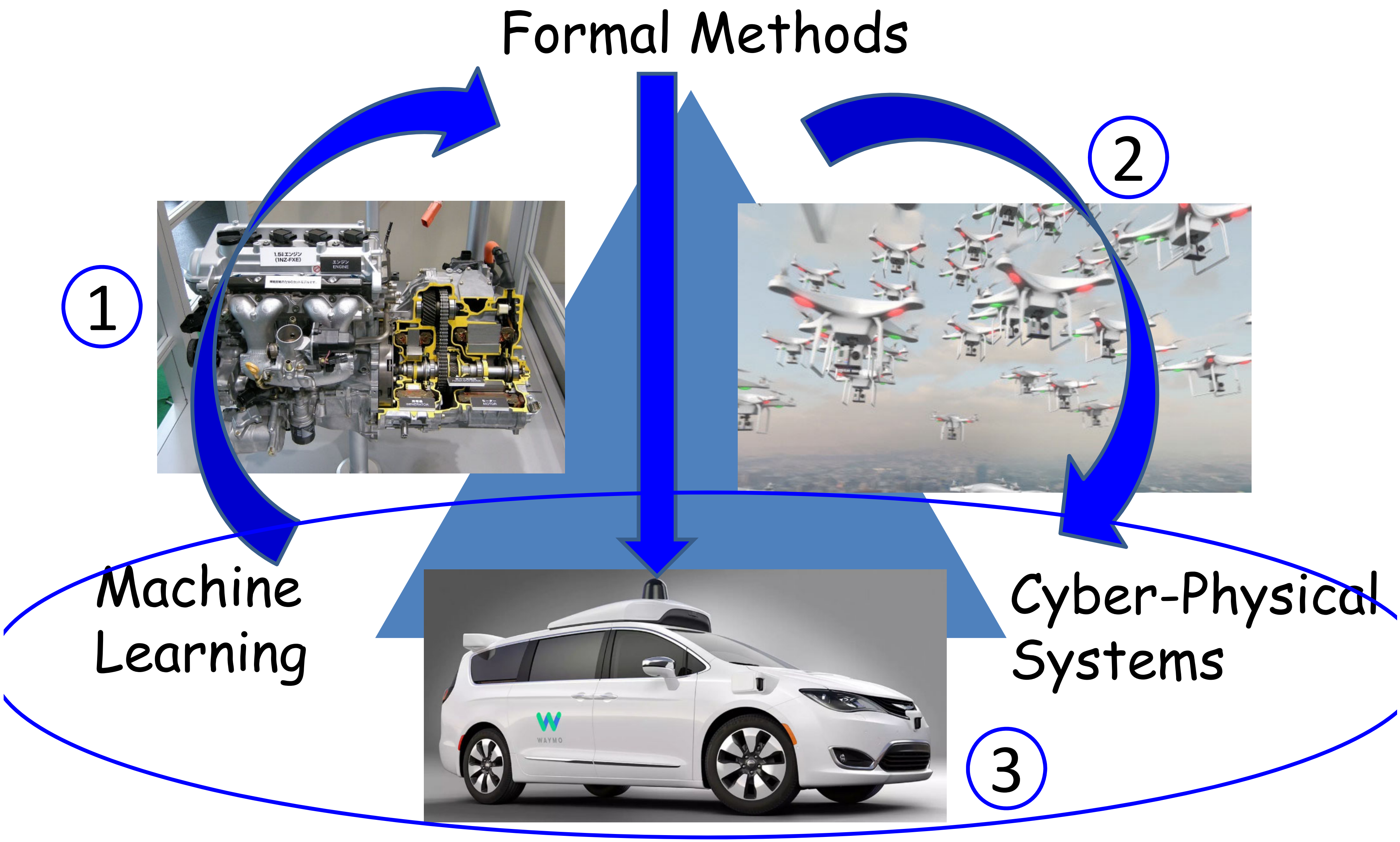
- Machine Learning → Formal Methods
  - Greater efficiency, ease of use/applicability
  - *Formal Inductive Synthesis*

- Formal Methods → Machine Learning
  - Stronger assurances of safety/correctness for learning systems

Further details:

1. S. A. Seshia, "Combining Induction, Deduction, and Structure for Verification and Synthesis", Proceedings of the IEEE, November 2015.
2. S. A. Seshia, D. Sadigh, and S. S. Sastry, "Towards Verified Artificial Intelligence", July 2016, http://arxiv.org/abs/1606.08514

Connections in this Lecture Series



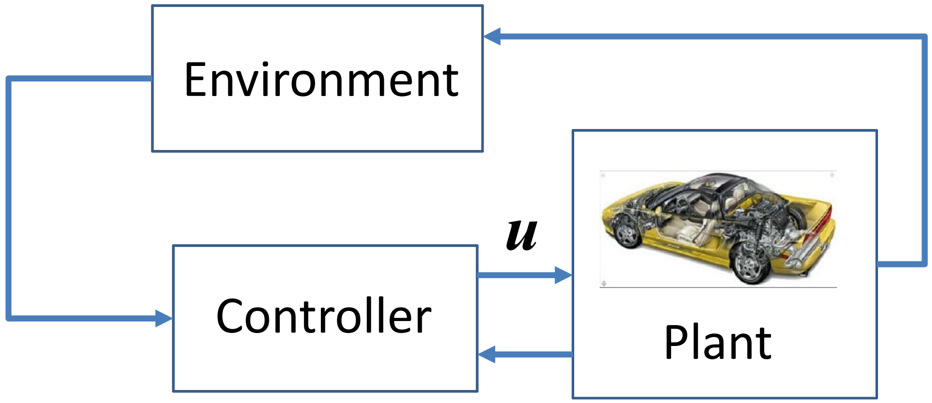
Specification & Verification of Industrial CPS via ***Formal Inductive Synthesis***

Joint work with:

Jyotirmoy Deshmukh, Alex Donze, Susmit Jha, Xiaoqing Jin, Tomoyuki Kaga, Tomoya Yamaguchi

[Jin, Donze, Deshmukh, Seshia, HSCC 2013, TCAD 2015;
Yamaguchi et al. FMCAD 2016;
Jha & Seshia, Acta Inf. 2017]

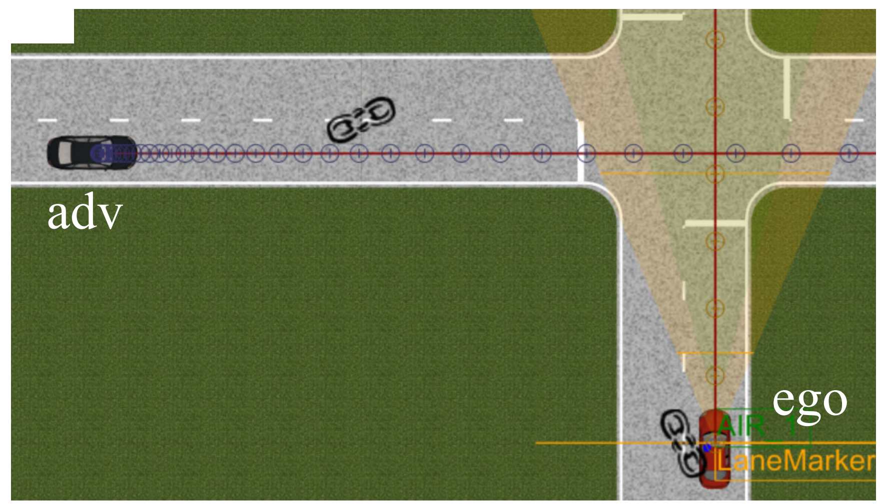
Typical Closed-Loop CPS Model/Verification



$$\mathbf{x}_k = \begin{bmatrix} x_k^{\text{ego}} & y_k^{\text{ego}} & x_k^{\text{adv}} & y_k^{\text{adv}} & v_k^{\text{ego}} & v_k^{\text{adv}} \end{bmatrix}^\top$$

$$\mathbf{u} = a_k^{\text{ego}} \qquad \mathbf{w} = a_k^{\text{adv}}$$

$$\begin{bmatrix} \dot{x}^{\text{ego}} \\ \dot{y}^{\text{ego}} \\ \dot{v}^{\text{ego}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^{\text{ego}} \\ y^{\text{ego}} \\ v^{\text{ego}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \mathbf{u}$$

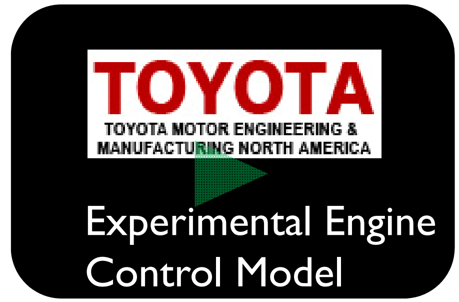


$$\begin{bmatrix} \dot{x}^{\text{adv}} \\ \dot{y}^{\text{adv}} \\ \dot{v}^{\text{adv}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^{\text{adv}} \\ y^{\text{adv}} \\ v^{\text{adv}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \mathbf{w}$$

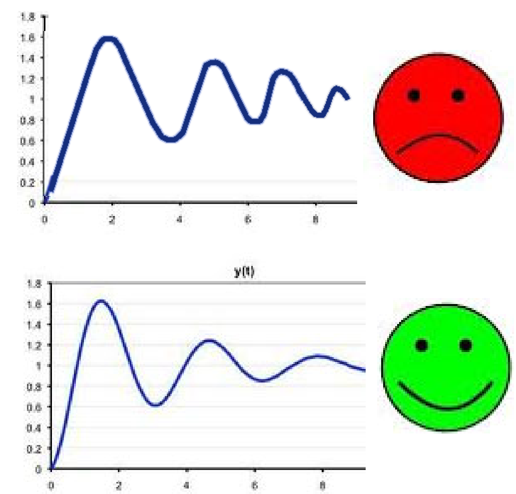
$$\varphi_e = \Box(|\mathbf{w} - \mathbf{w}^{\text{ref}}| < 0.1)$$
$$\varphi_s = \Box(|y_k^{\text{ego}} - x_k^{\text{adv}}| < 2) \implies \Box_{[0,2]}(|v_k^{\text{ego}}| < 0.1)$$

Challenges for Verification of Closed-Loop Automotive Control Systems

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



- 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: Applying Formal Methods to Legacy Systems

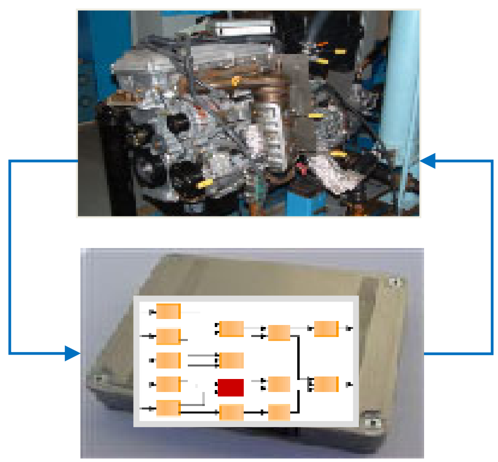
Our Solution: Requirement Mining

Value added by mining:

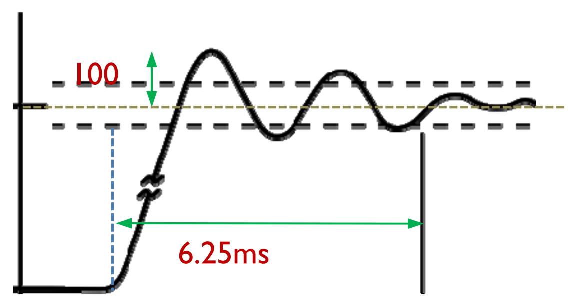
- Mined Requirements become useful documentation
- Use for code maintenance and revision
- Use during tuning and testing

Designer’s View of Our Solution

- Tool extracts properties of closed-loop design using a Simulator

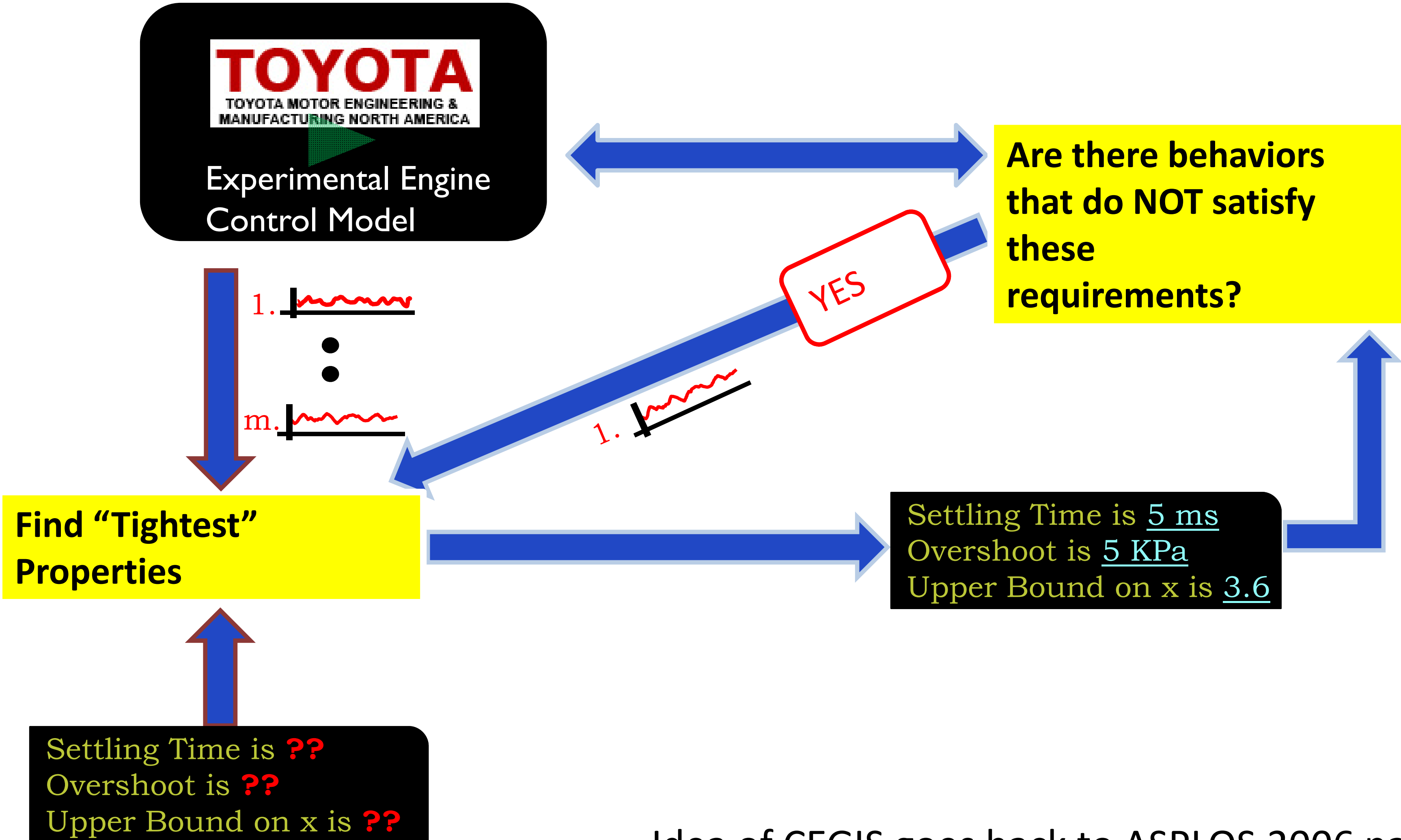


- Designer reviews mined requirements
  - “Settling time is 6.25 ms”
  - “Overshoot is 100 units”
  - Expressed in Signal Temporal Logic [Maler & Nickovic, ‘04]



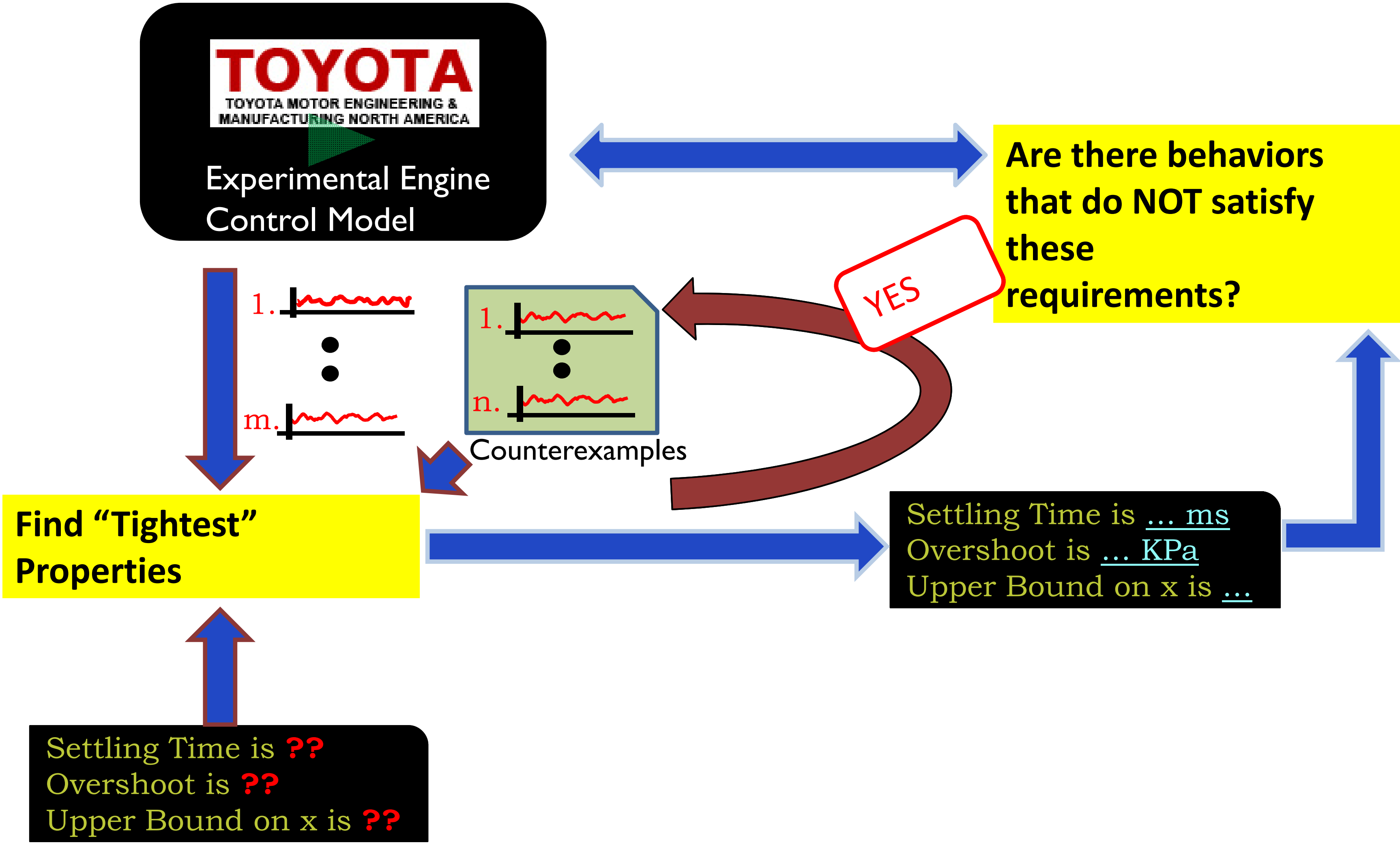
CounterExample Guided Inductive Synthesis (CEGIS)

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

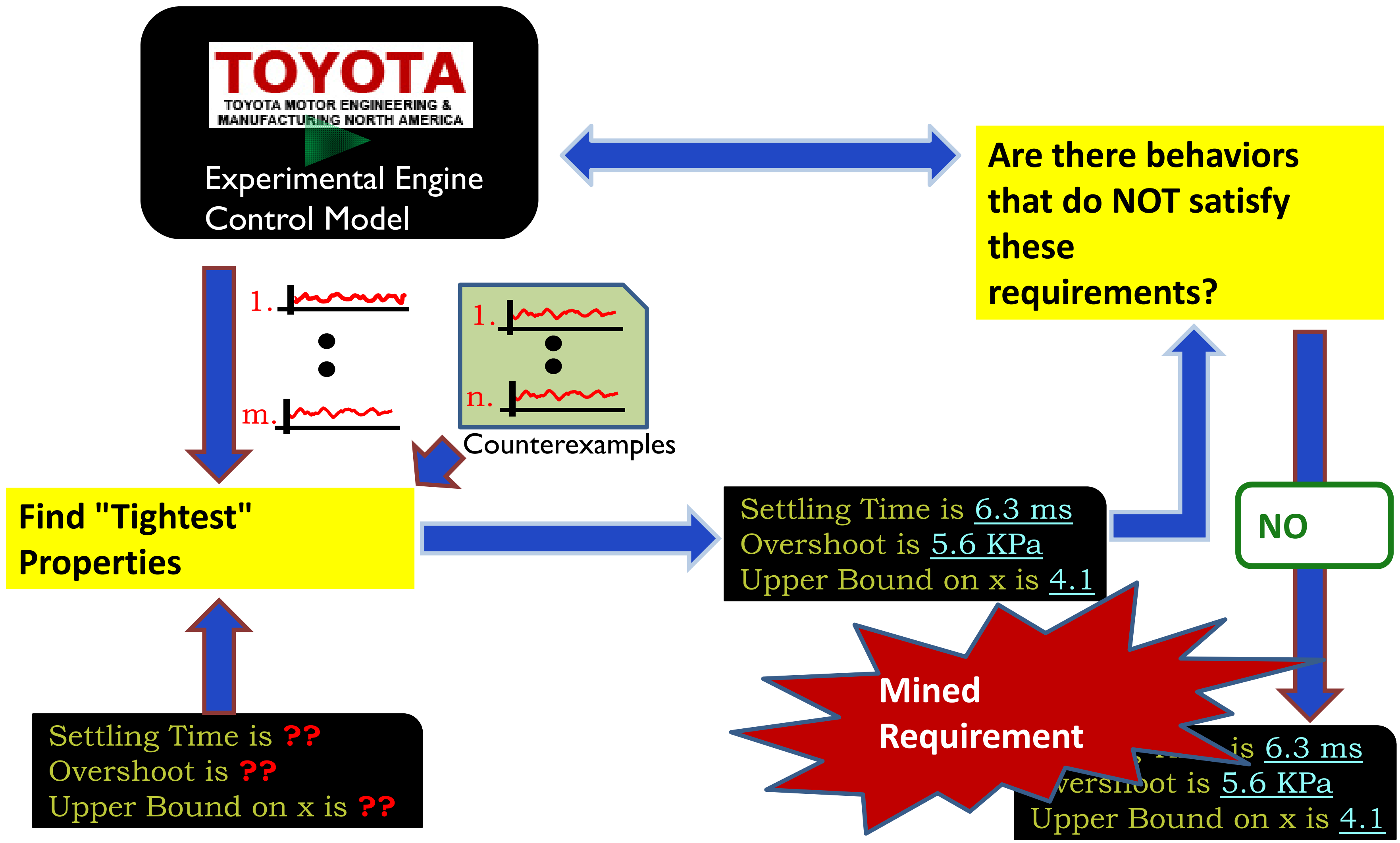


Idea of CEGIS goes back to ASPLOS 2006 paper by Solar-Lezama et al.

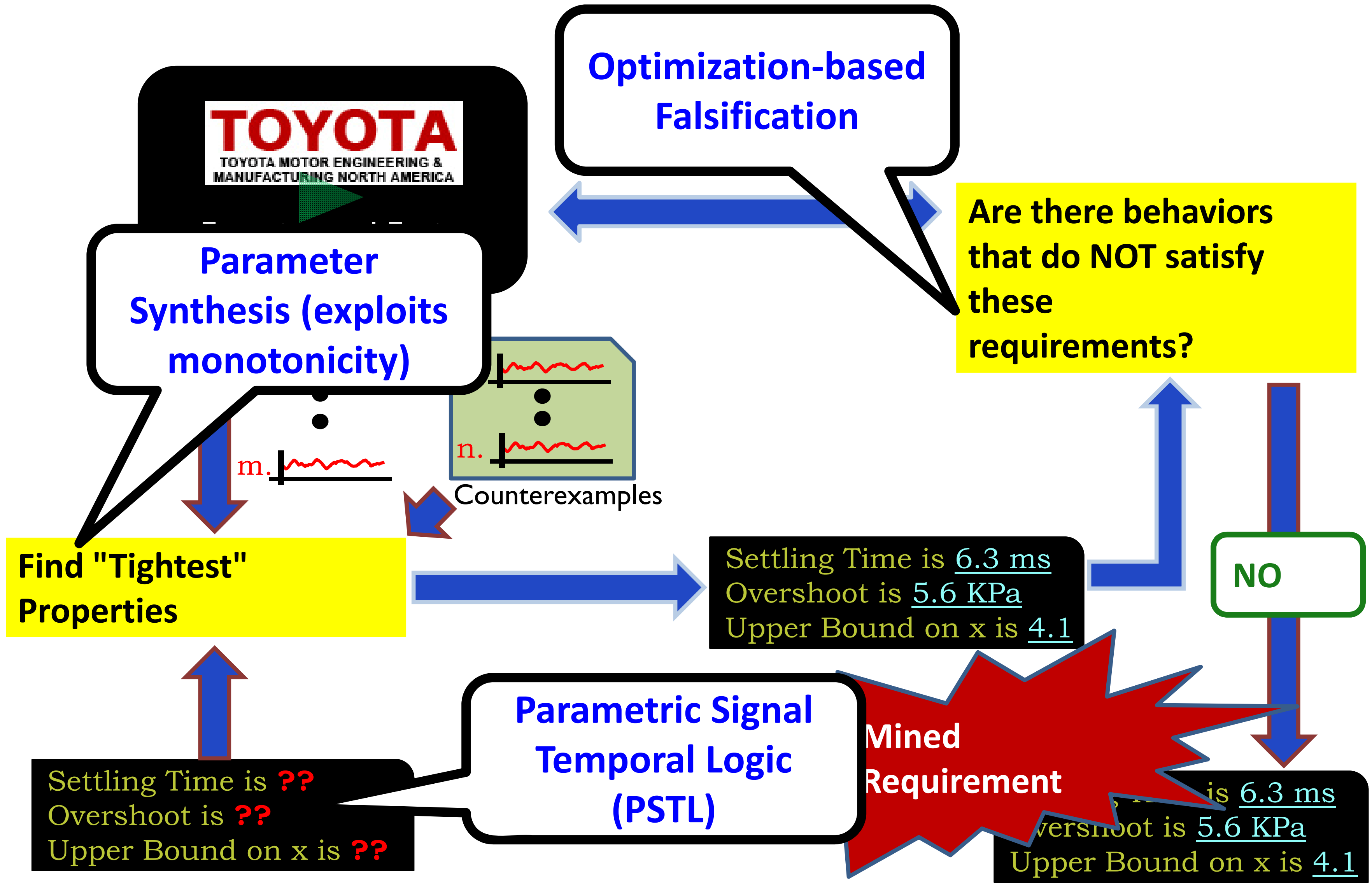
CounterExample Guided Inductive Synthesis (CEGIS)



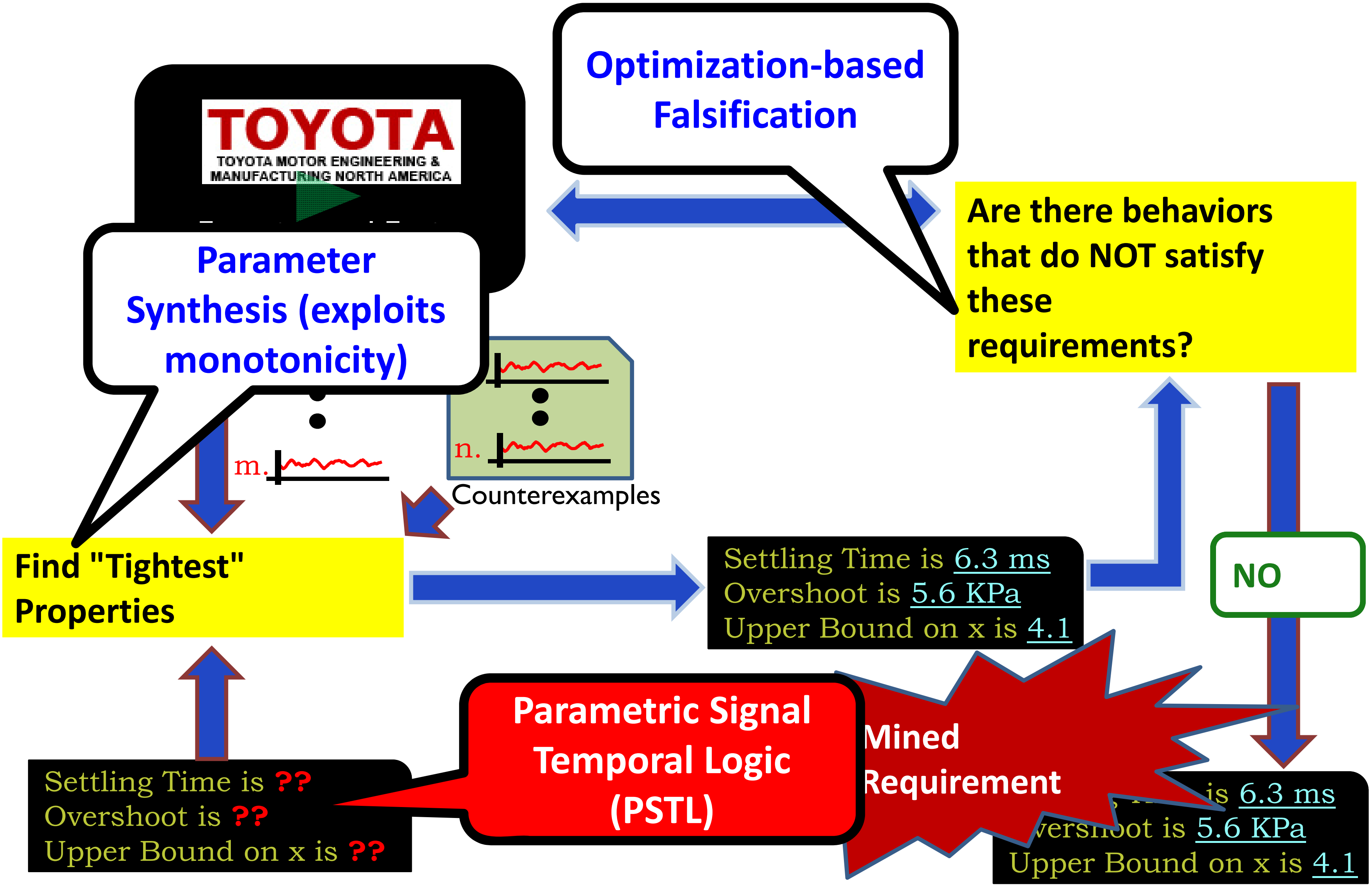
CounterExample Guided Inductive Synthesis



CounterExample Guided Inductive Synthesis



CounterExample Guided Inductive Synthesis

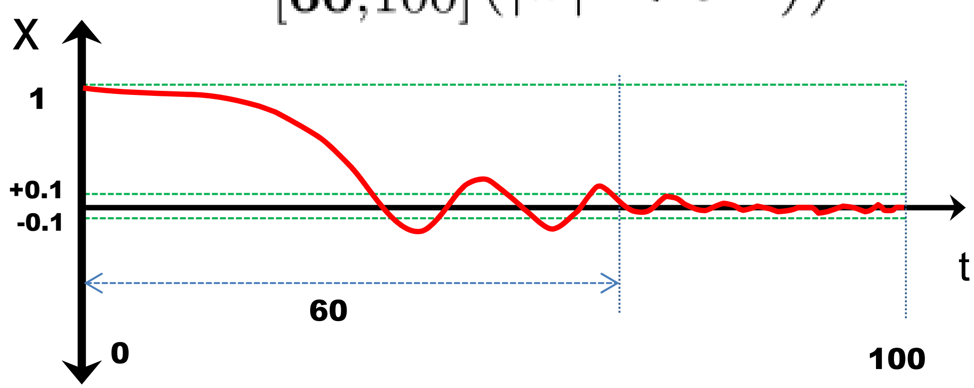


Signal Temporal Logic (STL)

[Maler & Nickovic, 2004]

- 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"*

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



Quantitative Satisfaction Function ρ for STL

- Function ρ that maps STL formula φ and a given trace (valuation of signals) to a numeric value
- Example: $\Box_{[\mathbf{60},100]}(|x| < 0.1))$

  ρ is $\inf_{[60,100]} (0.1 - |x|)$

- Quantifies "how much" a trace satisfies a property
  - Large positive value: trace easily satisfies φ
  - Small positive value: trace close to violating φ
  - Negative value: trace does not satisfy φ

Parametric Signal Temporal Logic (PSTL)

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

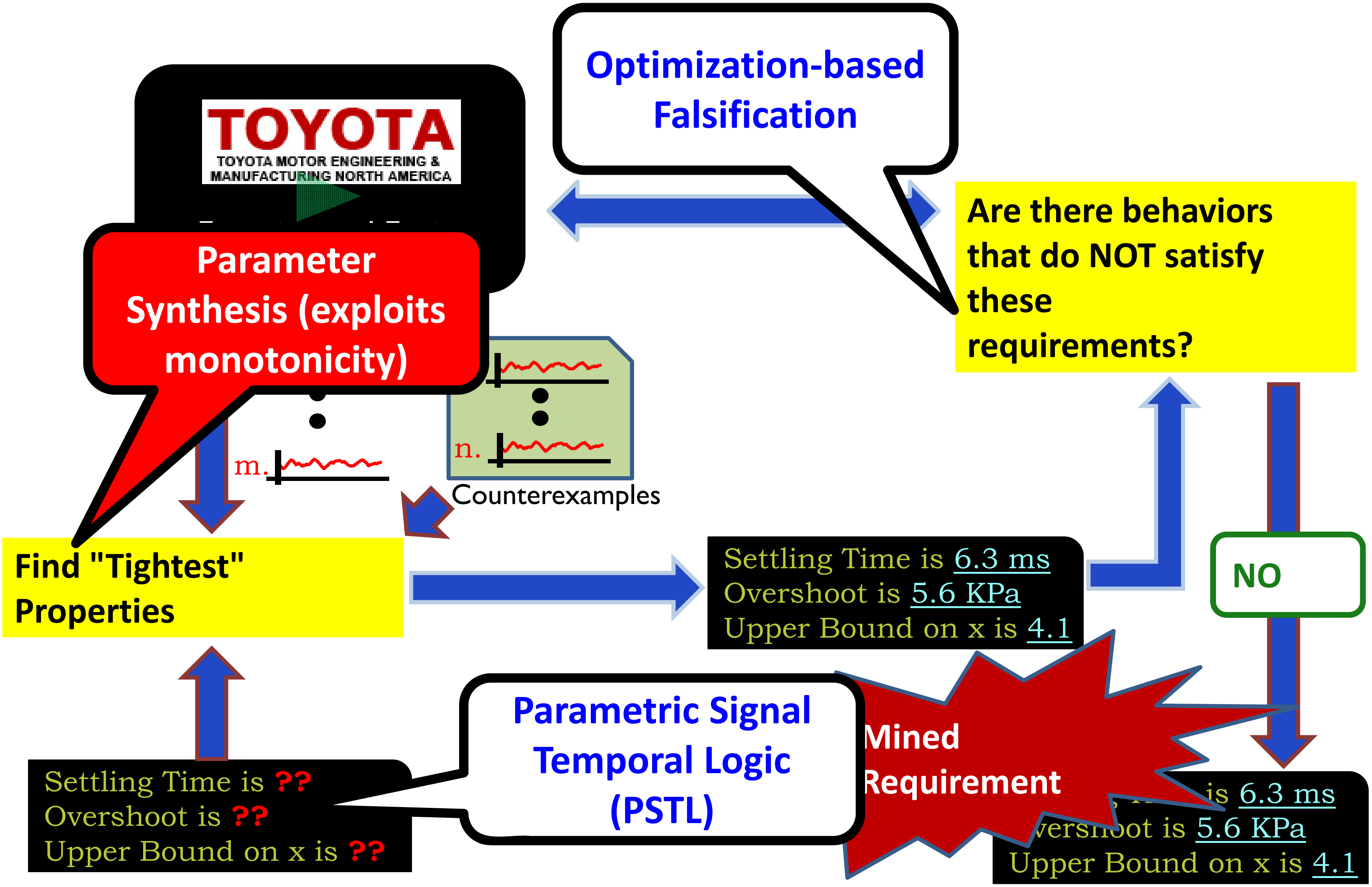
$$\varphi(\tau, \pi) \doteq \Box_{[\tau,10]}(x > \pi)$$

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

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

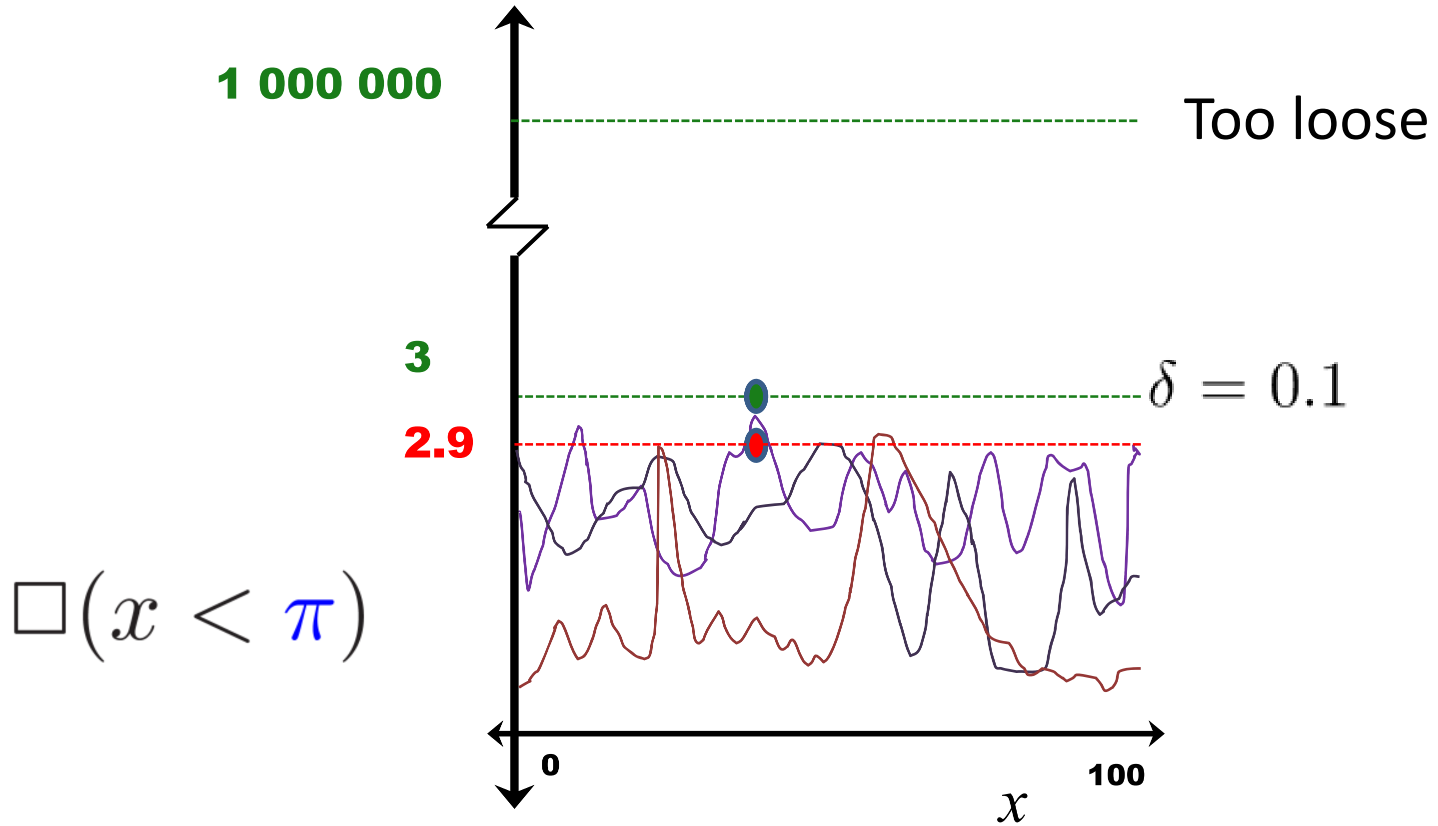
After transmission shifts to gear 2, it remains in gear 2 for at least τ secs

CounterExample Guided Inductive Synthesis



Parameter Synthesis = Find δ-tight values of params (for suitably small δ)

Find "Tightest" Properties



Want the value of π corresponding to the “tightest” satisfaction over a set of traces

Find "Tightest" Properties

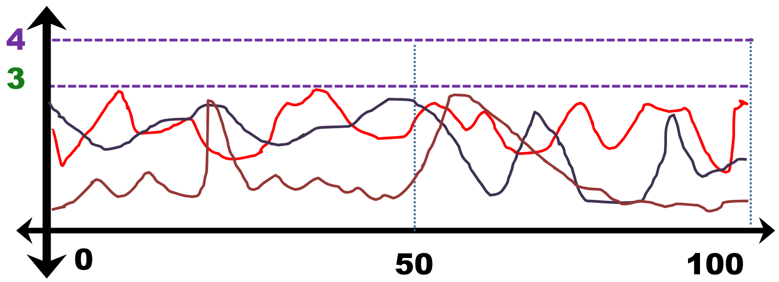
Parameter Synthesis

- Non-linear optimization problem
  - Satisfaction function for STL is non-linear in general
- Naïve (“strawman”) approach:
  - grid parameter space to δ precision
  - evaluate satisfaction value at each point
  - pick valuation with smallest satisfaction value
- Problem: Exponential number of grid points (in #parameters)

Find "Tightest" Properties

Satisfaction Monotonicity

- Satisfaction function monotonic in parameter value
- Example: $\square(x < \pi)$



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

- $\rho(\pi, x) = \inf_t (\pi - x(t))$
- For all x, $\rho(\pi, x)$ is a monotonic function of π
- Advantage: If monotonic, use binary search over parameter space, otherwise exhaustive search

Deciding Satisfaction Monotonicity

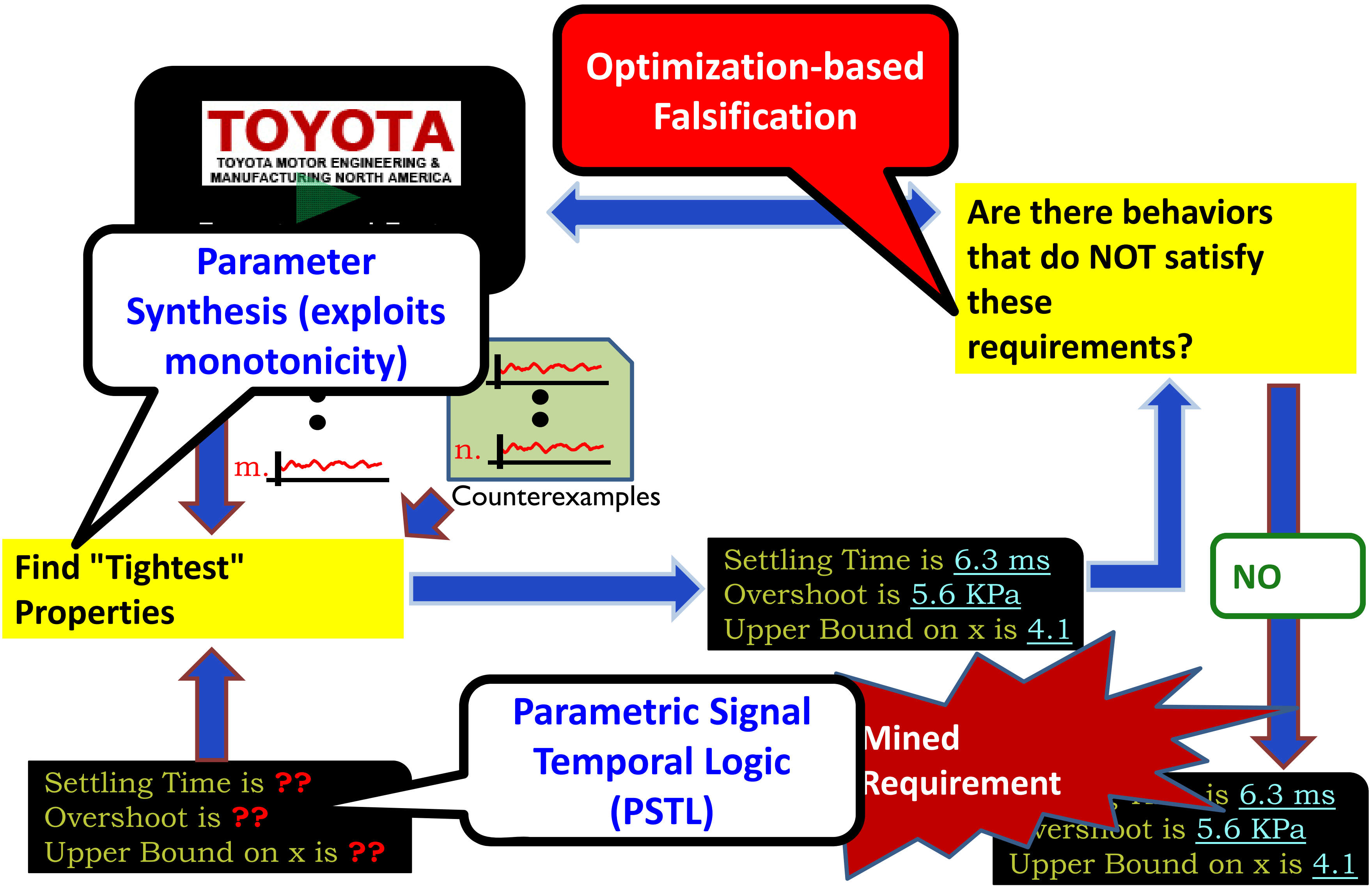
Find "Tightest" Properties

- Need to decide whether:

  For all x, $\rho(\pi, x)$ is a monotonic function of π

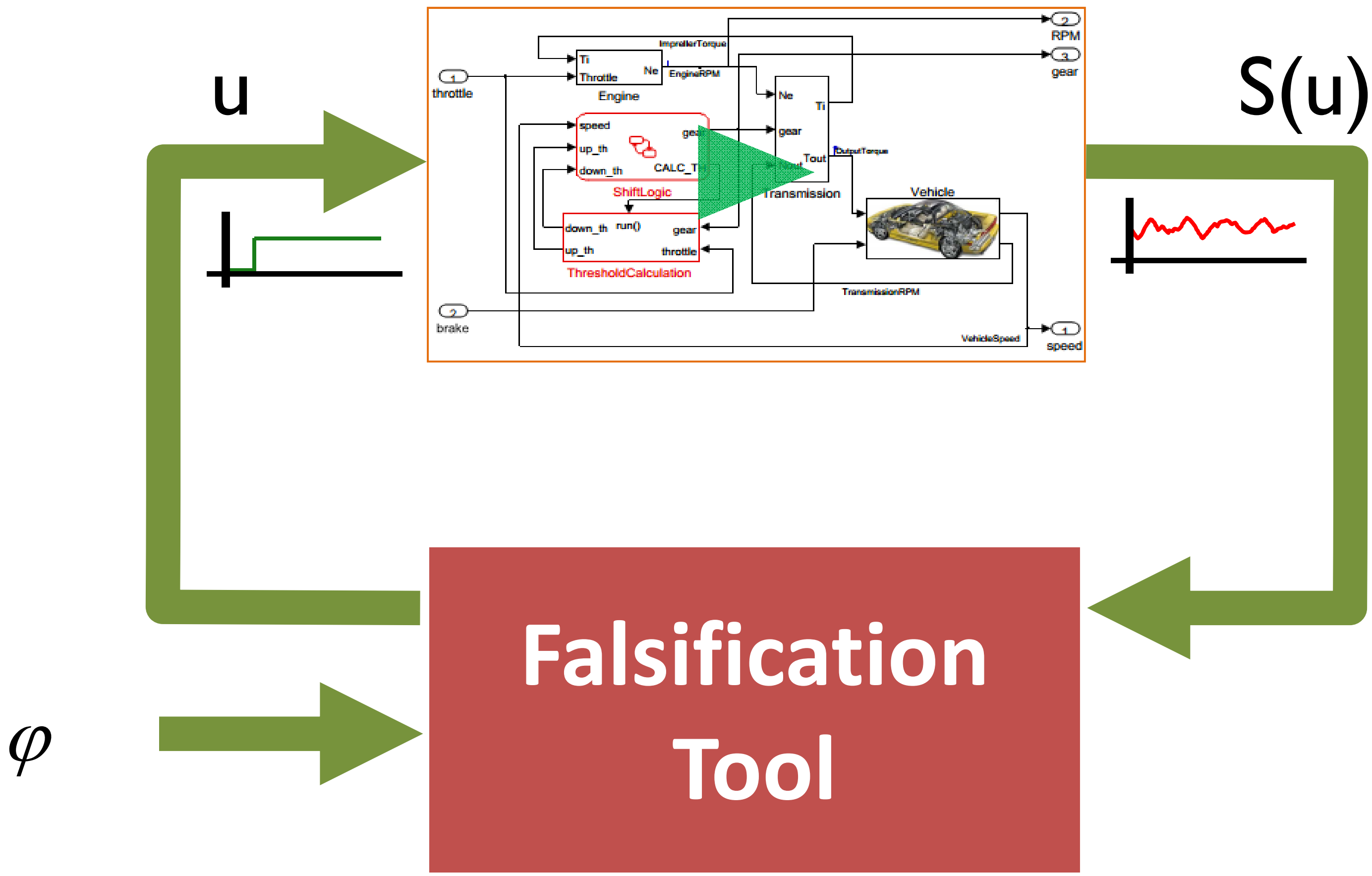
- Theorem: Deciding monotonicity of a PSTL formula is undecidable
- Use an encoding to satisfiability modulo theories (SMT) solving
  - Quantified formulas involving uninterpreted functions, and arithmetic over reals → linear arithmetic if PSTL predicates are linear
  - Solved easily with Z3

CounterExample Guided Inductive Synthesis



Black-Box Falsification Procedure

Are there behaviors that do NOT satisfy these requirements?



Falsification as Optimization

Are there behaviors that do NOT satisfy these requirements?

- Solve $\rho^* = \min_u \rho(\varphi, S(u))$
  - Leverages quantitative semantics of STL
  - Relies on standard numerical optimization methods (e.g. Nelder-Mead)
- If $\rho^* < 0$, found falsifying trace!

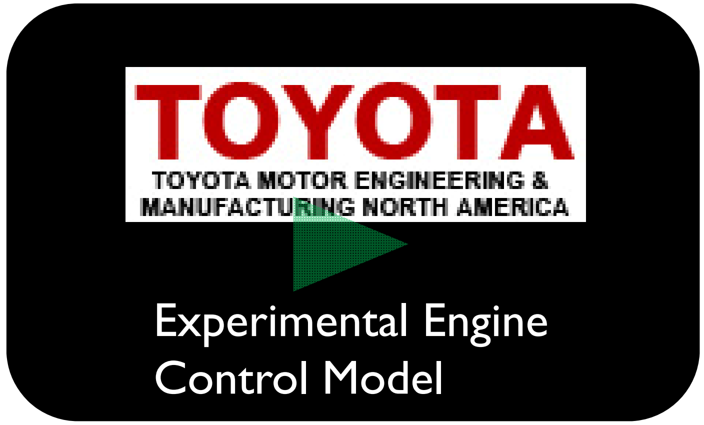
Nonlinear Optimization Problem, No exact solution, Limited theoretical guarantees

Experimental Evaluation Summary **[details in TCAD'15 paper]**

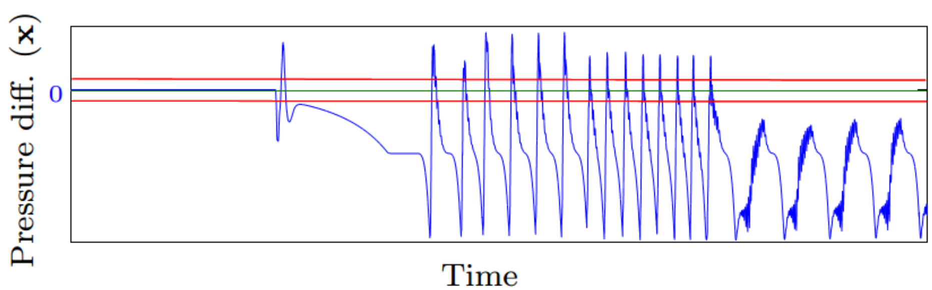
- Defined Templates for Common Requirements in Automotive Control – *all monotonic PSTL!!*
  - Dwell-Time requirements
  - Timed/Untimed Safety properties
  - Timed Inevitability (bounded liveness)
  - Input Profiles: assumptions on shape of input signals
  - Control-theoretic requirements on output signals (bounded overshoot/undershoot, settling time, error from reference signal, etc.)
- Three Benchmarks
  - Simple Simulink Automatic Transmission Model
  - Toyota HSCC'14 Challenge – Air-Fuel Ratio controller
  - Toyota Experimental Diesel Engine Airpath controller

Results on Industrial Airpath Controller

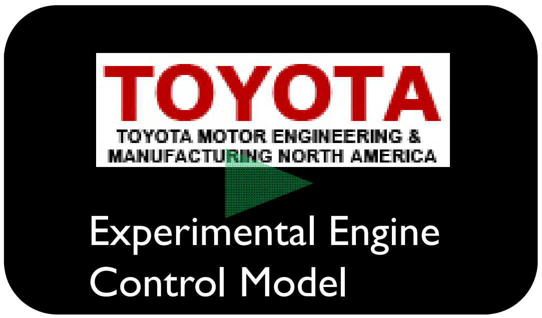
[Jin, Donze, Deshmukh, Seshia, HSCC 2013]



- 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)



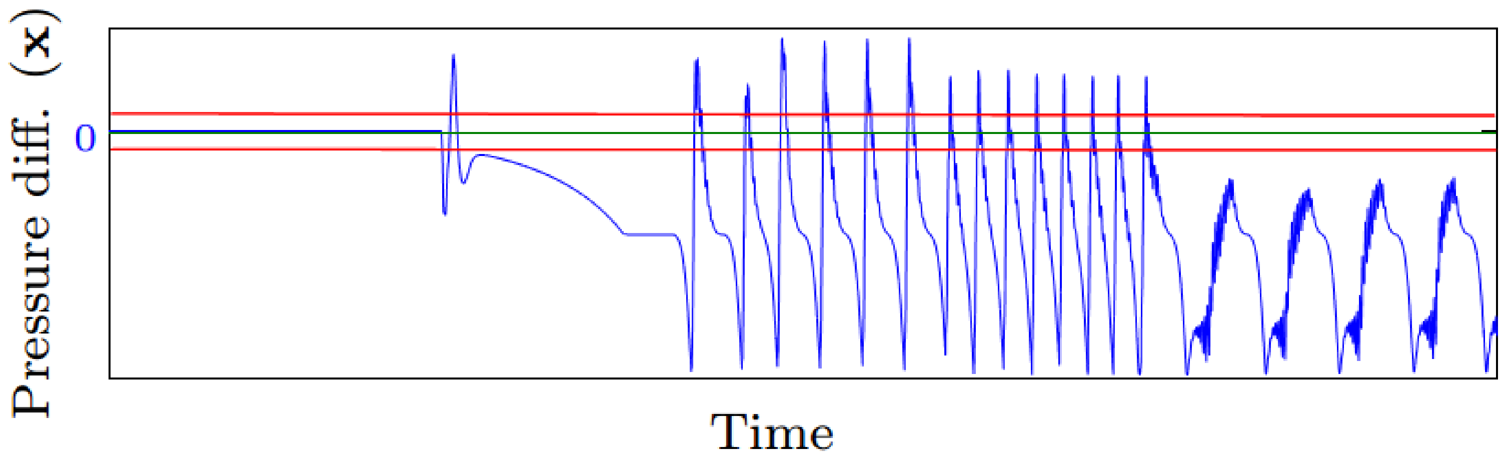
Mining can expose deep bugs

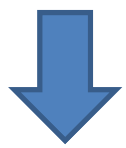


- Uncovered a tricky bug
  - Discussion with control designer revealed it to be a real bug
  - Root cause identified as wrong value in a look-up table, bug was fixed
- Why mining could be useful for bug-finding:
  - Can uncover subtle relations that should not hold
  - Looking for bugs ≈ Mine for negation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

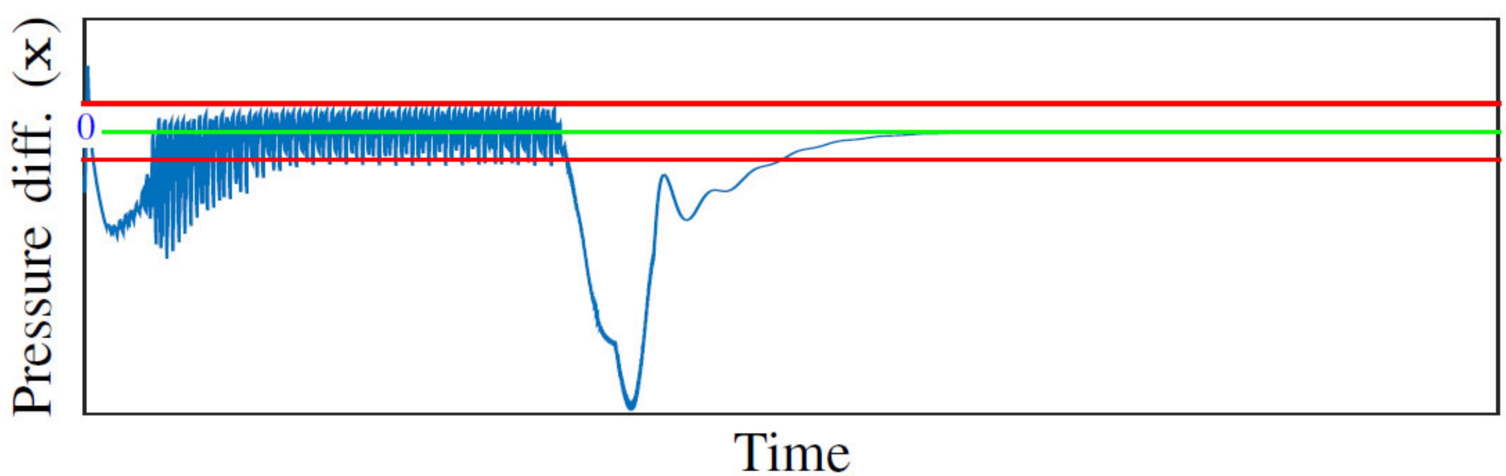
Bug fixed → Settling time successfully mined

OLD





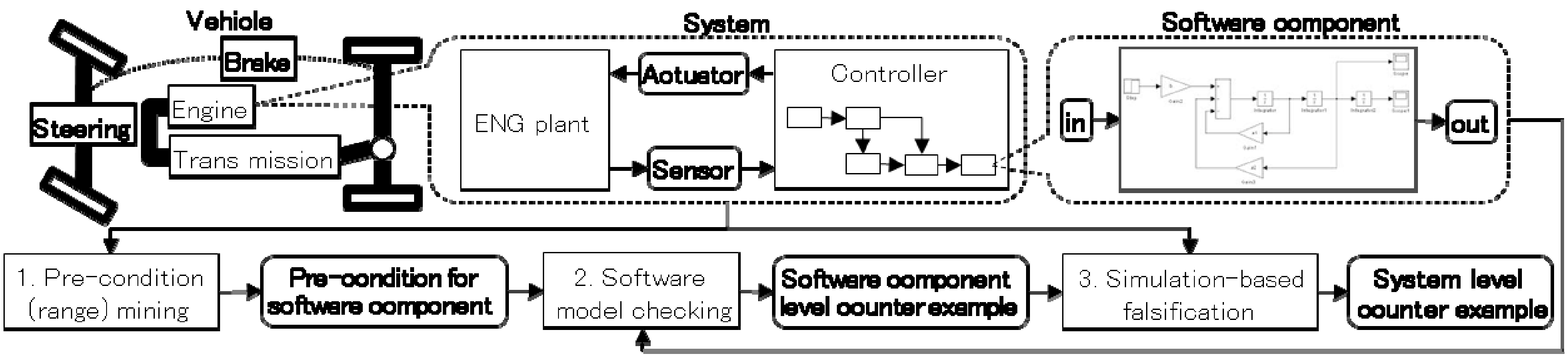
NEW



Industrial Case Studies with Toyota

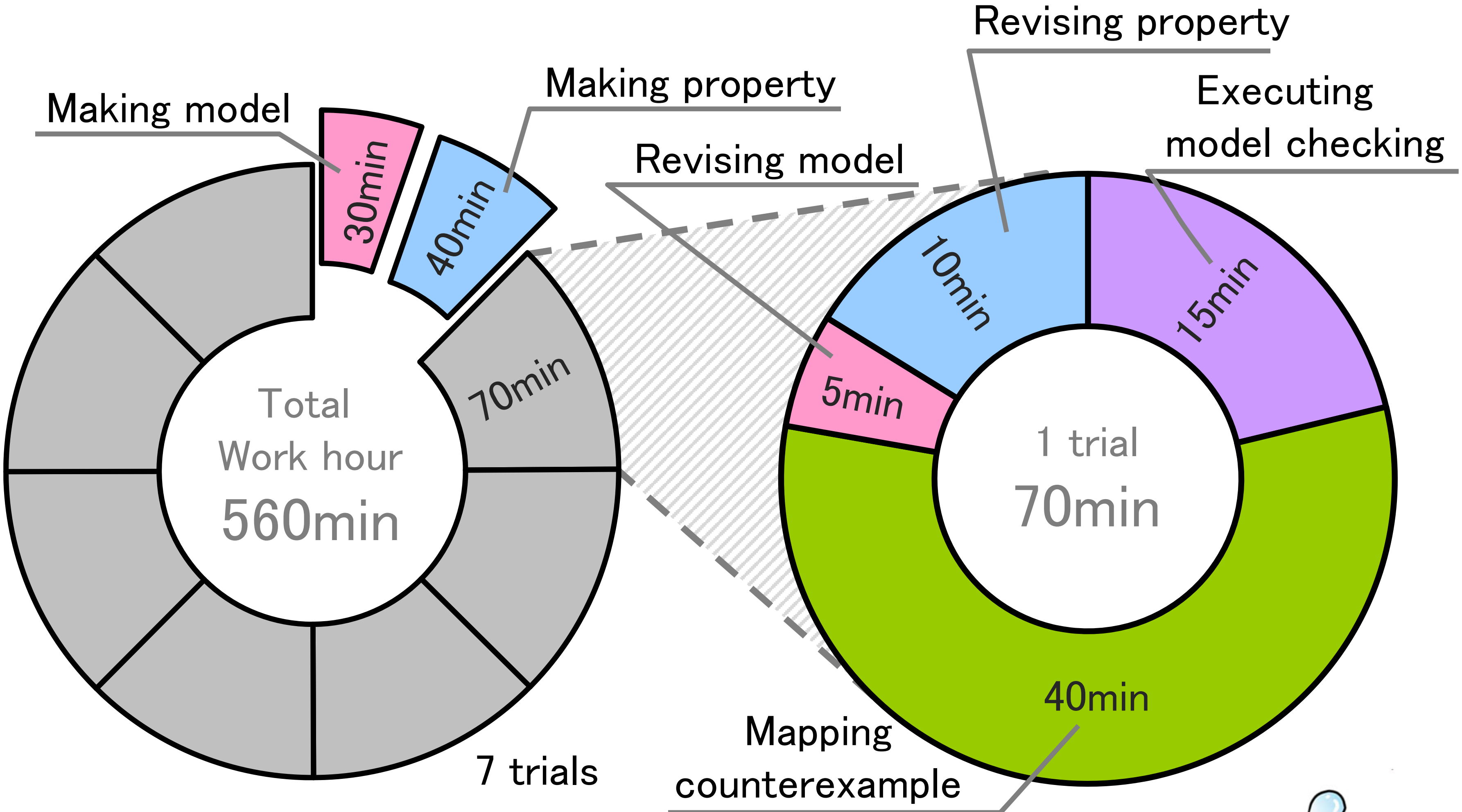
[Yamaguchi et al., FMCAD'16]

- Work with group @ Toyota Japan on *enabling software verification* by mining specifications on the closed-loop system
- Useful in a production setting:
  - Finds "issues" where previous methods fell short!
  - Reduced 70% of human effort



Toyota Unit's Experience with Model Checking

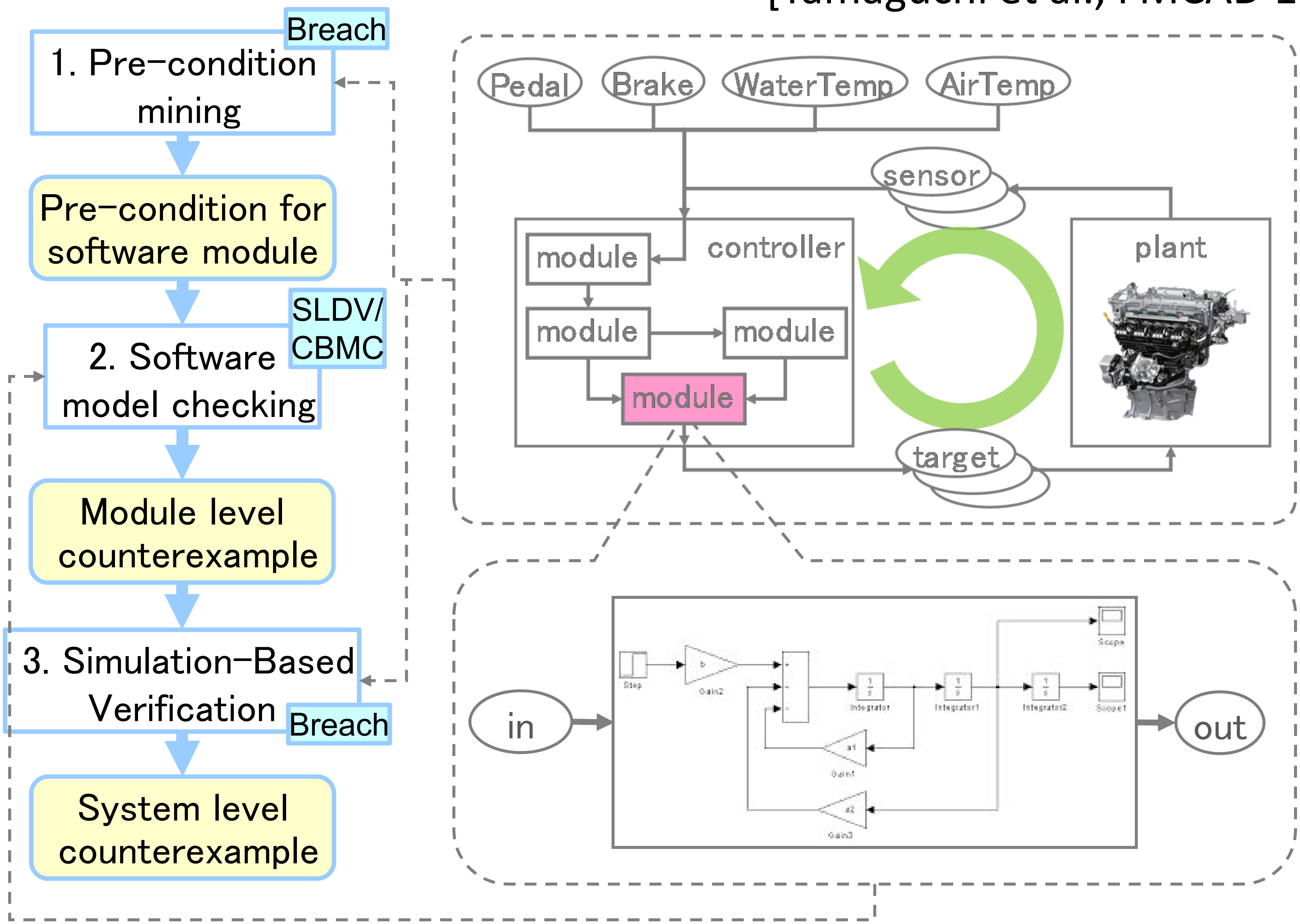
[Yamaguchi et al., FMCAD'16]



Making/revising property: 110 min
Mapping counterexample: 280 min for just 1 module

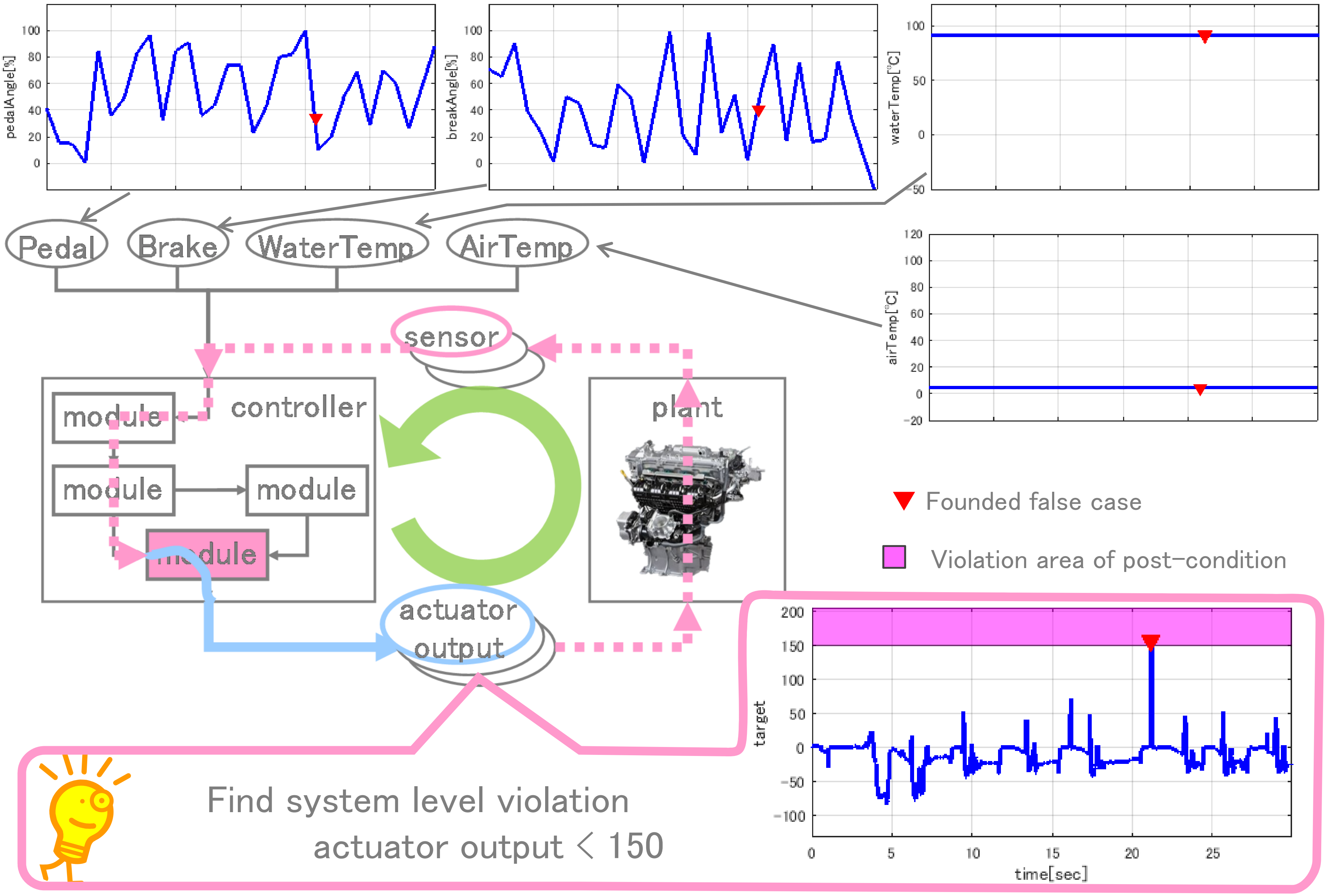
Overview of Methodology

[Yamaguchi et al., FMCAD’16]



Requirement Mining In Toyota Case Study

[Yamaguchi et al., FMCAD'16]



Summary: Specification & Verification of Industrial CPS

- Formal Specification often not available
- Inductive Synthesis from Models and Data can generate useful specifications *and* find bugs
  - Counterexample-Guided Inductive Synthesis (CEGIS)
- Simulation-based Temporal Logic Falsification is a scalable, industrially-applicable method
- Requirement Mining + Falsification can make Software Verification tools more effective

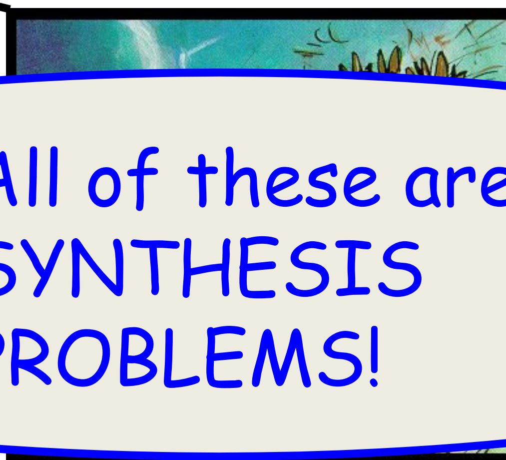
Frustrations of a “Formal Methodist”

How to strengthen the inductive invariant?

What are the right assertions/constraints?

Deductive Verificati

Simulation-Based ification



All of these are SYNTHESIS PROBLEMS!

Compositiona Verification

ogram Synthesis

How to decompose the specification into component-level contracts?

What’s the right environment model?

[Seshia, Proc. IEEE 2015]

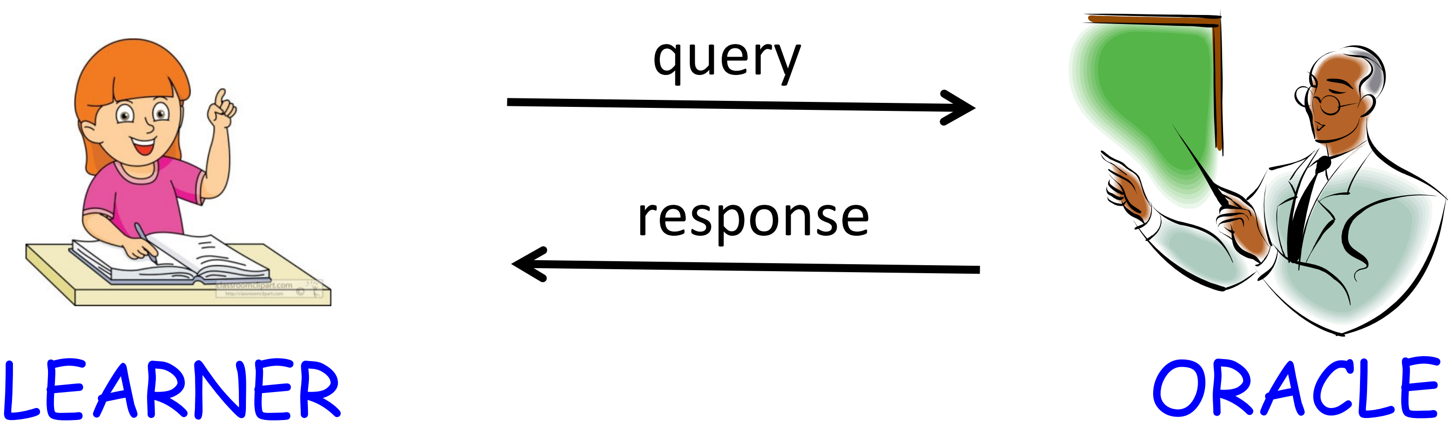
From CEGIS to Oracle-Guided Inductive Synthesis

Inductive Synthesis: Learning from Examples (ML)

Formal Inductive Synthesis: Learn from Examples *while satisfying a Formal Specification*

General Approach: **Oracle-Guided Learning**

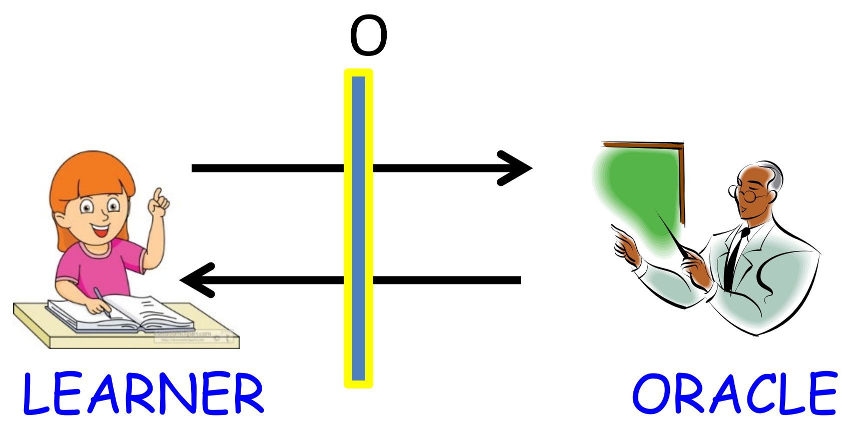
Combine Learner with Oracle (e.g., Verifier) that answers Learner's Queries



[Jha & Seshia, "A Theory of Formal Synthesis via Inductive Learning", 2015, Acta Informatica 2017.]

Formal Inductive Synthesis

- Given:
  - Class of Artifacts C
  - Formal specification ϕ
  - Domain of examples D
  - Oracle Interface O
    - Set of (query, response) types



- Find, by adhering to O, an f $\in$ C that satisfies ϕ
  - i.e. O defines protocol to access to D or ϕ

- To solve this: Design/Select *BOTH* Learner and Oracle

[Jha & Seshia, "A Theory of Formal Synthesis via Inductive Learning", 2015; 2017]

Common Oracle Query Types

(for trace property ϕ)

LEARNER

Positive Witness

$\longleftrightarrow$

$x \in \phi$, if one exists, else $\perp$

Negative Witness

$\longleftrightarrow$

$x \notin \phi$, if one exists, else $\perp$

Membership: Is $x \in \phi$?

$\longleftrightarrow$

Yes / No

Equivalence: Is $f = \phi$?

$\longleftrightarrow$

Yes / No + $x \in \phi \oplus f$

Subsumption/Subset: Is $f \subseteq \phi$?

$\longleftrightarrow$

Yes / No + $x \in f \setminus \phi$

Distinguishing Input: $f, X \subseteq f$

$\longleftrightarrow$

f' s.t. $f' \neq f \wedge X \subseteq f'$, if it exists;
o.w. $\perp$

ORACLE

Comparison*

[see also, Jha & Seshia, 2015]

Feature	Formal Inductive Synthesis	Machine Learning
Concept/Program Classes	Programmable, Complex	Fixed, “Simple”
Learning Algorithms	General-Purpose Solvers	Specialized
Learning Criteria	Exact, w/ Formal Spec	Approximate, w/ Cost Function
Oracle-Guidance	*Common (can select/design Oracle)*	*Rare (black-box oracles)*

* Between typical inductive synthesizer and machine learning algo

Inductive Synthesis for CPS: A Growing Area

- Synthesizing Specifications for CPS
  - Requirements for Closed-Loop (Automotive) Control Systems [HSCC'13]
  - Environment Assumptions for Robotics [MEMOCODE'11]
  - Temporal Logic Testers for Auto-Grading in Lab-Based Courses [EMSOFT'14]
  - …
- Controller Synthesis for CPS
  - Human-in-the-Loop Control – Semi-Autonomous Driving [TACAS'14]
  - Reactive Model Predictive Control (MPC) [HSCC'15]
  - Applications to Semi-Autonomous Driving [HSCC'16, RSS'16, IROS'16]
  - …

Query Types for Counterexample-Guided Inductive Synthesis (CEGIS)

Positive Witness

$x \in \phi$, if one exists, else $\perp$

Counterexample to f?

Yes + counterexample x / $\perp$

Finite memory vs
Infinite memory

Type of counter-example given

Concept class: Any set of recursive languages

Some Initial Theoretical Results on CEGIS

[Jha & Seshia, 2015; Jha, Seshia, Zhu, 2016]

- Finite-sized Concept/Program Classes:
  - Teaching Dimension [Goldman & Kearns '90] is a lower bound on query complexity
  - TD of n-dimensional rectangles is O(n), of n-dimensional octagons is $O(n^2)$
  - Relevance for Invariant Inference
- Infinite-sized Concept Classes:
  - Analyze CEGIS variants for "learning in the limit" [Gold, 1967]
  - Minimizing counterexamples does not change learnability
  - Getting "positive-bounded" counterexamples can enable one to learn more than standard CEGIS when learner buffer size is finite
- Much more to be investigated!!!

Summary

- Formal Inductive Synthesis
  - Counterexample-guided inductive synthesis (CEGIS)
  - General framework for solution methods: Oracle-Guided Inductive Synthesis (OGIS)
  - Theoretical analysis
- Industrial Case Study: Synthesis of Specifications
- Lots of potential for future work!

S. A. Seshia, “Combining Induction, Deduction, and Structure for Verification and Synthesis”, Proceedings of the IEEE, November 2015.