Formal Methods meets Machine Learning: Explorations in Cyber-Physical System Design

Sanjit A. Seshia

Professor

UC Berkeley

Joint work with:

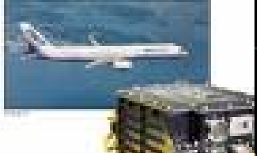
Jyo Deshmukh, Tommaso Dreossi, Alex Donze, Dorsa Sadigh, Susmit Jha, Xiaoqing Jin, Tomoyuki Kaga, Tomoya Yamaguchi, S. Shankar Sastry

Stanford University

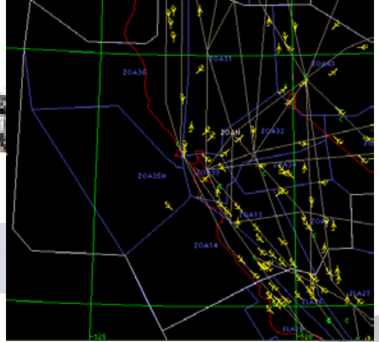
December 4, 2017

Cyber-Physical Systems (CPS):

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



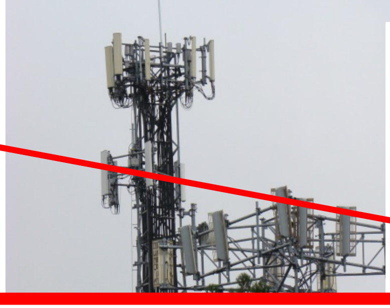
Avionics

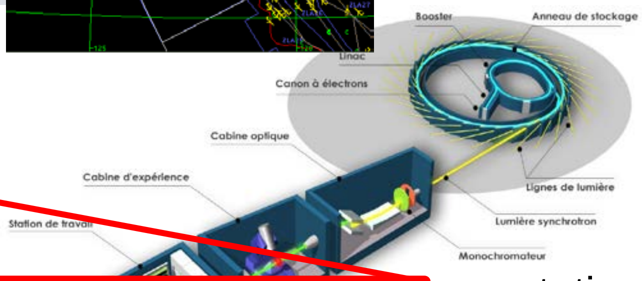


Transportation (Air traffic control at SFO)

Building Systems

Telecommunications

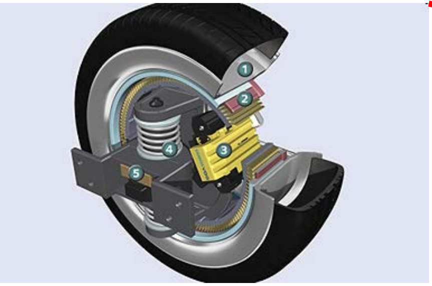




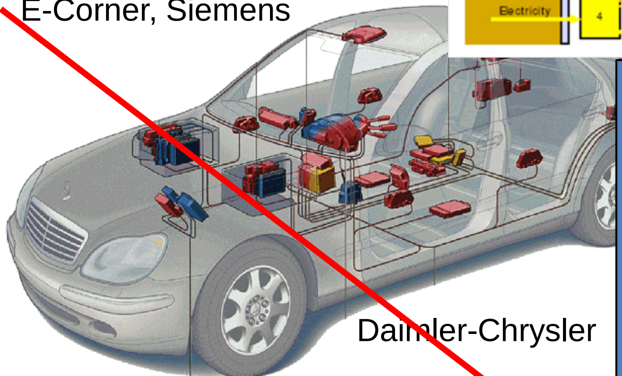
umentation

ynchrotron)

Automotive



E-Corner, Siemens

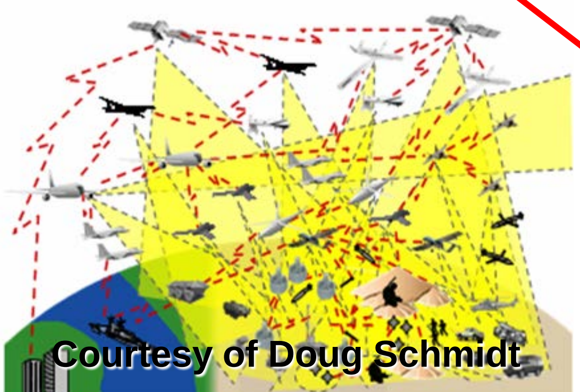


Daimler-Chrysler

Automotive domain representative of key societal challenges:

- Smart Cities / Infrastructure
- Energy Efficiency
- Climate Change
- Humans and Automation
- …

Military systems:



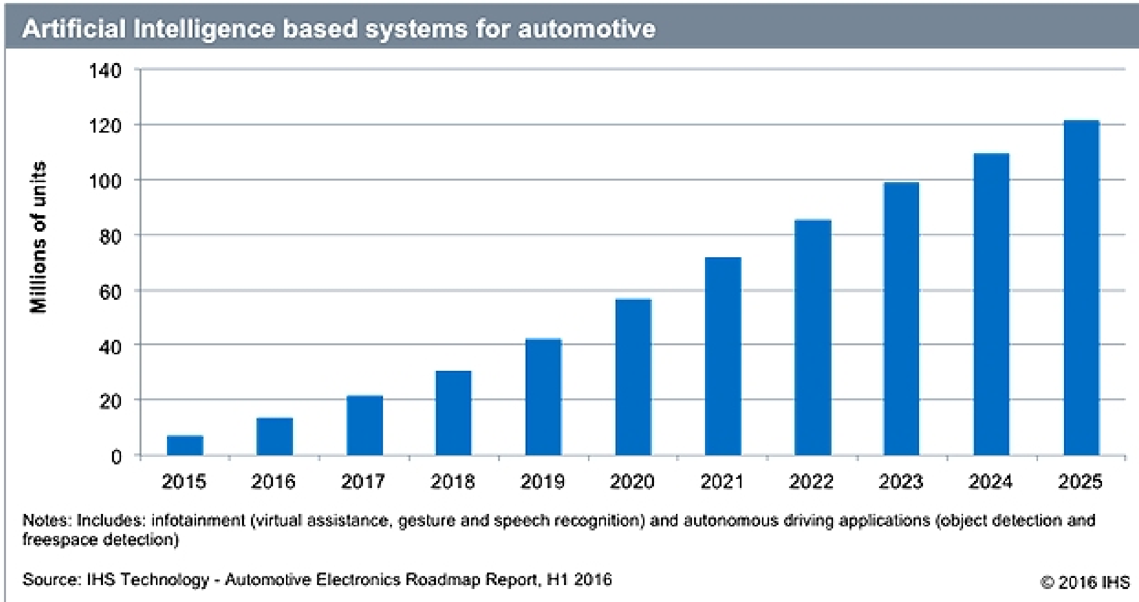
Courtesy of Doug Schmidt

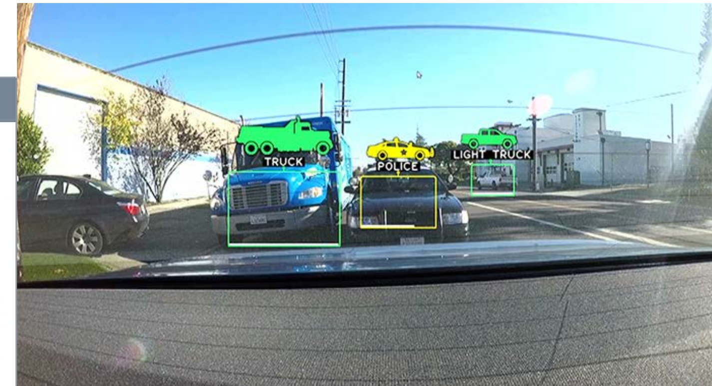
Courtesy of General Electric

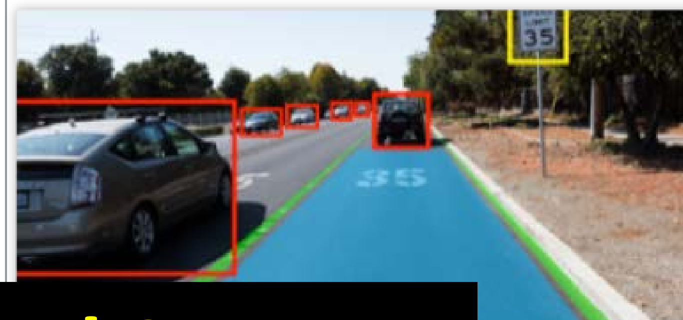
Courtesy of Kuka Robotics Corp.

[E. A. Lee]

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



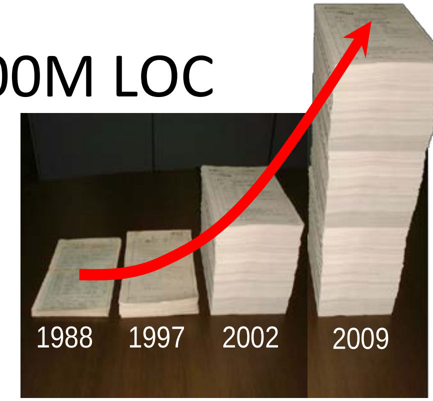




Many Safety-Critical Systems

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 autopilot incidents: reasons still unclear

…

Formal Methods to the Rescue?

- Industry need for higher assurance → Increasing interest in Formal Methods
- 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)
- 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

Outline

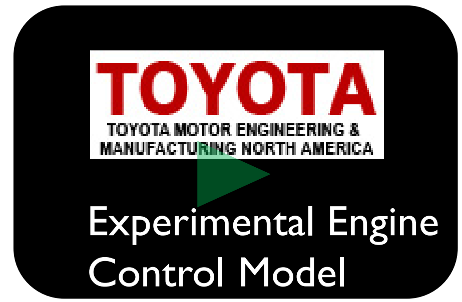
- Synthesizing Requirements for Closed-Loop Control Systems
  - Industrial Tech Transfer to Toyota
- Falsification of Deep Learning based CPS
  - Context: autonomous driving
- Conclusion

Mining Requirements for Closed-Loop Control Systems

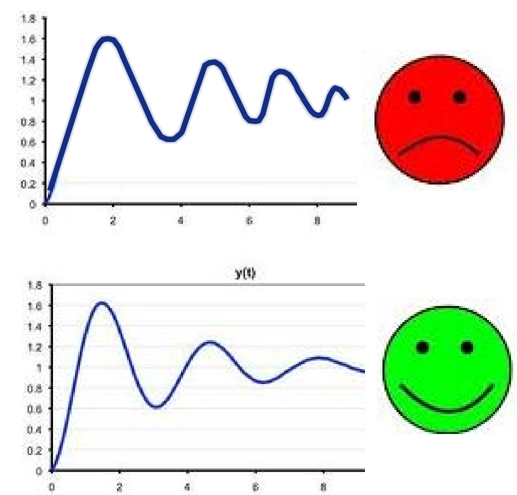
[Jin, Donze, Deshmukh, Seshia, HSCC 2013, TCAD 2015;
Yamaguchi et al. FMCAD 2016]

Challenges for Verification of 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: Requirements Mining

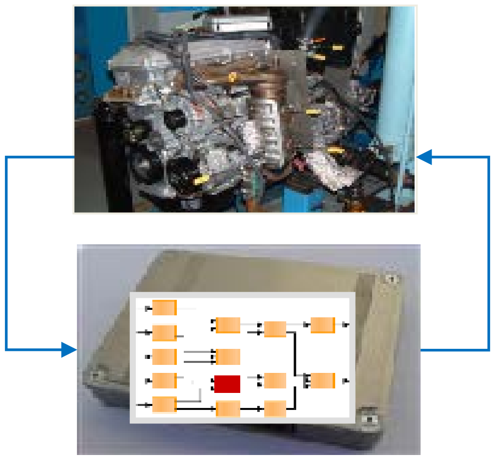
Value added by mining:

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

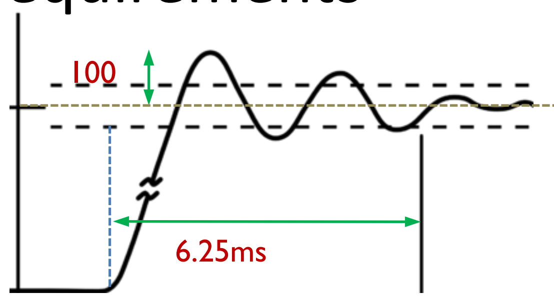
It's working, but I don't understand why!

Designer's View of Our Solution

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

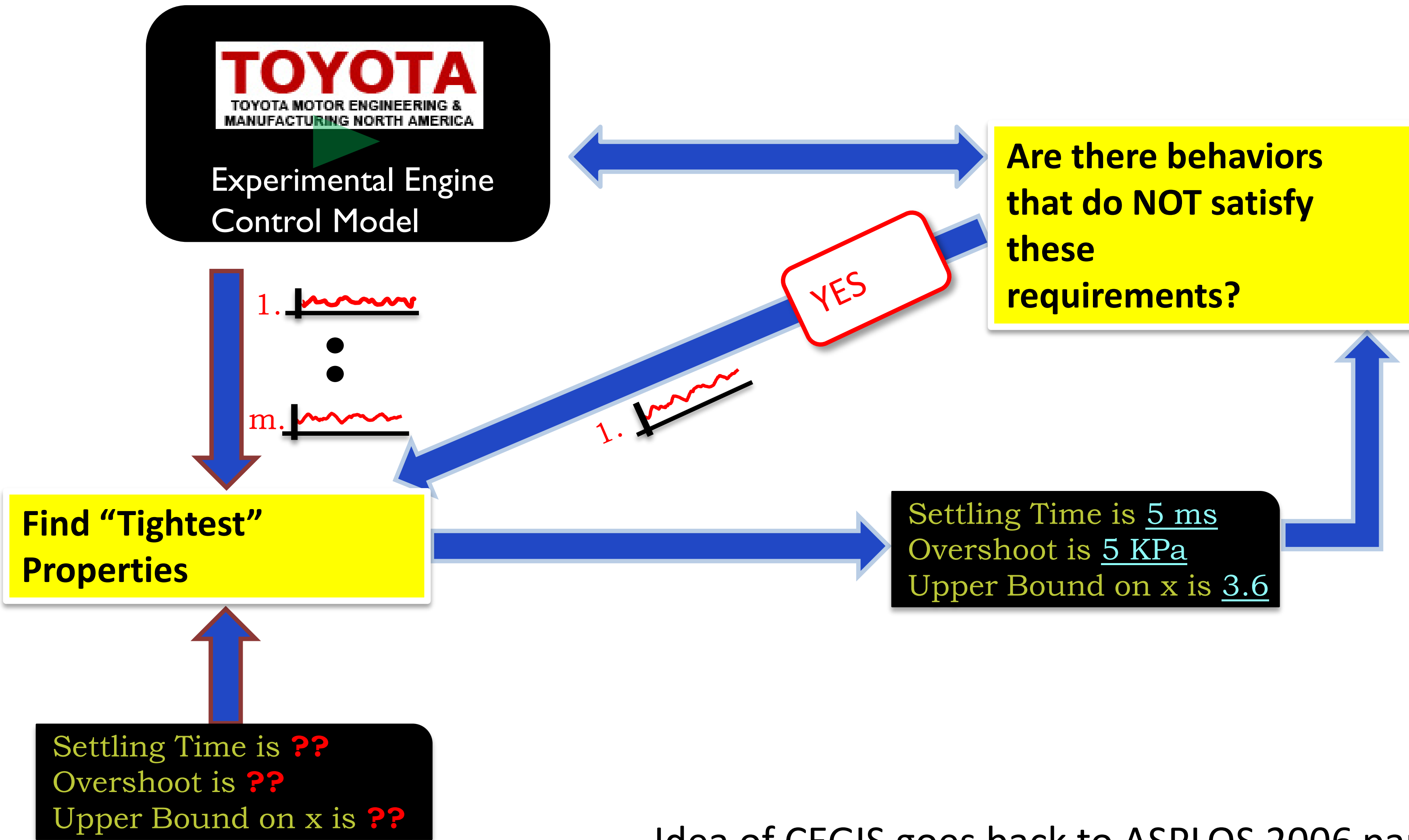


- Designer reviews mined requirements
  - "Settling time is 6.25 ms"
  - "Overshoot is 100 units"
  - Expressed in Signal Temporal Logic [Maler & Nickovic, '04]



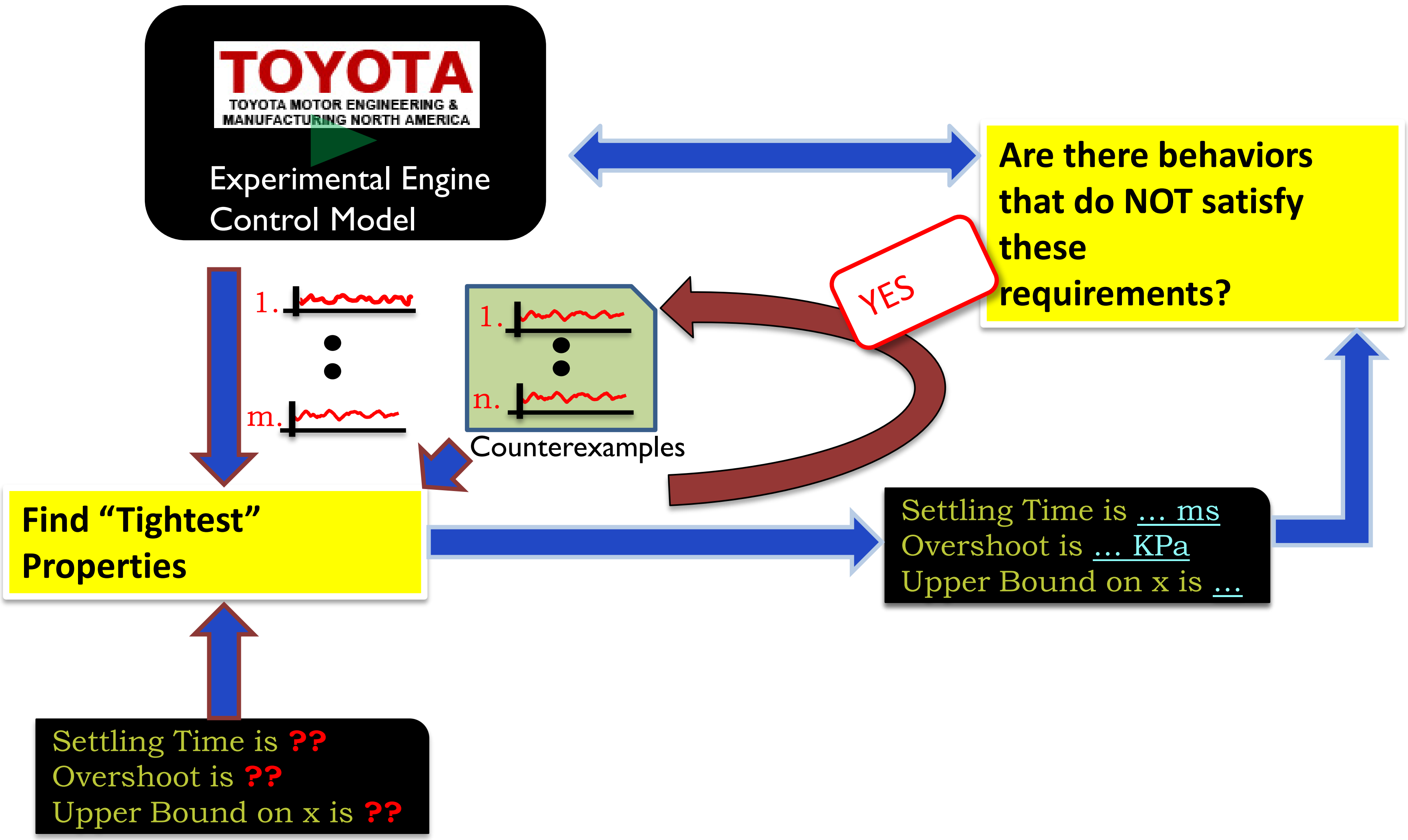
CounterExample Guided Inductive Synthesis (CEGIS)

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

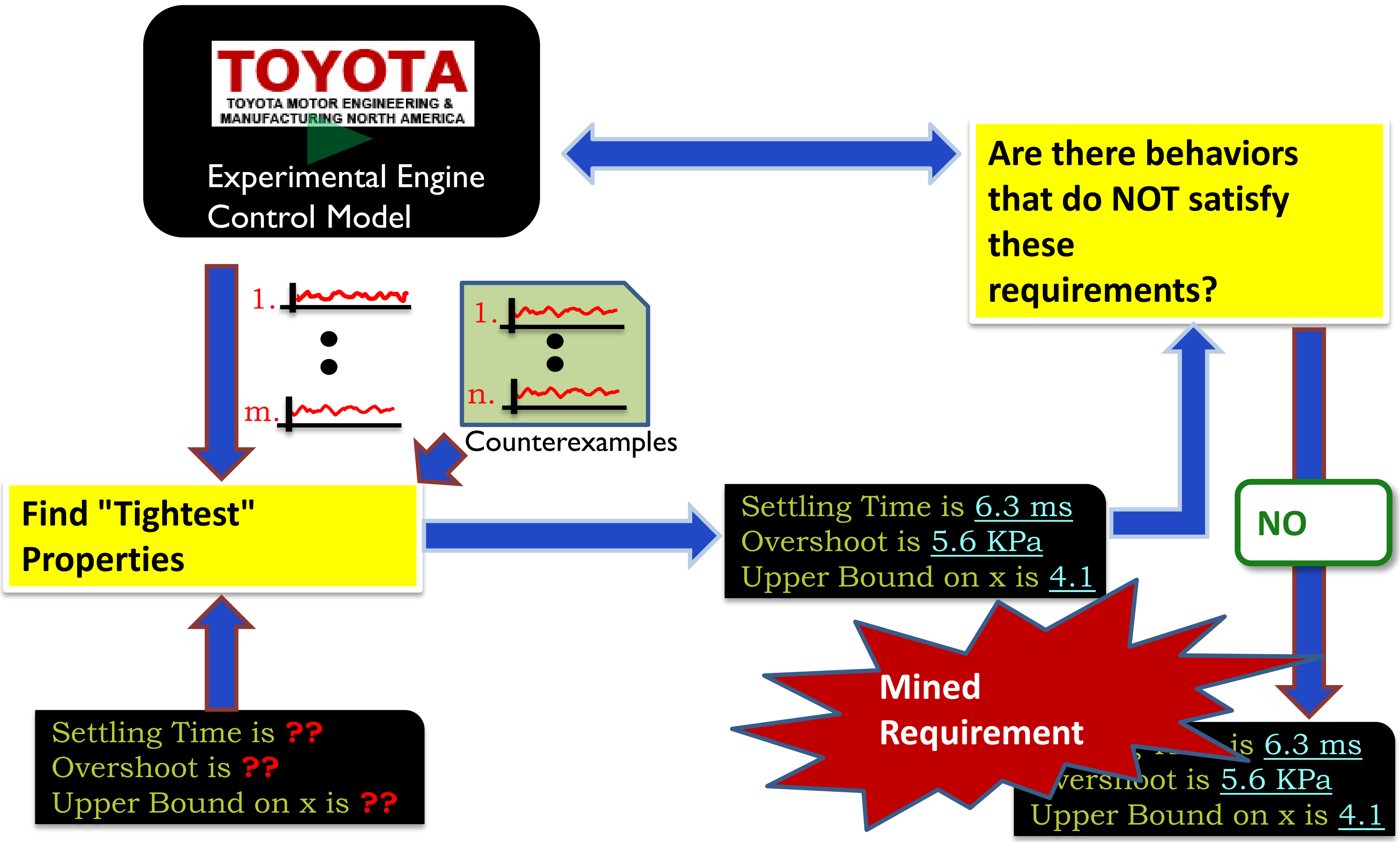


Idea of CEGIS goes back to ASPLOS 2006 paper by Solar-Lezama et al.

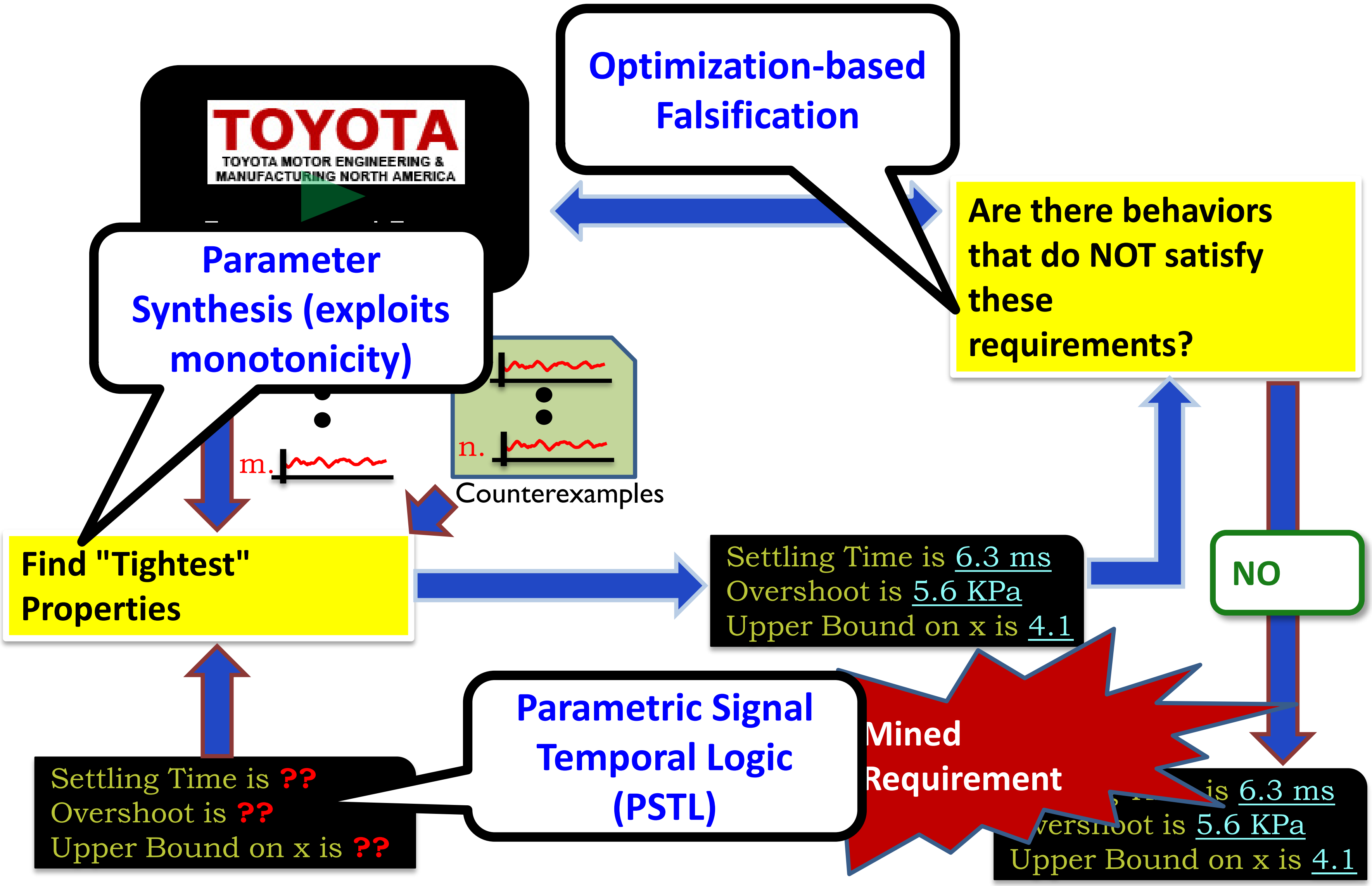
CounterExample Guided Inductive Synthesis (CEGIS)



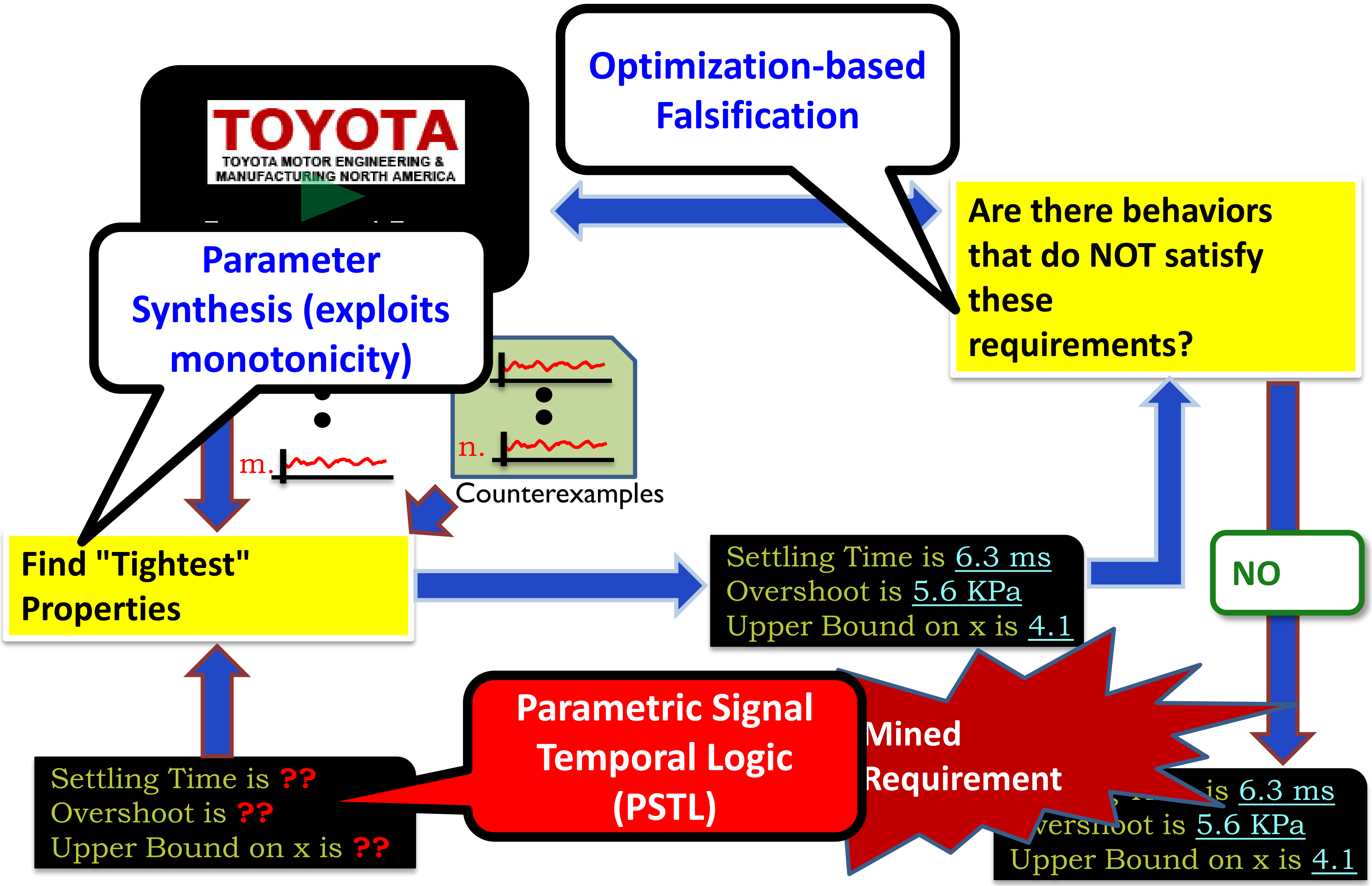
CounterExample Guided Inductive Synthesis



CounterExample Guided Inductive Synthesis



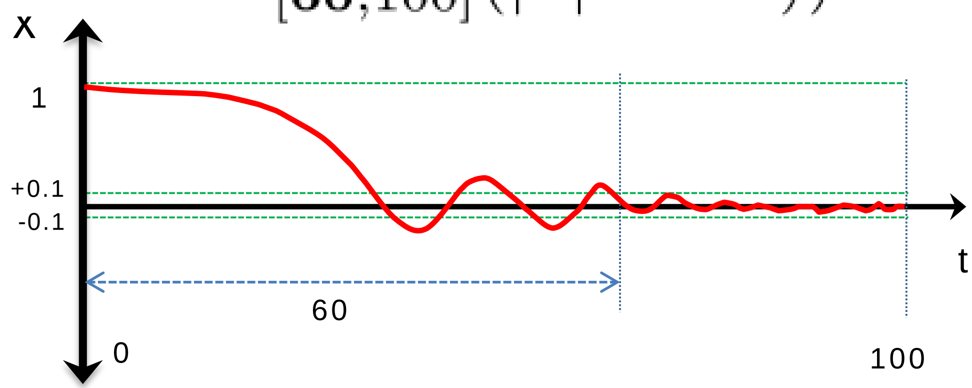
CounterExample Guided Inductive Synthesis



Signal Temporal Logic (STL)

- 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"*

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



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:

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

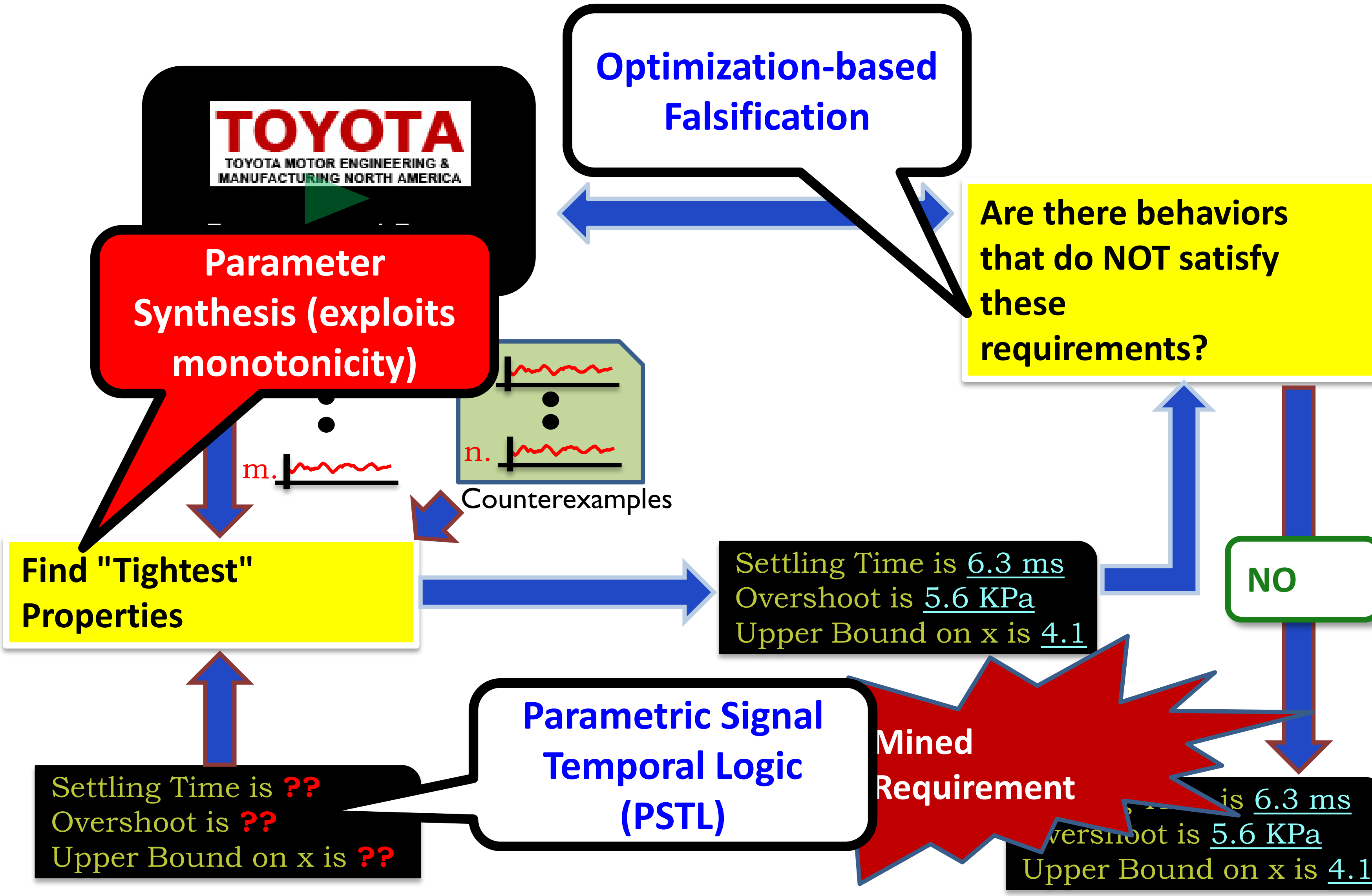
Between some time τ and 10 seconds, x remains greater than some value π

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

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

---

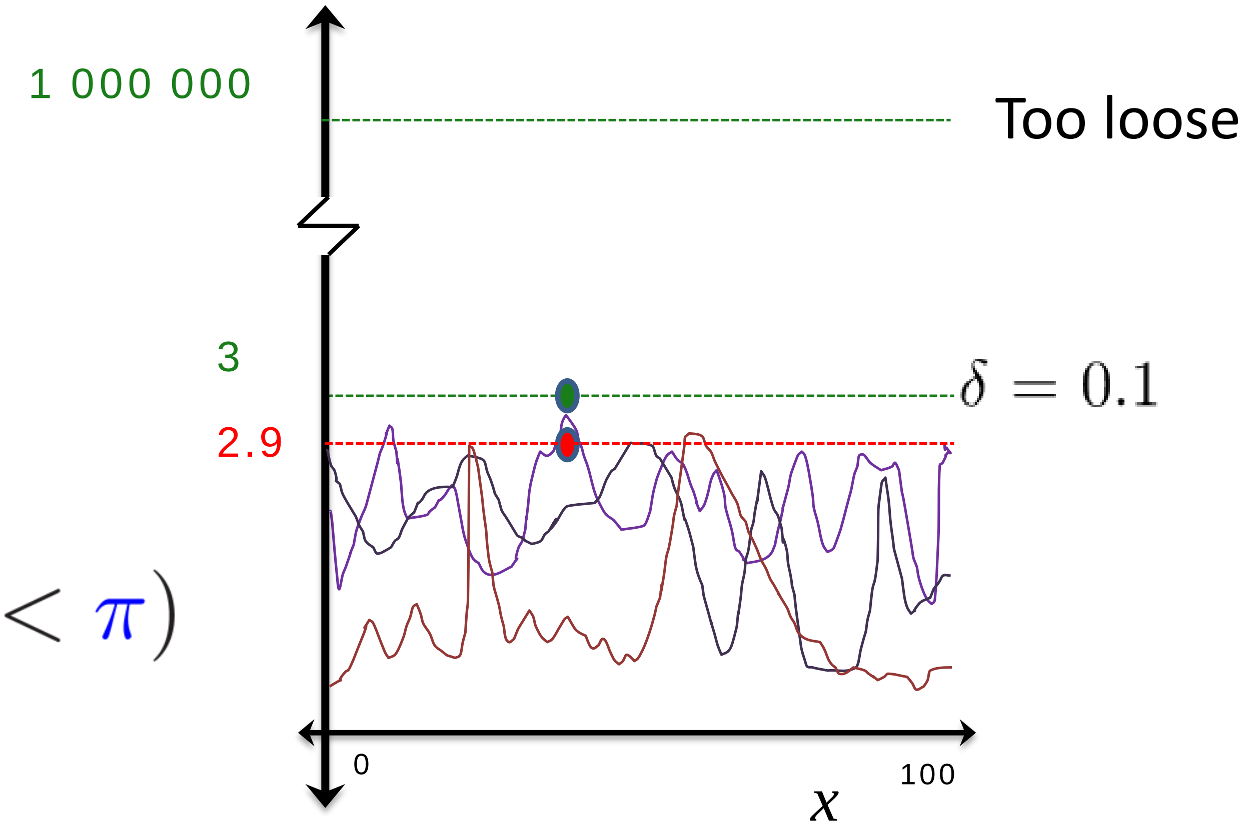
CounterExample Guided Inductive Synthesis



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

Find "Tightest" Properties

$$\Box(x < \pi)$$



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

Parameter Synthesis

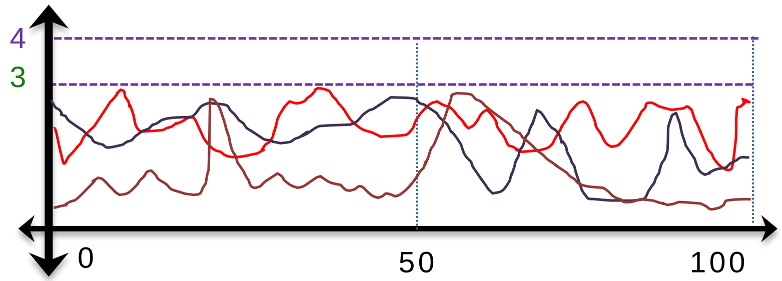
Find "Tightest" Properties

- 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)

Satisfaction Monotonicity

Find "Tightest" Properties

- 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

Find "Tightest" Properties

Deciding Satisfaction Monotonicity

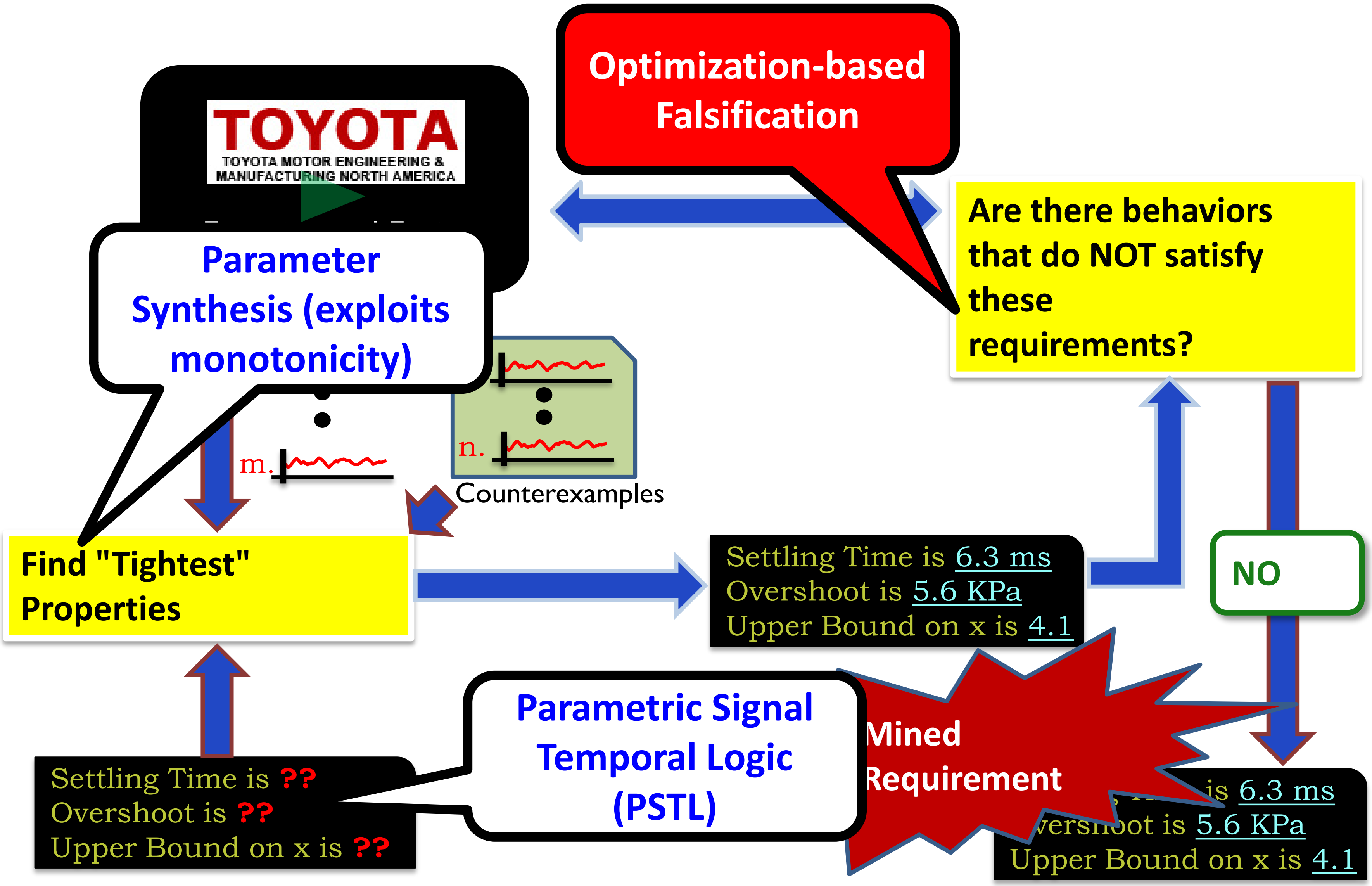
- Need to decide whether:

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

- Theorem: Deciding monotonicity of a PSTL formula is undecidable

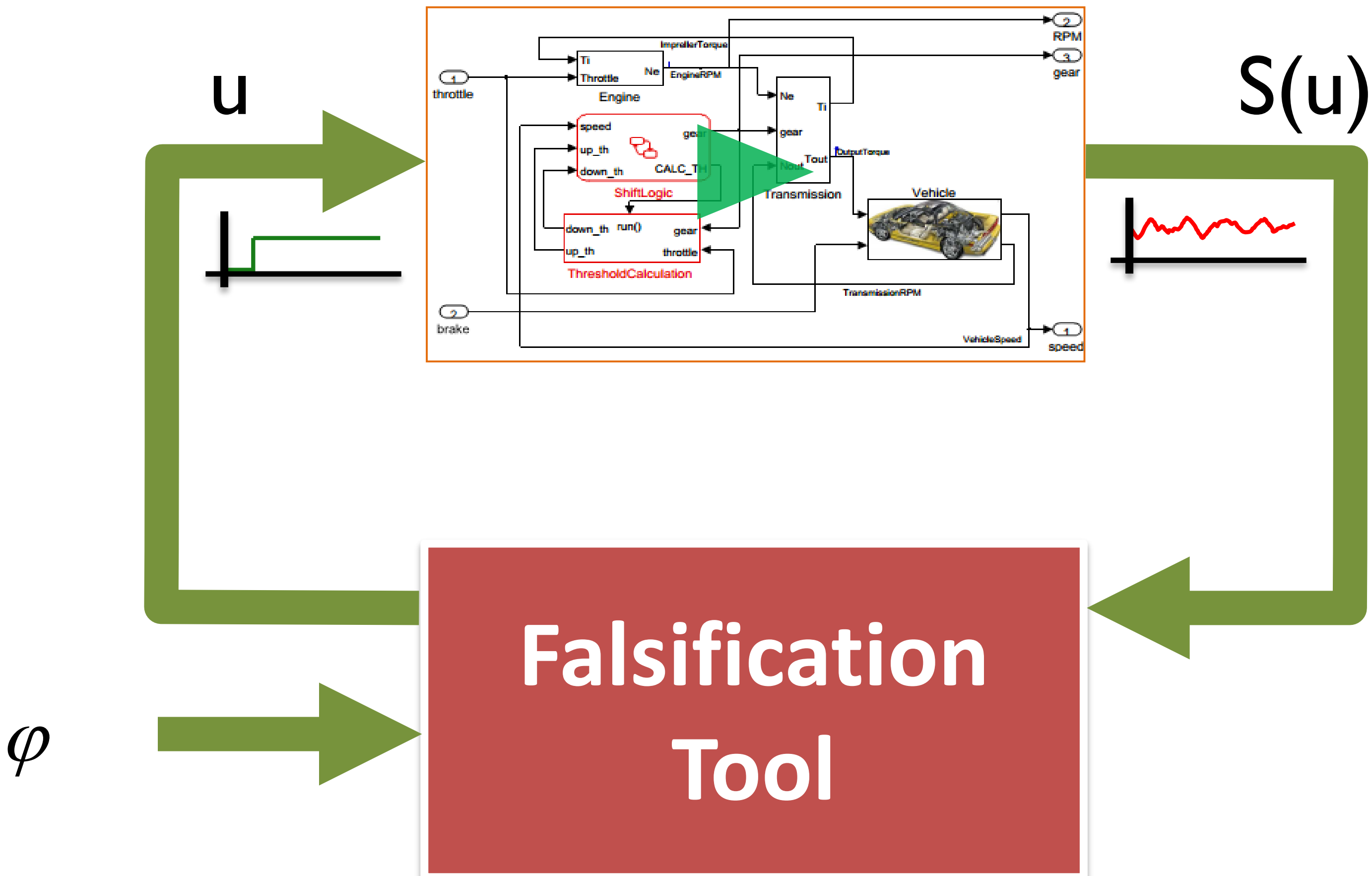
- Use an encoding to satisfiability modulo theories (SMT) solving
  - Quantified formulas involving uninterpreted functions, and arithmetic over reals → linear arithmetic if PSTL predicates are linear
  - Solved easily with Z3

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]

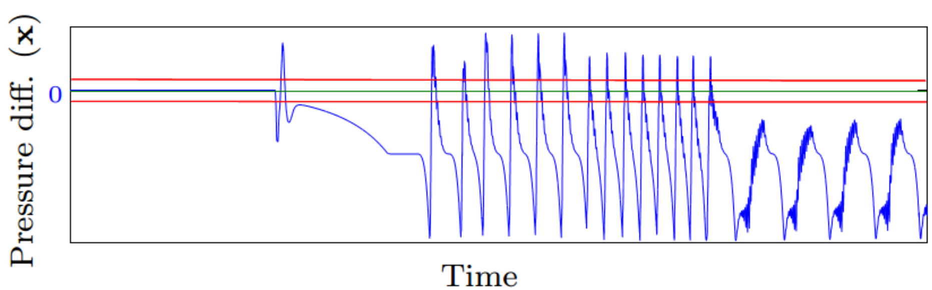
- Defined Templates for Common Requirements in Automotive Control – *all monotonic PSTL!!*
  – Dwell-Time requirements
  – Timed/Untimed Safety properties
  – Timed Inevitability (bounded liveness)
  – Input Profiles: assumptions on shape of input signals
  – Control-theoretic requirements on output 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

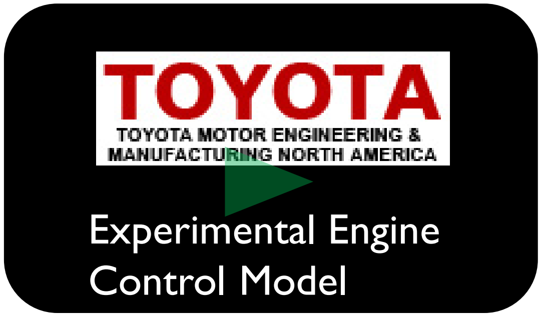
[Jin, Donze, Deshmukh, Seshia, HSCC 2013]

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)



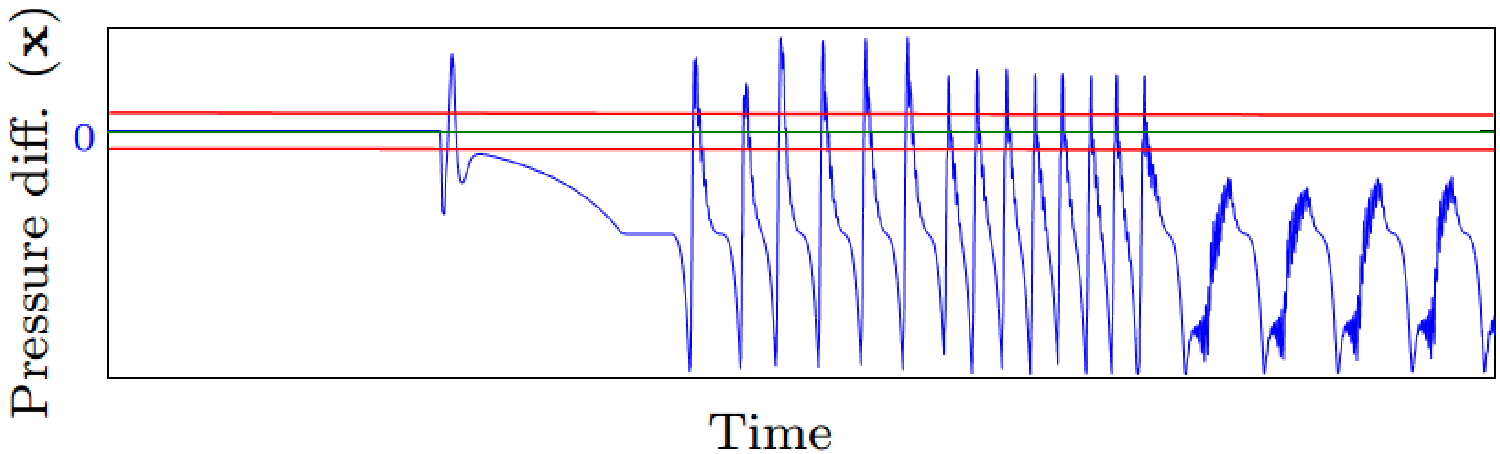
Mining can expose deep bugs



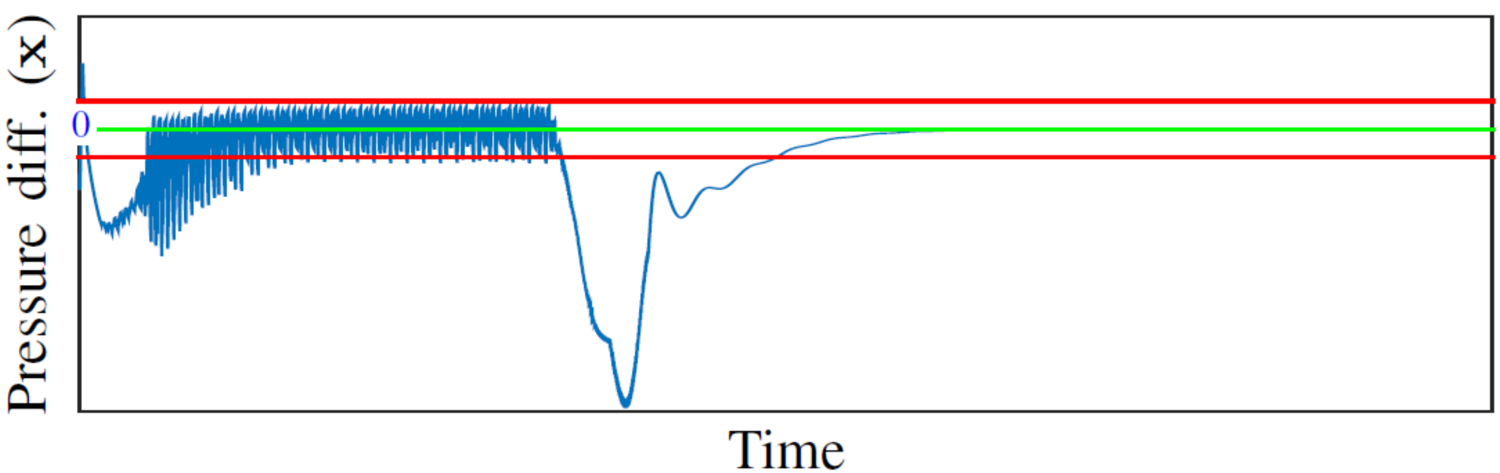
- Uncovered a tricky bug
  - Discussion with control designer revealed it to be a real bug
  - Root cause identified as wrong value in a look-up table, bug was fixed
- Why mining could be useful for bug-finding:
  - Can uncover subtle relations that should not hold
  - Looking for bugs ≈ Mine for negation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

Bug fixed → Settling time successfully mined

OLD



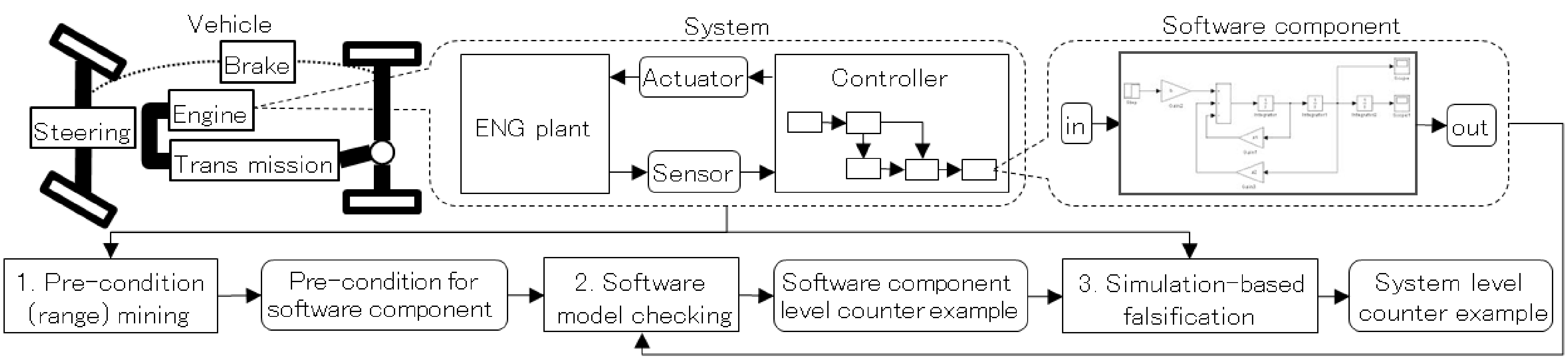
NEW



Industrial Case Studies with Toyota

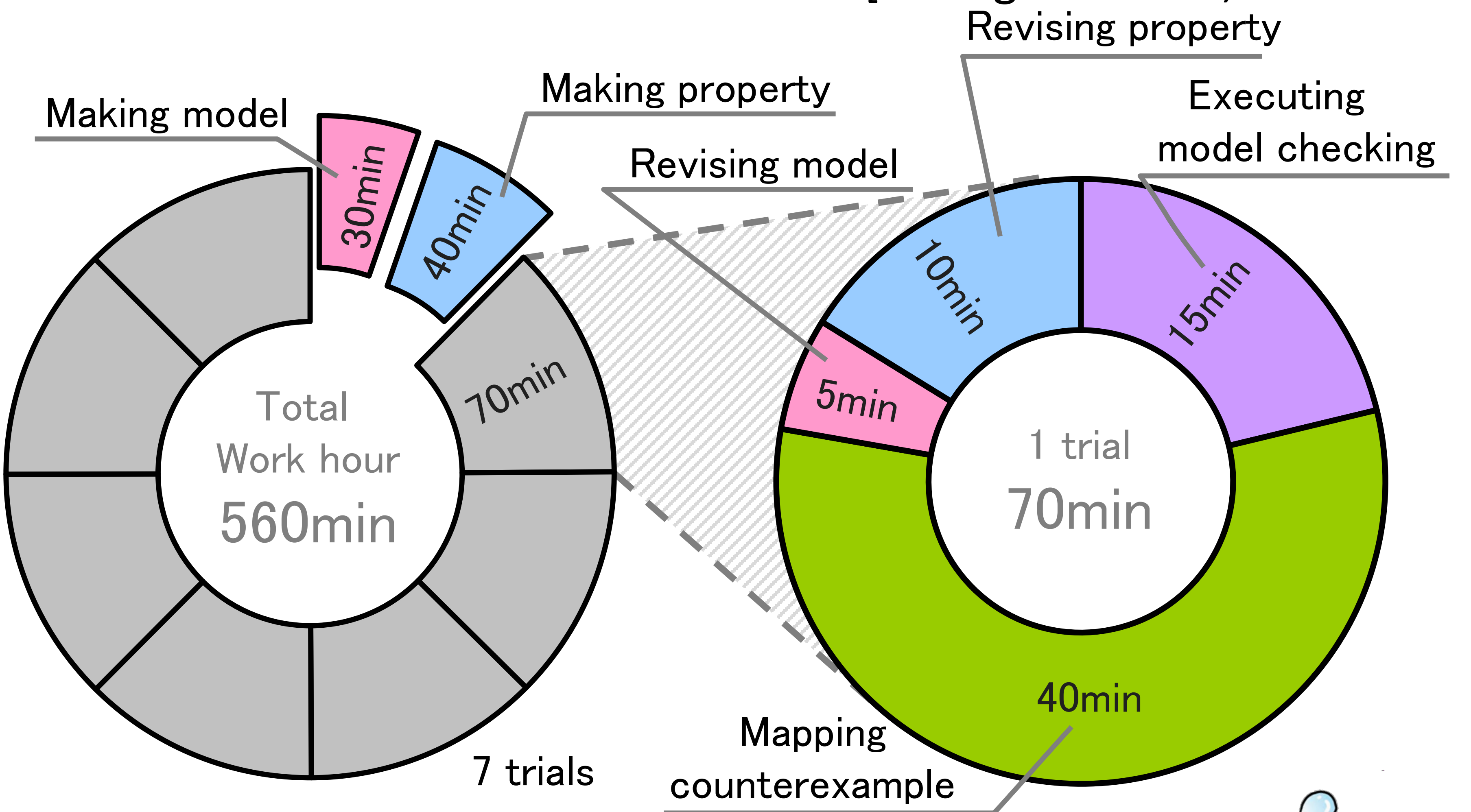
[Yamaguchi et al., FMCAD'16]

- 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

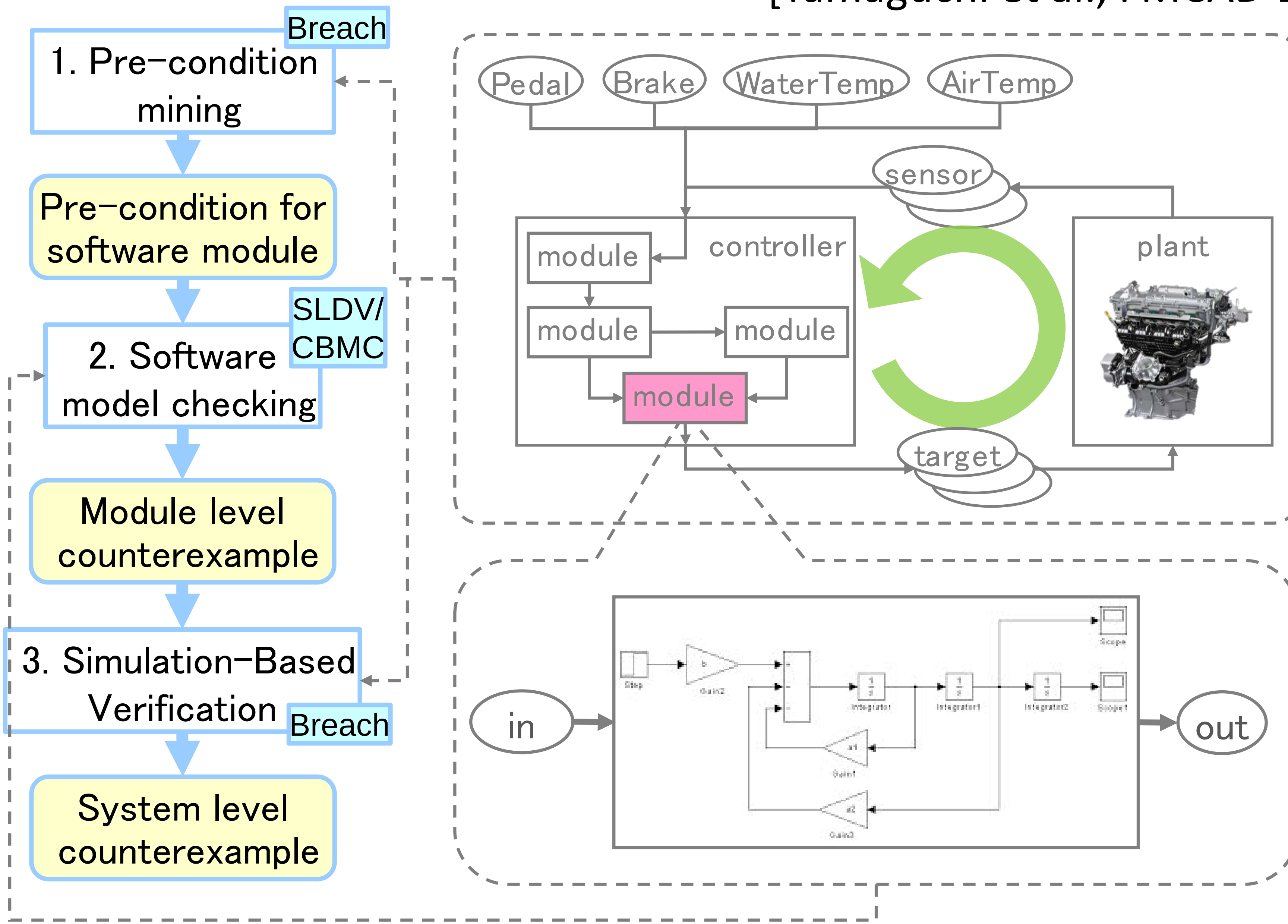
[Yamaguchi et al., FMCAD'16]



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

Overview of Methodology

[Yamaguchi et al., FMCAD'16]



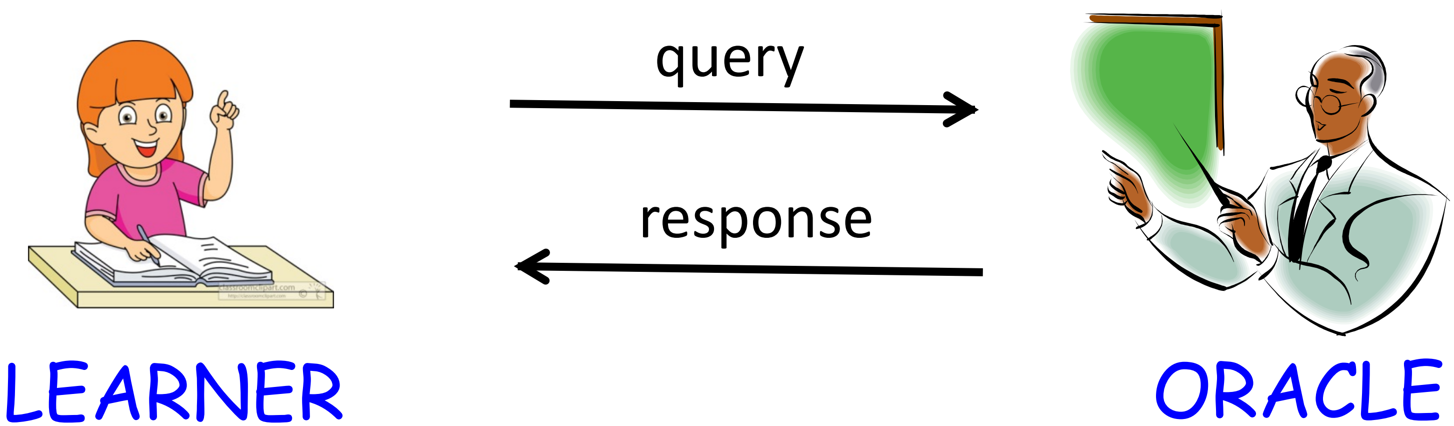
From CEGIS to Oracle-Guided Inductive Synthesis

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



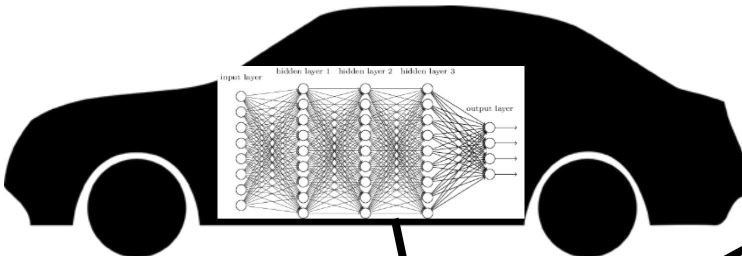
[Jha & Seshia, "A Theory of Formal Synthesis via Inductive Learning", 2015, Acta Informatica 2017.]

Falsification of Cyber-Physical Systems with Machine Learning Components

T. Dreossi, A. Donze, and S. A. Seshia. *Compositional Falsification of Cyber-Physical Systems with Machine Learning Components*, In NASA Formal Methods Symposium, May 2017.

Challenges for Verified AI

S. A. Seshia, D. Sadigh, S. S. Sastry. *Towards Verified Artificial Intelligence*. July 2016. https://arxiv.org/abs/1606.08514.



System **S**

Environment **E**

Specification **φ**

Does S || E satisfy φ?

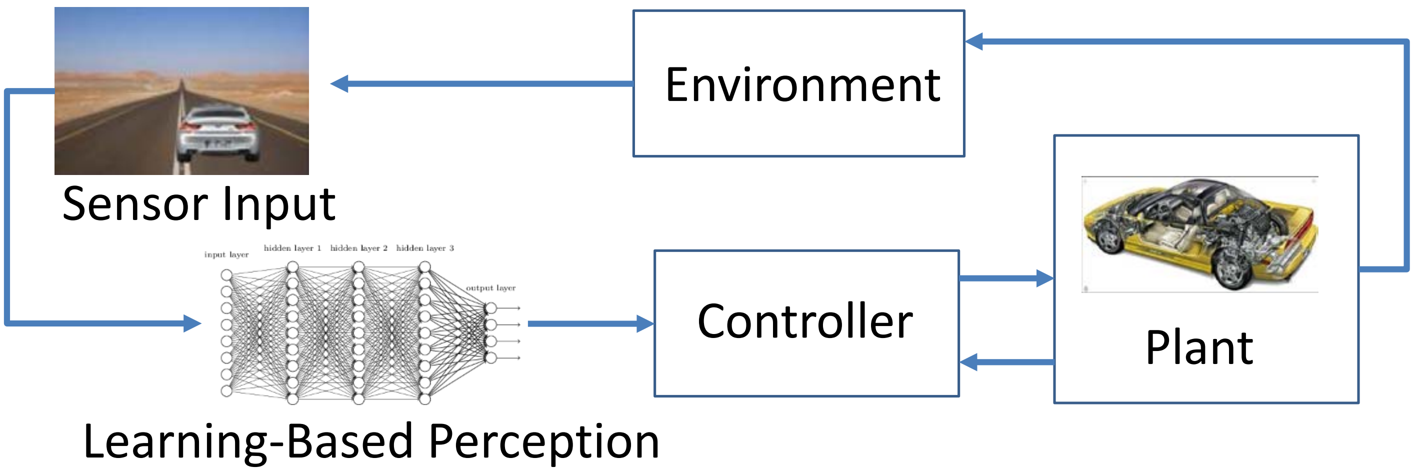
YES [+ proof]

NO [+ counterexample]

Design Correct-by-Construction instead?

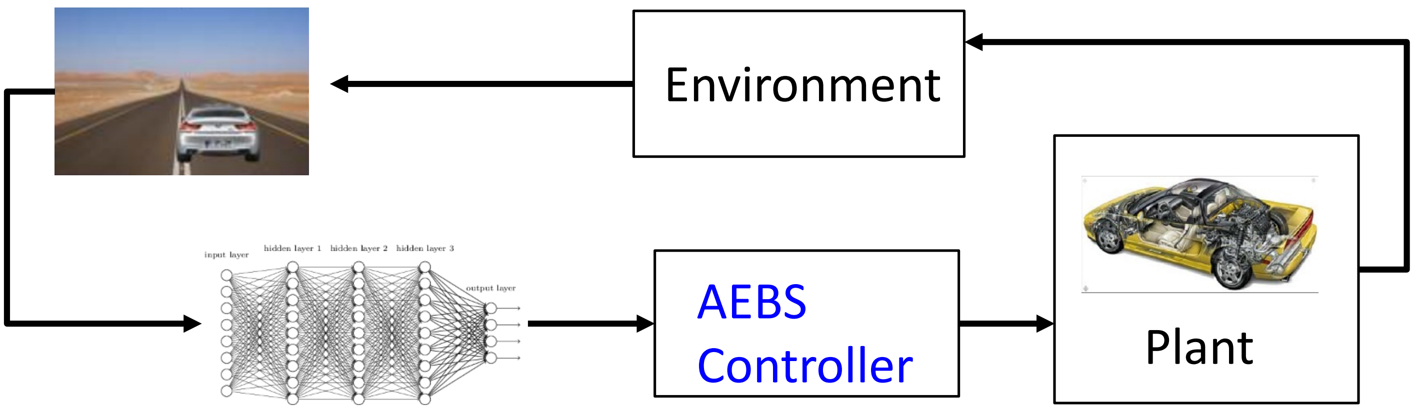
Counterexamples, etc. from Rich Signal Spaces?

Problem: Verify Automotive System (CPS) that uses ML-based Perception



- Focus:
  - Falsification: finding scenarios that violate safety properties
  - Test (Data) Generation: generate “interesting” data for training / testing → improve accuracy
  - Deep Neural Networks, given the increasing interest and use in the automotive context.

Automatic Emergency Braking System (AEBS)



Deep Learning-Based Object Detection

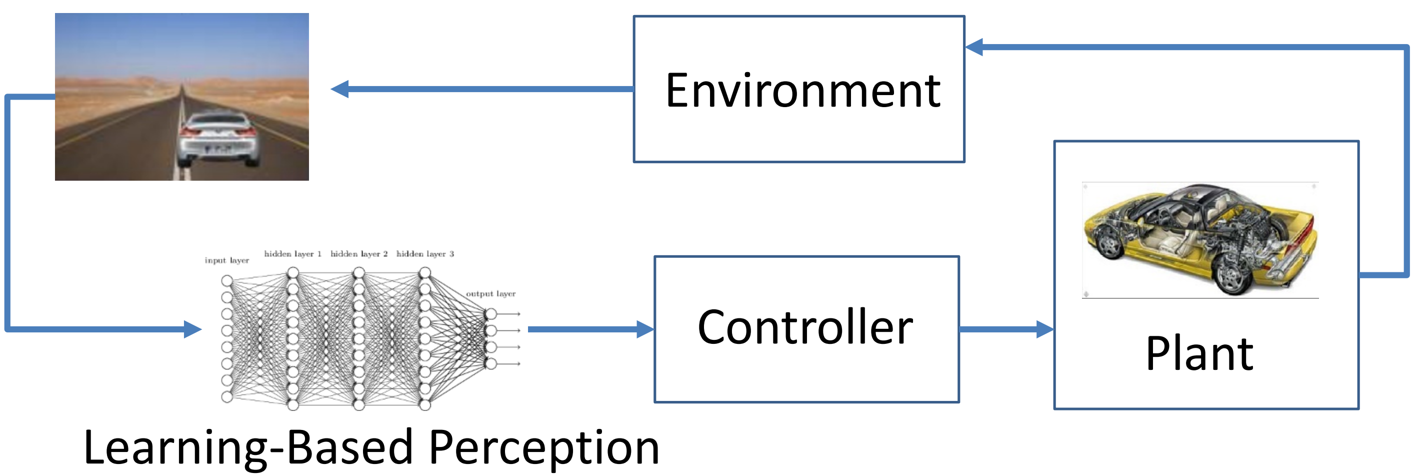
- Goal: Brake when an obstacle is near, to maintain a minimum safety distance
  - Controller, Plant, Env models in Matlab/Simulink
- Object detection/classification system based on deep neural networks
  - Inception-v3, AlexNet, … trained on ImageNet
  - more recent: squeezeDet, Yolo, … trained on KITTI

Our Approach: Use a System-Level Specification

- ✗ "Verify the Deep Neural Network Object Detector"
- ✓ "Verify the System containing the Deep Neural Network"

Formally Specify the *End-to-End Behavior* of the System

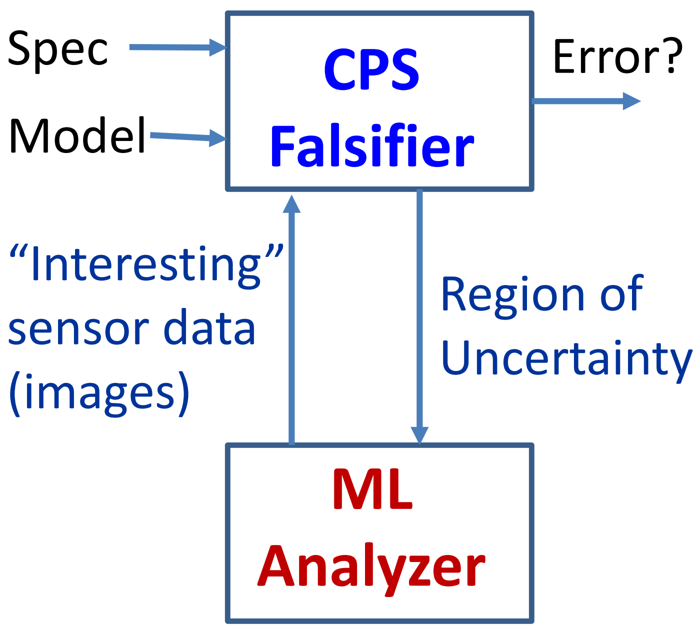
STL Formula: **G** (*dist*(ego vehicle, env object) > Δ)



Approach: Simulation-Based Falsification

- *Challenge:* Very High Dimensionality of Input Space!
- Standard solution: Use *Compositional (Modular)* Verification
- However: *no formal spec*. for neural network component!
- Compositional Verification *without* Compositional Specification?!!

Our Approach: Combine Temporal Logic CPS Falsifier with ML Analyzer

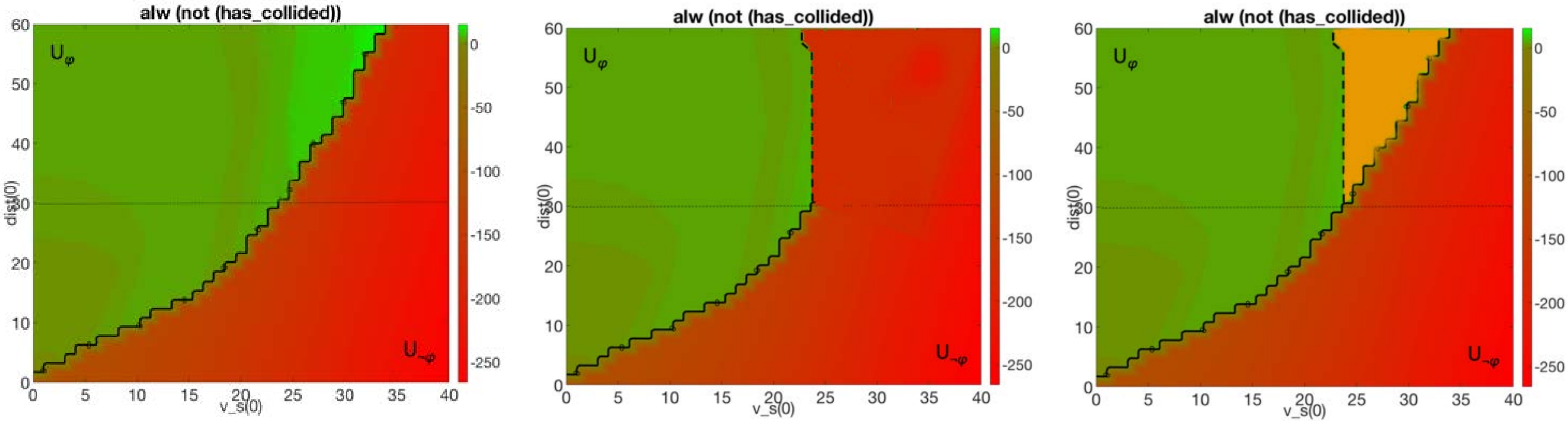


- CPS Falsifier uses abstraction of ML component
  - Optimistic analysis: assume ML classifier is always correct
  - Pessimistic analysis: assume classifier is always wrong
- Difference is the region of uncertainty where output of the ML component "matters"

Compositional:

CPS Falsifier and ML Analyzer can be designed and run independently (& communicate)!

Identifying Region of Uncertainty (*ROU*) for Automatic Emergency Braking System

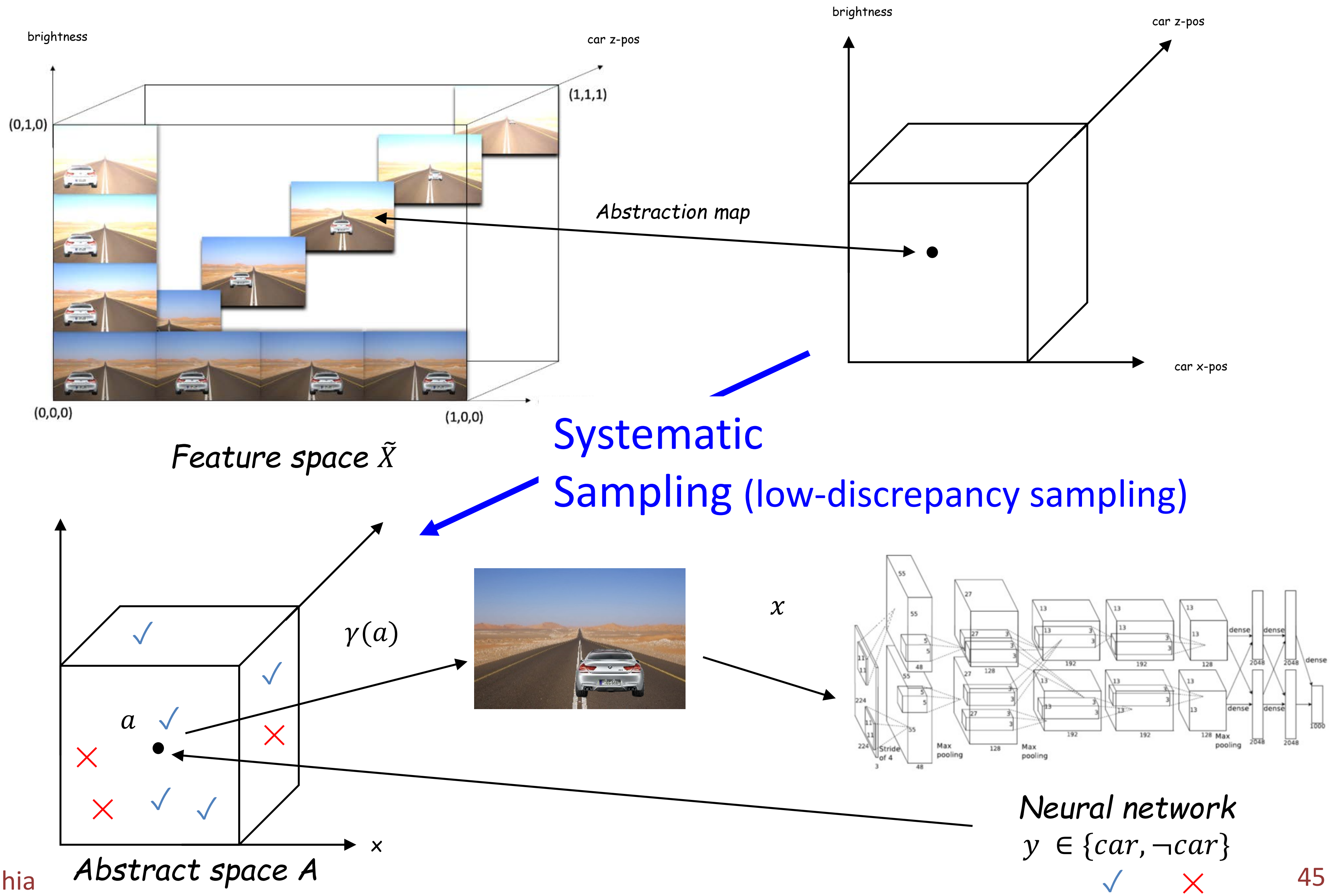


ML always correct | *ML always wrong* | *Potentially unsafe region (ROU) depending on ML component (yellow)*

Perform Optimistic and Pessimistic Analyses on the Deep Neural Network

Machine Learning Analyzer

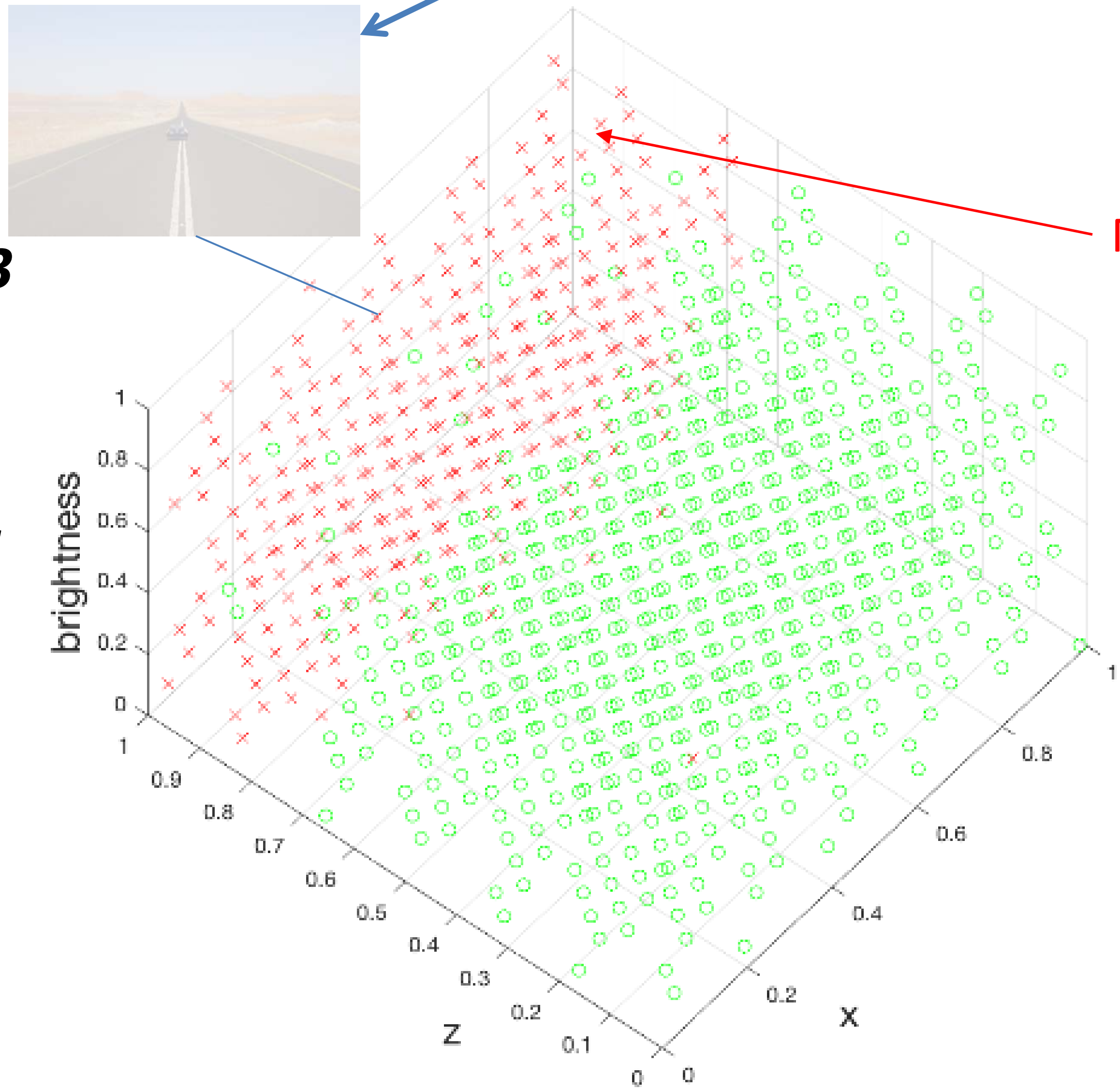
Systematically Explore ROU in the Image (Sensor) Space



Sample Result

This misclassification may not be of concern

Misclassifications



Inception-v3

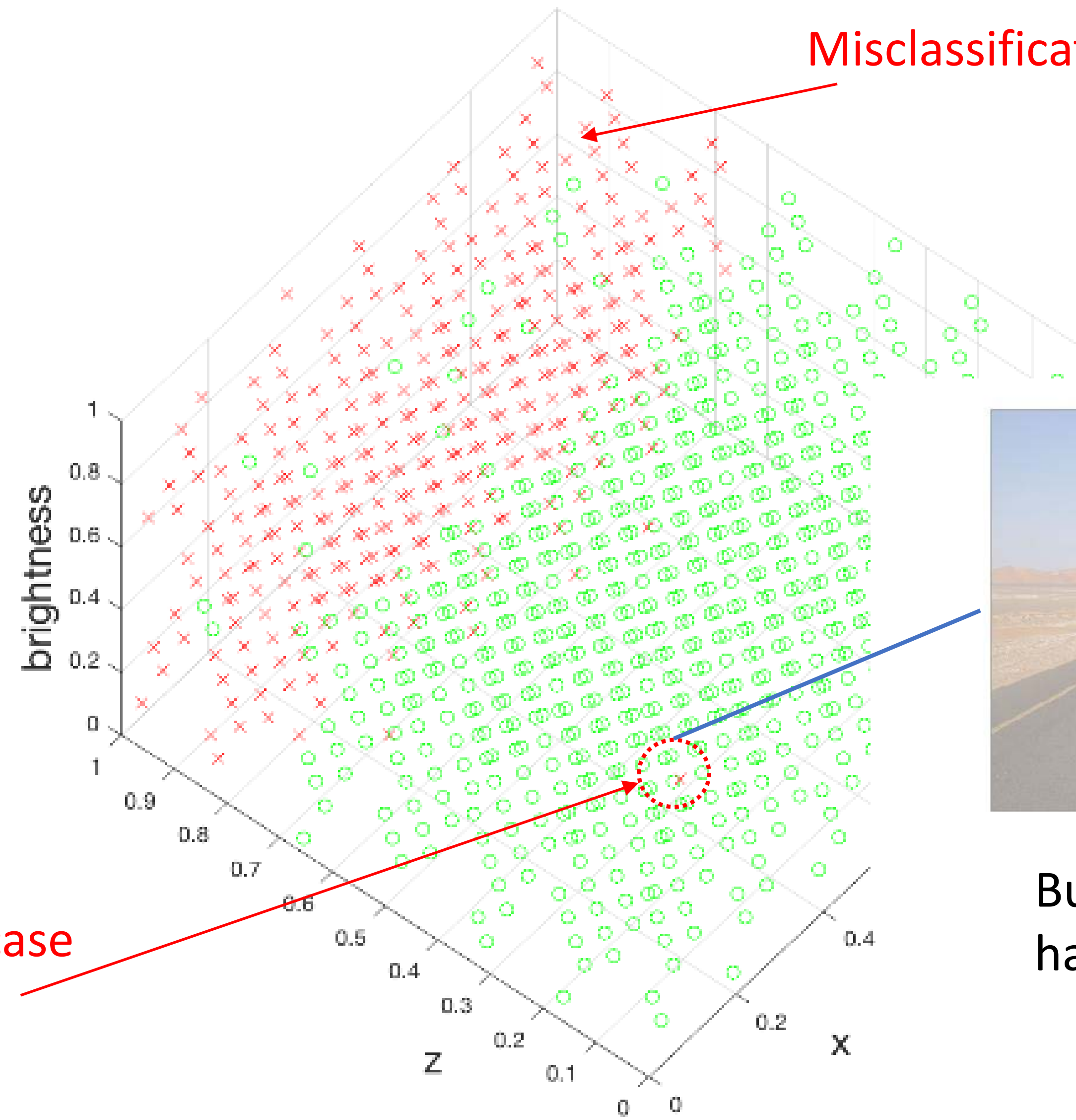
Neural Network

(pre-trained on ImageNet using TensorFlow)

Sample Result

Inception-v3
Neural Network
(pre-trained on ImageNet using TensorFlow)

Misclassifications

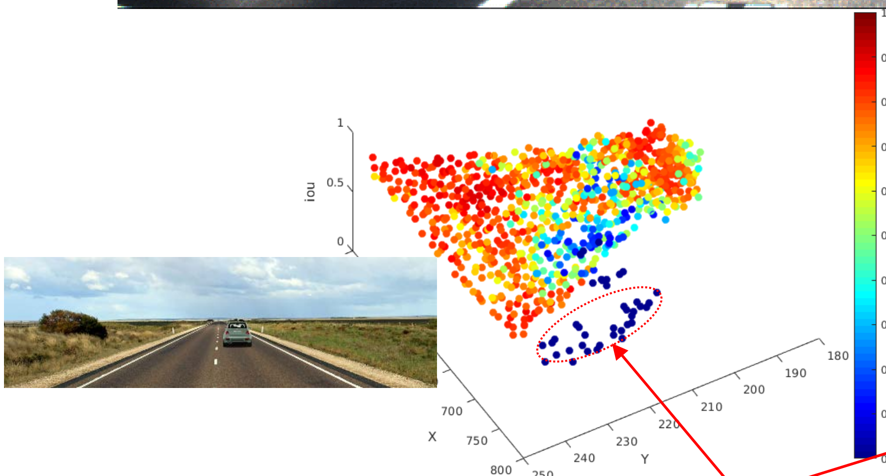


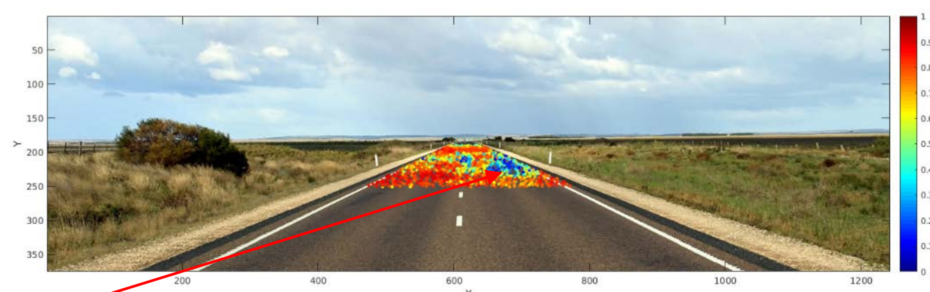
Corner case Image

But this one is a real hazard!

Image Streams

[Dreossi, Ghosh, et al., ICML 2017 workshop]





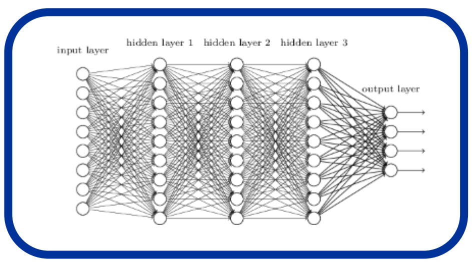
Superimposition of tests on background

Blind spots

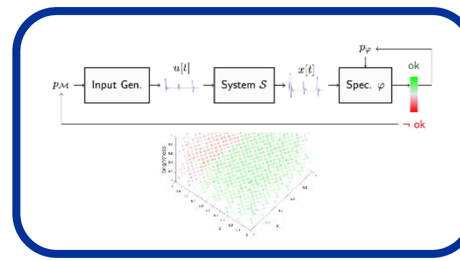
Results on squeezeDet NN and KITTI dataset for autonomous driving

Verifier-Guided Training of Deep Neural Networks

- Instance of Oracle-Guided Inductive Synthesis
- Oracle is Verifier (CPSML Falsifier) used to perform counterexample-guided training of DNNs
- Substantially increase accuracy with only few additional examples



Learned Classifier



DEEP NEURAL NETWORK

FALSIFIER (CPS + ML)

Conclusion: Formal Methods meets Machine Learning

- Formal Methods can play an important role in CPS Design with high assurance
  - Industrial scale and machine learning pose particular challenges
- Machine Learning → Formal Methods
  - Formal Inductive Synthesis (of specifications, programs, etc.)
- Formal Methods → Machine Learning
  - Compositional reasoning about learning-based systems

Towards Verified Learning-based CPS

Challenges		Principles
1. Environment (incl. Human) Modeling	→	Data-Driven, Introspective Environment Modeling
2. Specification	→	System-Level Specification; Robustness/Quantitative Spec.
3. Learning Systems Complexity	→	Abstract & Explain
4. Efficient Training, Testing, Verification	→	Verification-Guided, Adversarial Analysis and Improvisation
5. Design for Correctness	→	Formal Inductive Synthesis

Exciting Times Ahead!!! Thank you!

S. A. Seshia, D. Sadigh, S. S. Sastry. *Towards Verified Artificial Intelligence*. July 2016. https://arxiv.org/abs/1606.08514.