$\square((g=5 \wedge \omega<x) \rightarrow \diamond_{[0,\tau]}\, g=4)$

$\square(\text{idle} \rightarrow \omega>1100\ \text{RPM})$

$\square(\text{turnoff} \rightarrow \diamond_{[0,\tau]}\, cc=\text{off})$

$\square((g\geq 1 \wedge \text{“other”} \rightarrow \omega_{em}>0)$

Testing “it” in an open world assumption: a case for formal requirements

Georgios Fainekos
Toyota NA R&D

Oct 2023 @ "It Works Really Well!": Verification in Theory and Practice (IROS 2023)

Disclaimer

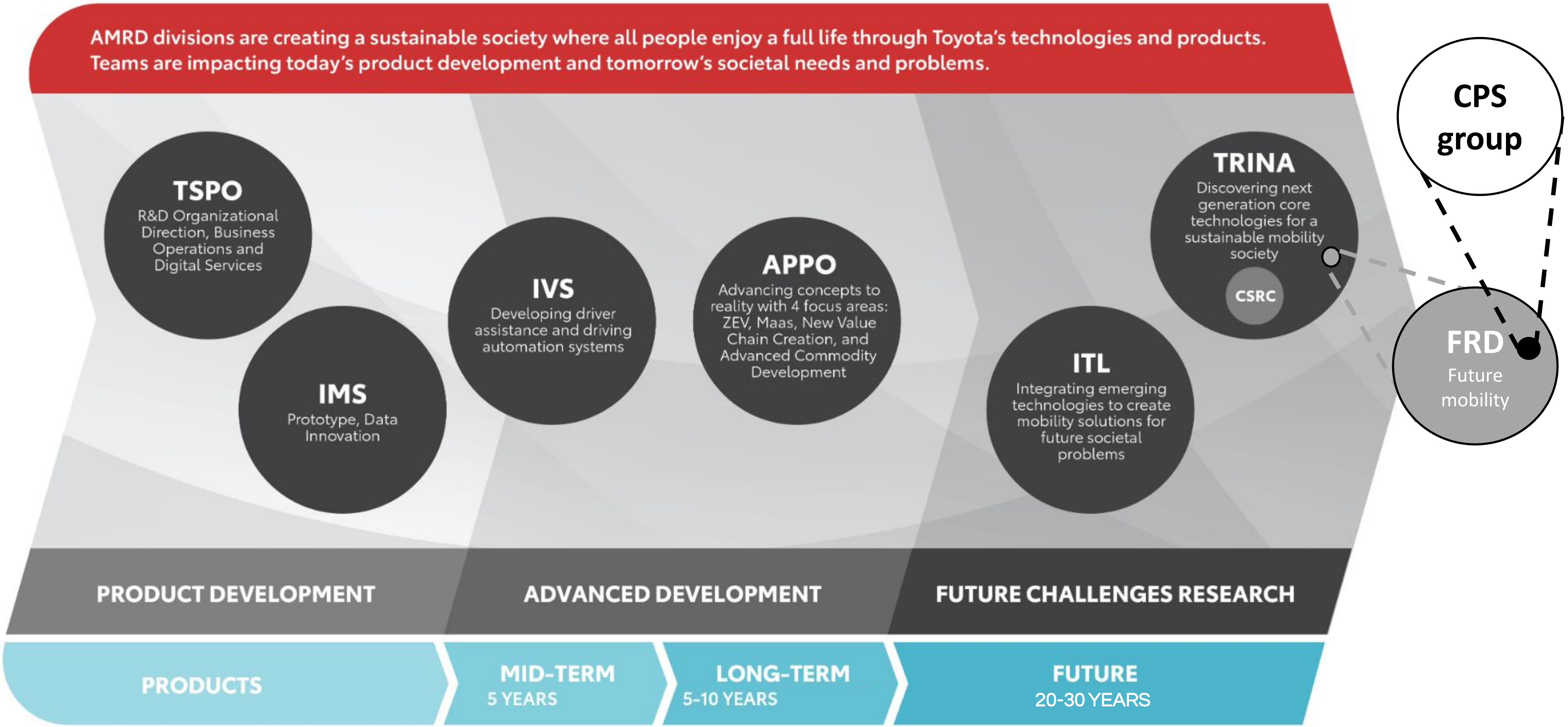
The views, opinions and technical results presented here are those of myself and my co-authors as subject matter experts, and they do not necessarily reflect the official policy or position of Toyota (or any of its member companies).

The presenter is solely responsible for the accuracy and validity of the information presented.

Many of the results presented here are with external collaborators, and the code is (or will be) available for use under standard open-source code permissions.

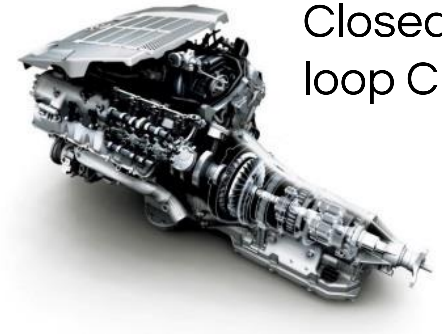
Advanced Mobility R&D within TMNA R&D

Advancing R&D to discover better ways of moving people, goods and information



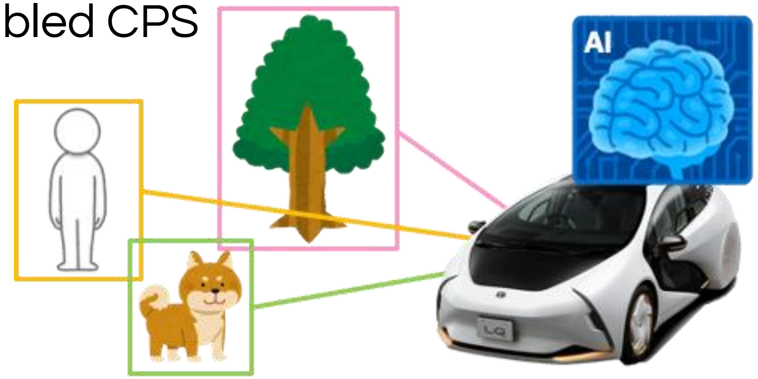
Our research focus

① Requirements, verification, validation, testing, certification



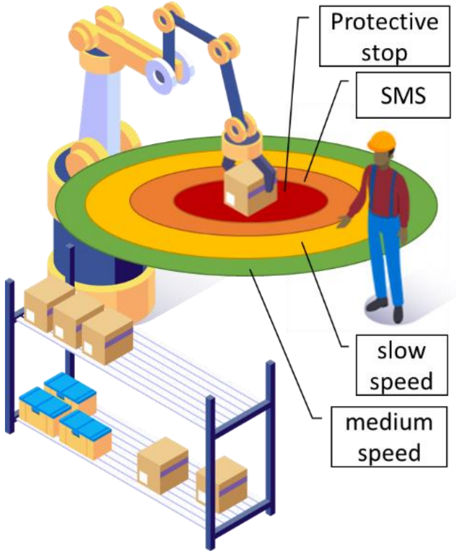
Closed-loop CPS

Machine learning enabled CPS



② Planning and control of heterogeneous multi-agent systems

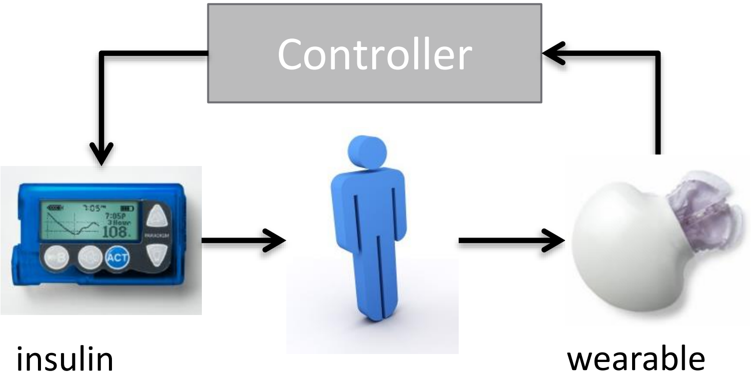
③ Human-robot interaction (planning, communication, coordination)



Moving on …

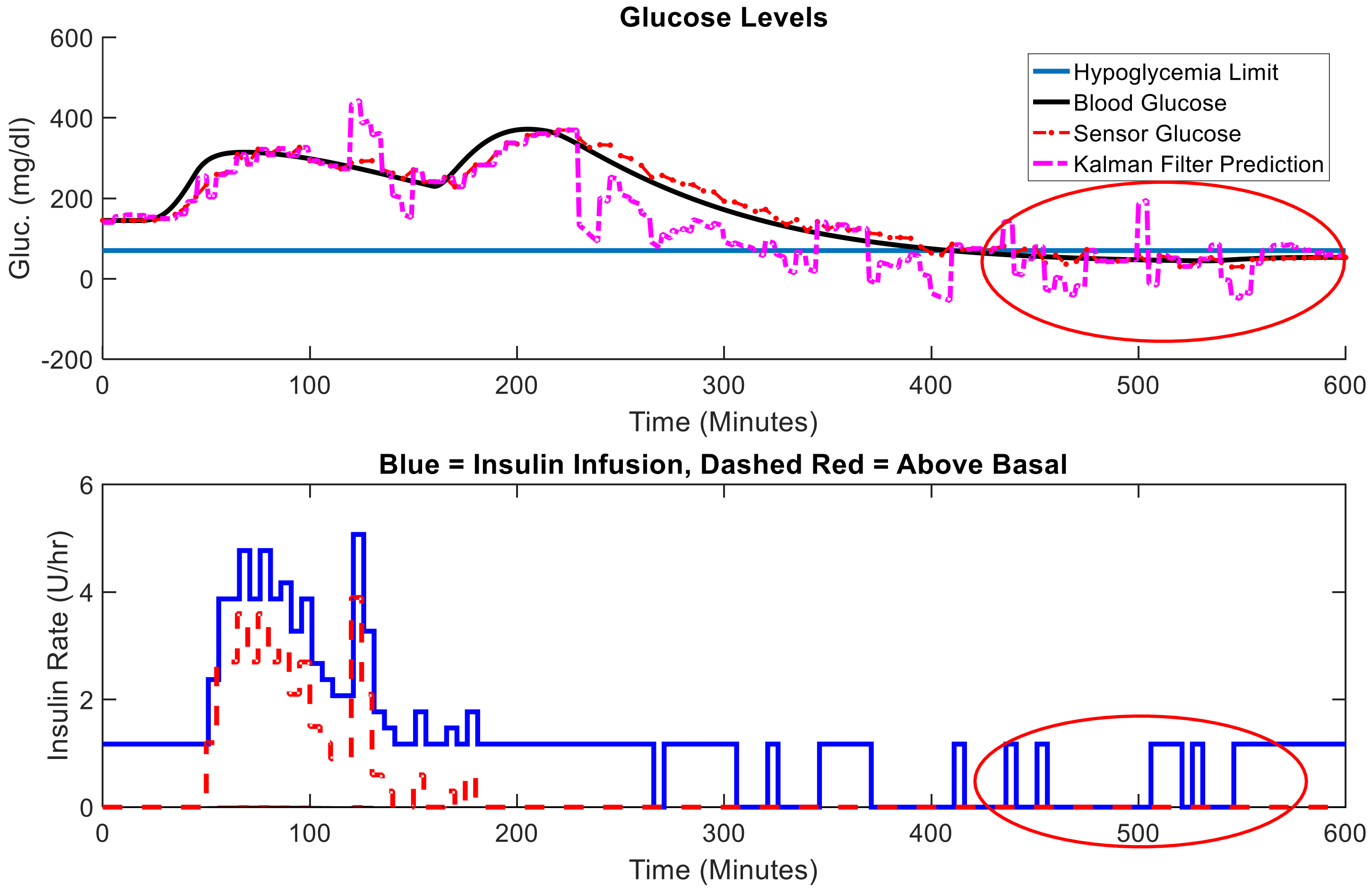
What is “it”: System-under-test (SUT)

Medical Devices: Artificial Pancreas



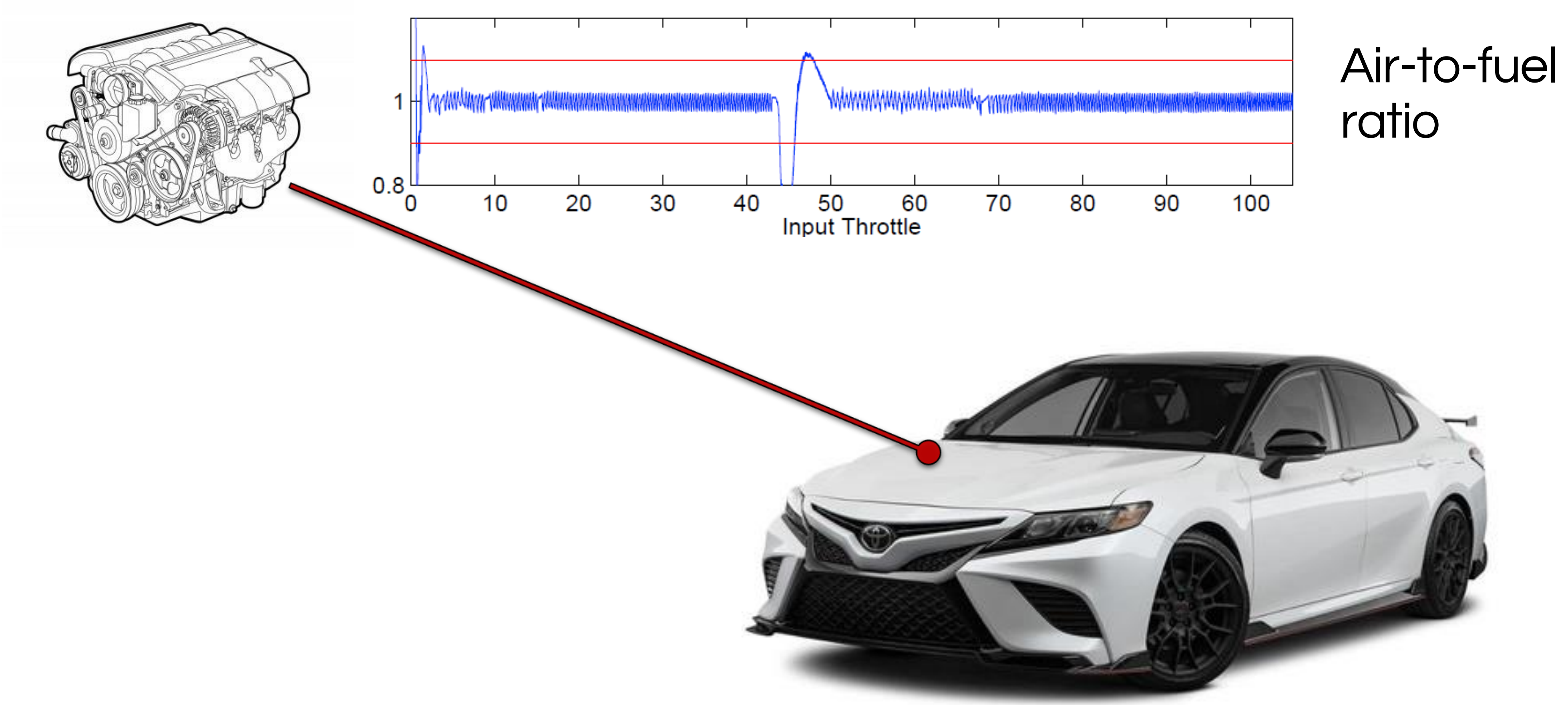
insulin
Infusion Pump

wearable
continuous
glucose
monitor



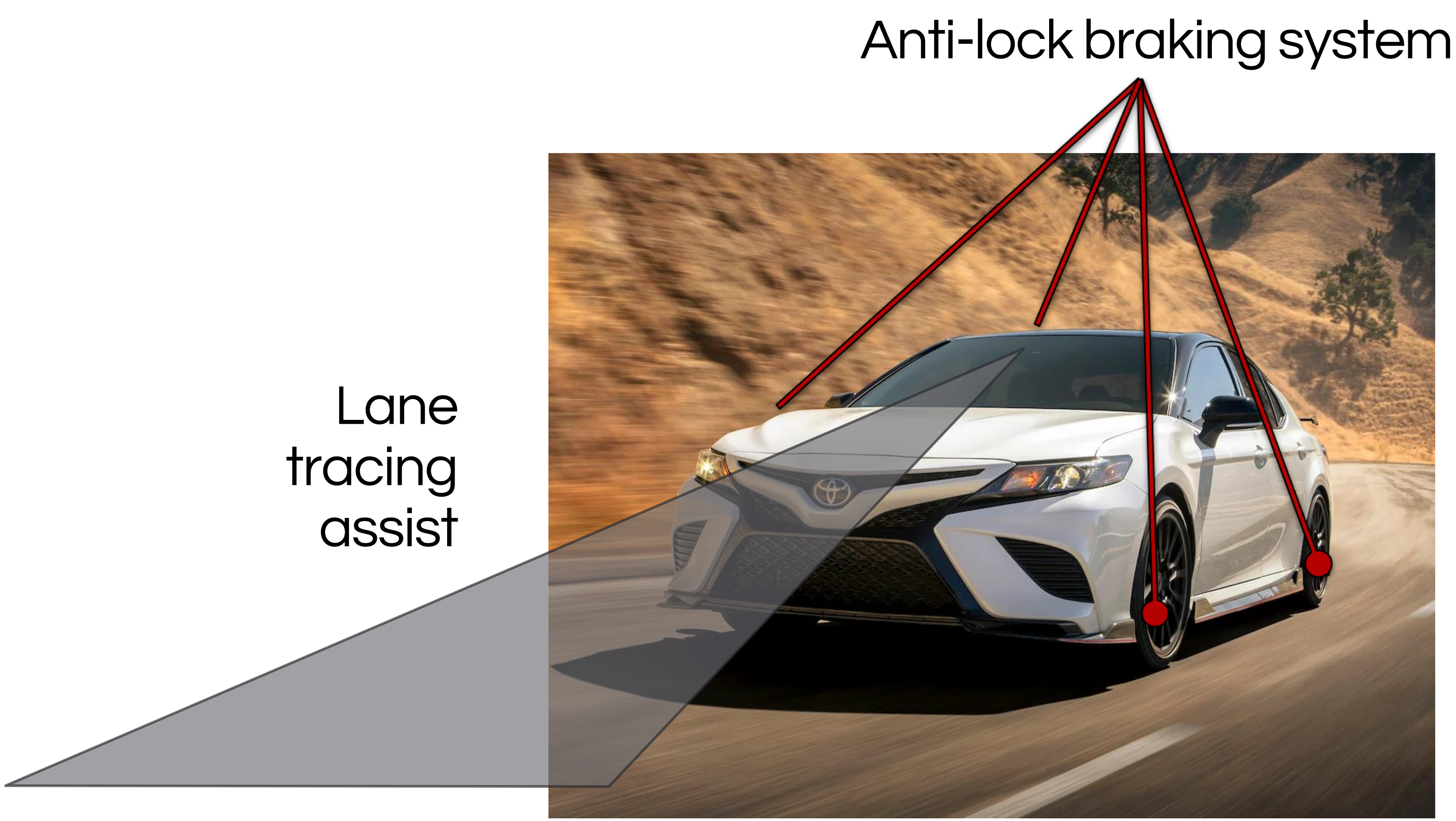
Sankaranarayanan et al, Model-based falsification of an artificial pancreas control system, ACM SIGBED Review 14 (2), 24-33

What is “it”: System-under-test (SUT)



What is "it": System-under-test (SUT)

Standard driver assistance systems



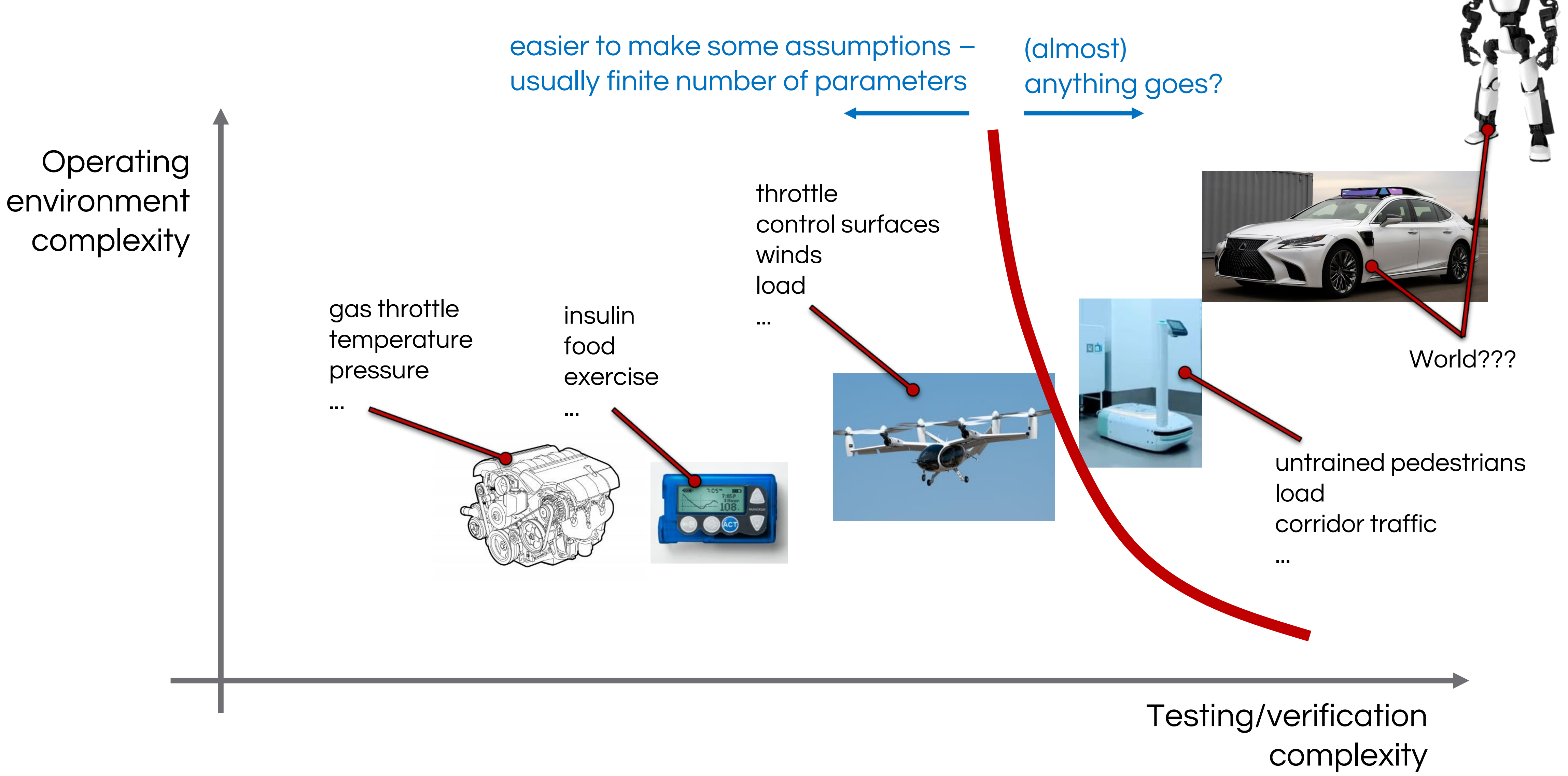
What is “it”: System-under-test (SUT)

From automated driving systems to autonomy?

Toyota Memorial Hospital

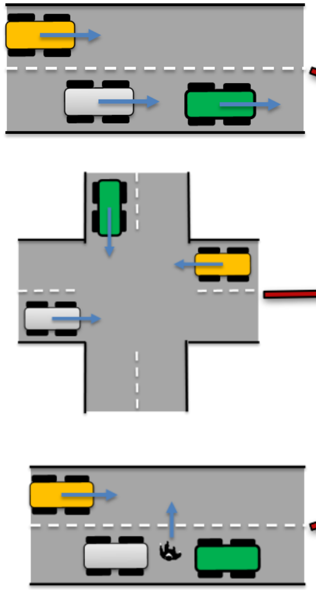
e-Palette

Testing/verification complexity*?



For this graph, we ignore criticality/importance of the verification task

Current practice in testing and verification



Scenarios		Parameters						
		p_1			…	p_m		
		v_{11}	…	v_{1n}	…	v_{m1}	…	v_{mn}
	s_1	95		100				
	s_2	10		99				
	s_3	0		55				
	⋮							

Pass/Fail entries, or even better some **quantitative metric**

Benefits:

- Easy: A way to systemize testing (regression, system level, unit, etc)
- Informative: Is my new version better than the old one?

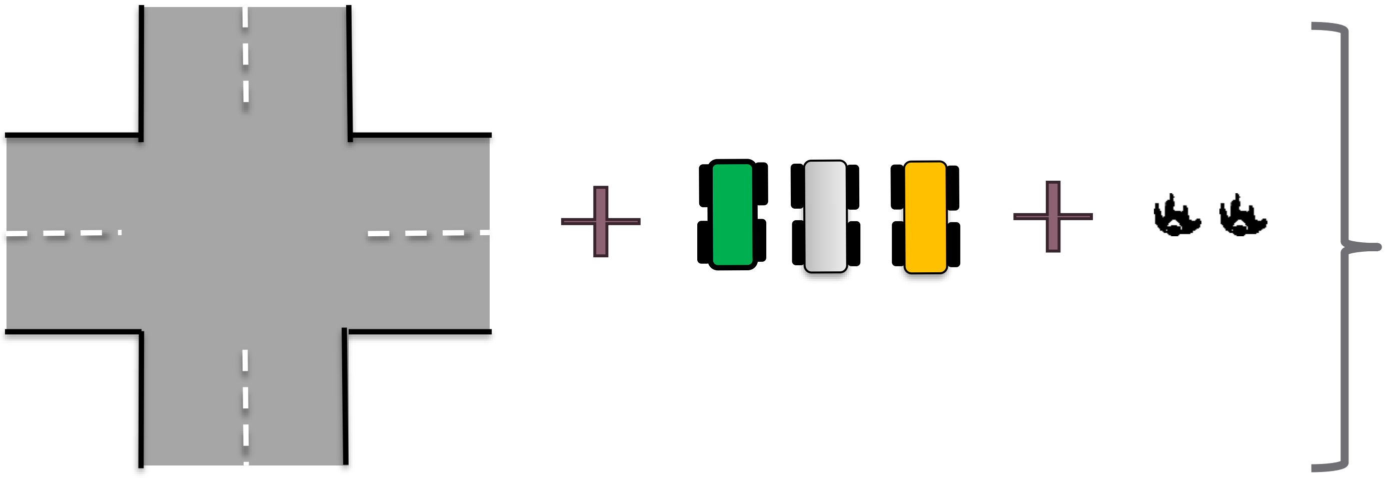
Challenges with table driven V&V

1. What is an appropriate functional/performance requirements language to provide quantitative metrics?
   - Are you running the right tests? Are all tests useful?

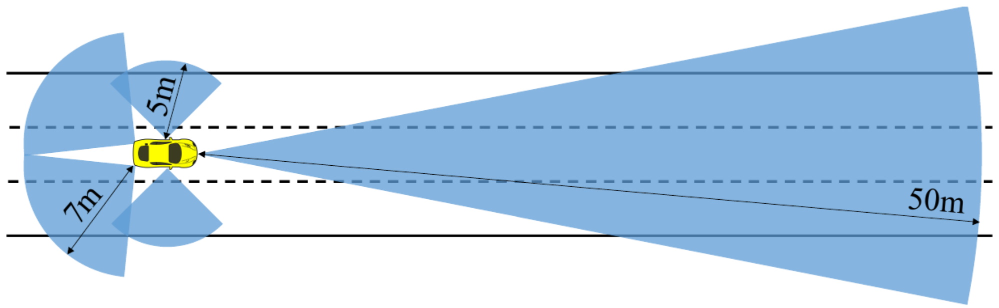
2. How do you know that your parameter discretization/quantization is good enough?
   - What guarantees can you provide?

Scenario exploration

- Can we avoid continuous parameter discretization?



Combinatorial search over discrete parameters, e.g., number of lanes, number of cars, etc

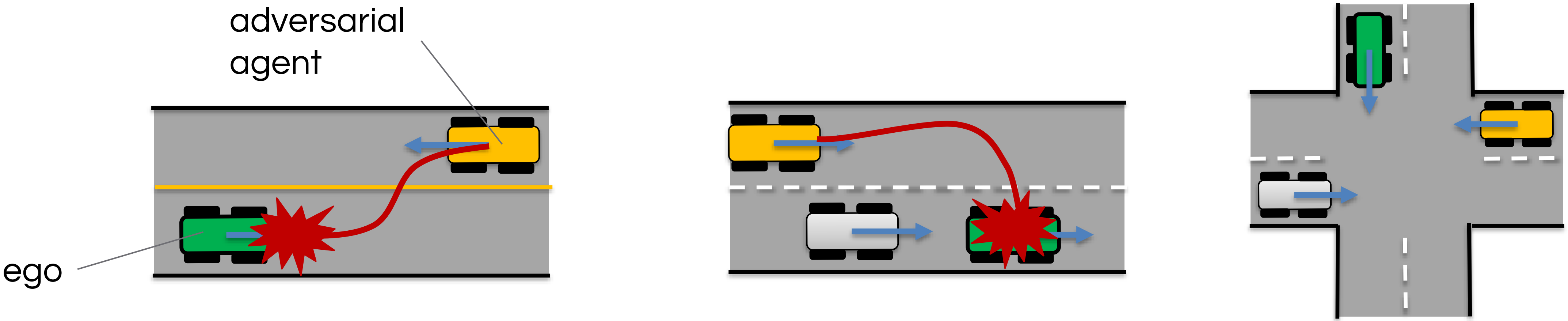


Tuncali, Fainekos, Rapidly-exploring random trees for testing automated vehicles, IEEE ITSC 2019 (best paper award)

Requirements and quantitative measures

To classify something as an "error" (unsafe behavior), we must formalize what a correct behavior is …

- If we know what safe behaviors are, then we can search for violations
- A simple requirement for ego as "*Never crash*" is not sufficient!



More useful requirement: "Never crash" unless what?

Formal requirements to capture the “unless”

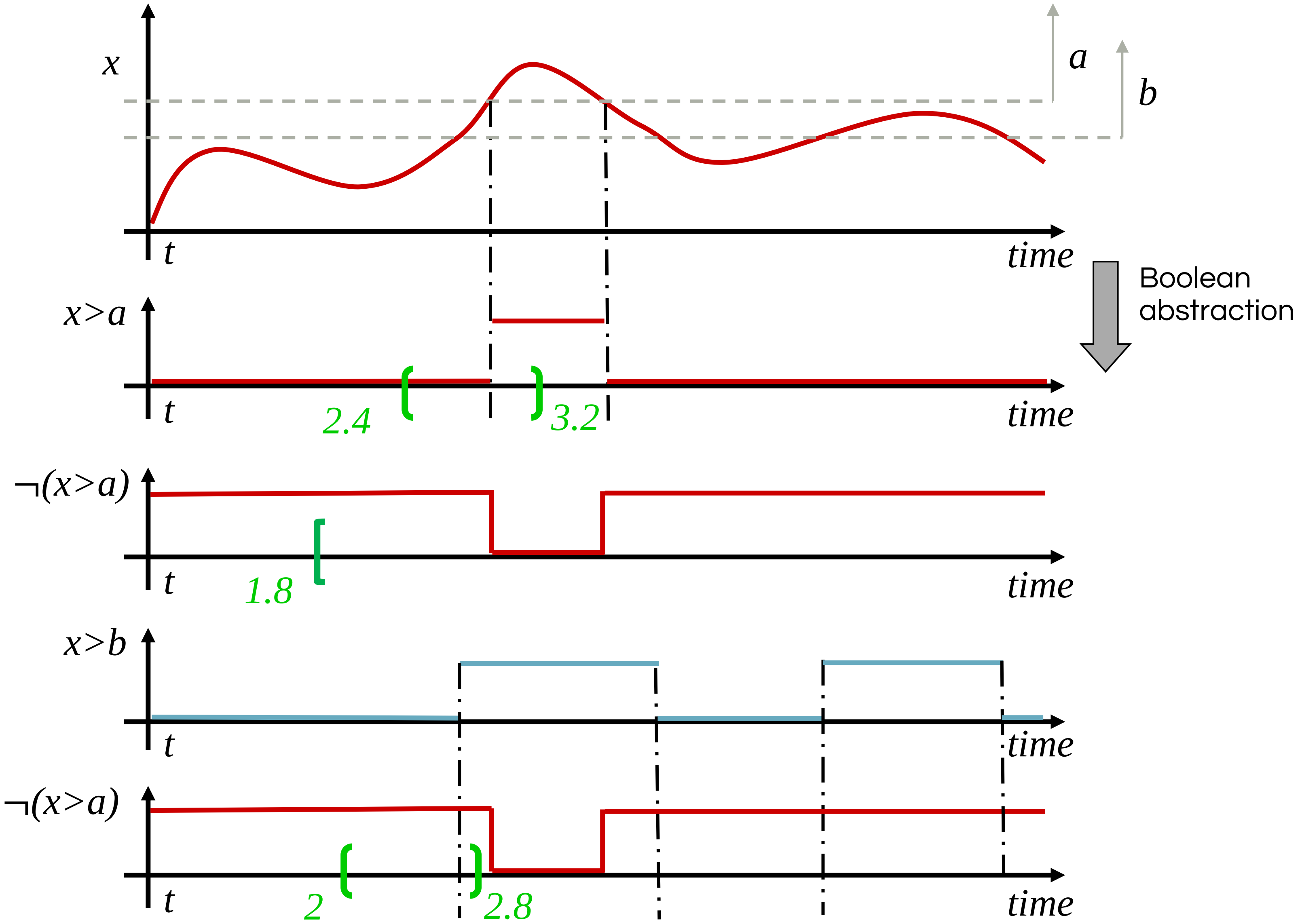
Challenge 1

Signal Temporal Logic* (STL) : Examples on signals

$F_{[2.4,3.2]}(x>a)$

$G_{[1.8,\infty)}\neg(x>a)$

$(\neg(x>a))\ U_{(2,2.8)}(x>b)$



Maler and Nickovic. "Monitoring temporal properties of continuous signals" FORMATS 2004

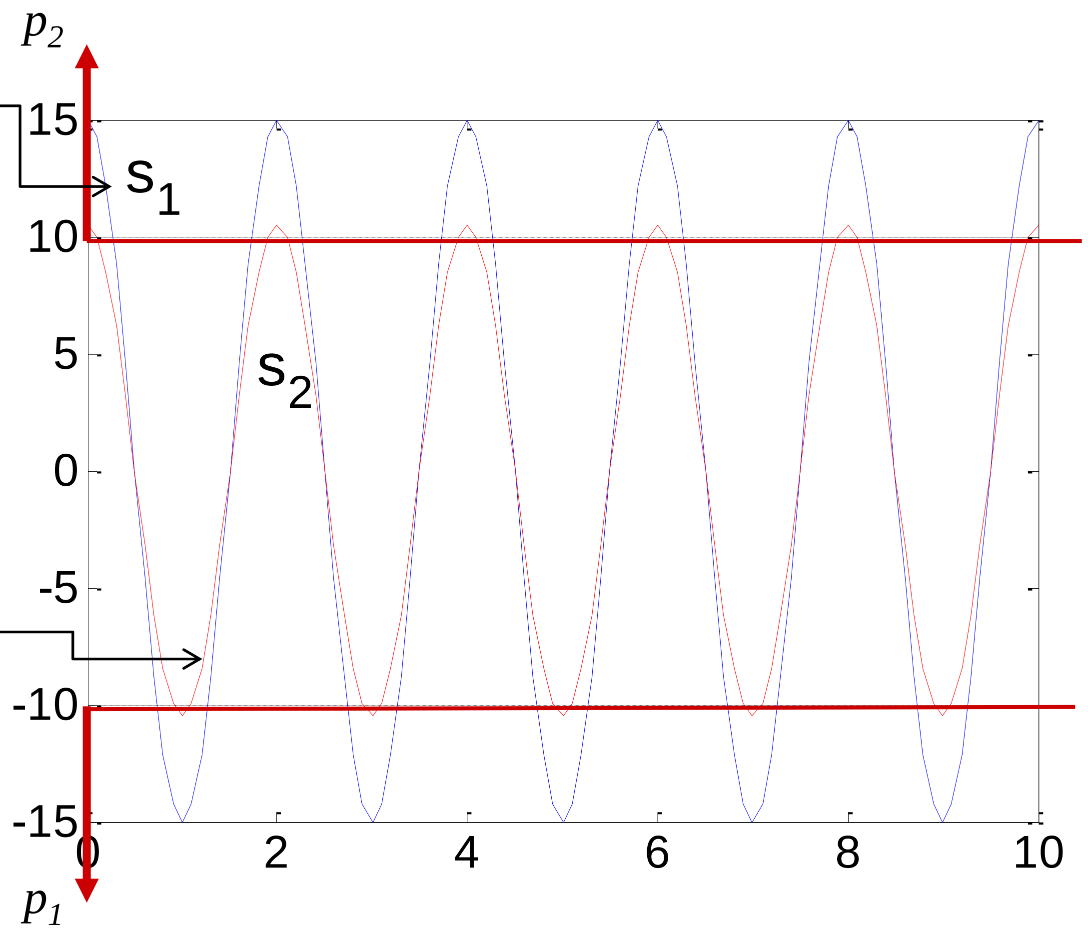
Now satisfaction can be quantified …

$[\![G(p_1 \rightarrow F_{\leq 2}\, p_2)]\!](s_1) = 5$

STL Spec:

$G((s_i \leq -10) \rightarrow F_{\leq 2}\,(s_i \geq 10))$

$(s_i \leq -10)$: p_1; $(s_i \geq 10)$: p_2



$[\![G(p_1 \rightarrow F_{\leq 2}\, p_2)]\!](s_2) = 0.5$

Fainekos and Pappas, Robustness of temporal logic specifications, FATES/RV, 2006

Example: Formalize responsible driving

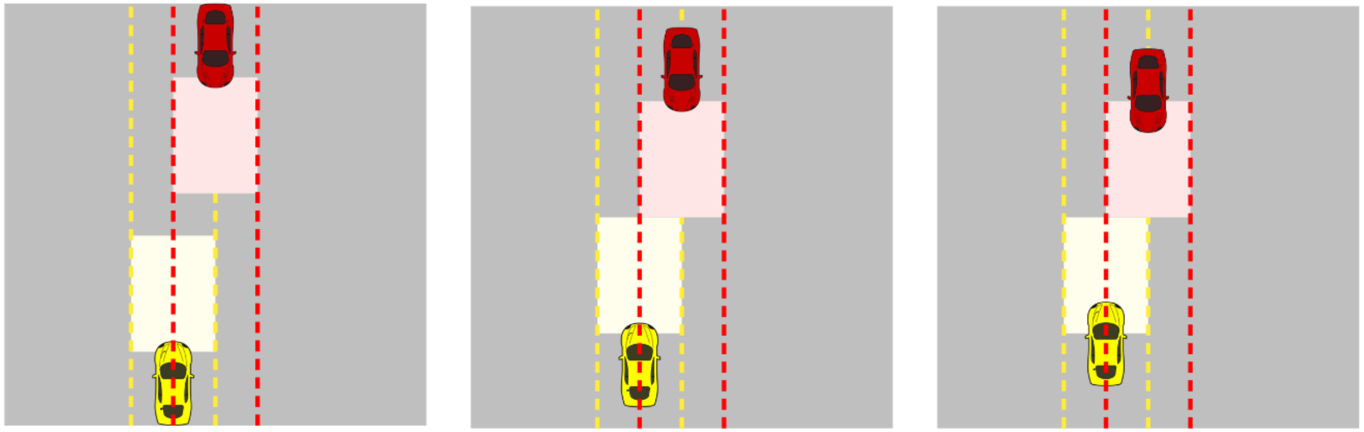
or when an ADS should not be blamed for an accident

Capture safe driver behavior for automated driving systems

1. **Intel Mobileye:** Responsibility Sensitive Safety (RSS) Rules
   - S. Shalev-Shwartz, S. Shammah, and A. Shashua, “On a formal model of safe and scalable self-driving cars,” arXiv:1708.06374v6, 2018
2. **NVIDIA:** The Safety Force Field

Example:

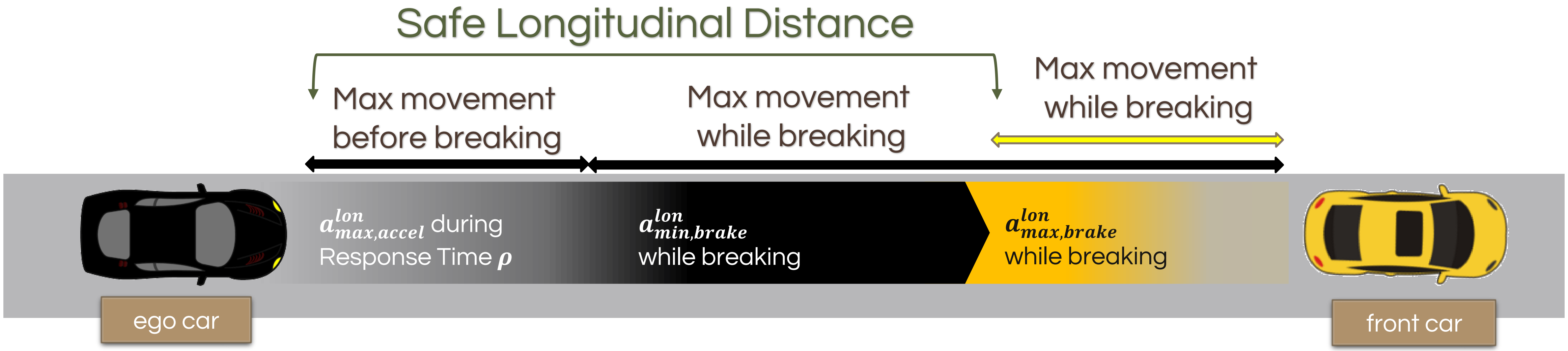
From Figure 3 from [1]



“… before the Danger Threshold time there was a safe longitudinal distance, in an on coming scenario, hence both car should brake longitudinally.”

RSS: Safe Longitudinal Distance in One-Way Traffic

All cars move at the same direction from left to the right



$$\varphi^{lon}_{resp} \equiv G((S^{lon}_{b,f} \wedge X\neg S^{lon}_{b,f}) \rightarrow X\boldsymbol{P}^{lon})$$

$$\boldsymbol{P}^{lon} \equiv \left(S^{lon}_{b,f}\bar{\mathcal{R}}_{[0,\rho)}(A^{lon}_{b,maxAcc} \wedge A^{lon}_{f,maxBr})\right) \wedge \left(S^{lon}_{b,f}\bar{\mathcal{R}}_{[\rho,+\infty)}(A^{lon}_{b,minBr} \wedge A^{lon}_{f,maxBr})\right)$$

$$S^{lon}_{b,f} \equiv \gamma(y_f, x_f)_y - \gamma(y_b, x_b)_y - d_{min,lon} > 0$$

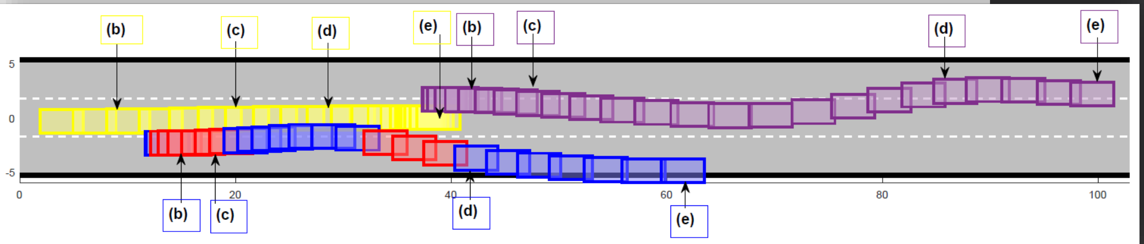
Hekmatnejad, et al, Encoding and Monitoring Responsibility Sensitive Safety (RSS) Rules for Automated Vehicles in Signal Temporal Logic, ACM-IEEE MEMOCODE 2019

Basic Proper Response Specification

- $\boldsymbol{\varphi_{resp}^{lat,lon}} \equiv \varphi^{lon} \wedge \varphi^{lat} \wedge \varphi^{lat,lon}$
- $\varphi^{lon} \equiv \square\left(\left(\neg S_{l,r}^{lat} \wedge S_{b,f}^{lon} \wedge \circ\left(\neg S_{l,r}^{lat} \wedge \neg S_{b,f}^{lon}\right)\right) \rightarrow \circ P_{lat}^{lon}\right)$
- $\varphi^{lat} \equiv \square\left(\left(\neg S_{b,f}^{lon} \wedge S_{l,r}^{lat} \wedge \circ\left(\neg S_{b,f}^{lon} \wedge \neg S_{l,r}^{lat}\right)\right) \rightarrow \circ P_{lon}^{lat}\right)$
- $\varphi^{lat,lon} \equiv \square\left(\left(S_{l,r}^{lat} \wedge S_{b,f}^{lon} \wedge \circ\left(\neg S_{l,r}^{lat} \wedge \neg S_{b,f}^{lon}\right)\right) \rightarrow \circ\left(P_{lat}^{lon} \vee P_{lon}^{lat}\right)\right)$
- P_{lat}^{lon} and P_{lon}^{lat} are modified versions of P^{lon} and P^{lat} where the propositions $S_{l,r}^{lat}$ and $S_{b,f}^{lon}$ are replaced with the formula $(S_{l,r}^{lat} \vee S_{b,f}^{lon})$.

TRINA TOYOTA RESEARCH INSTITUTE OF NORTH AMERICA

Experiments and Results



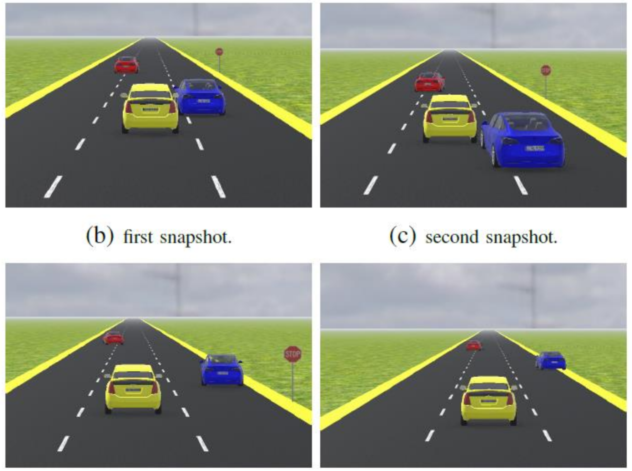
The Necessity of RSS in Testing

- 1000 test scenarios
- 23% RSS violation vs 60% CAS violation
- 19 tests did not lead to accident but violated RSS

Improving Search-based Testing through RSS

- falsifying the CAS specification
- 1000 test scenarios
- finds more dangerous test-driving scenarios
- 60% RSS violation vs 98% CAS violation
- 20 tests did not lead to accident but violated RSS

- falsifying the RSS specifications
- 350 test scenarios
- finds more relevant test-driving scenarios
- 16% RSS violation vs 85% CAS violation
- 1 test did not lead to accident but violated RSS
- classify test scenarios based on their violated constraints



(b) first snapshot. (c) second snapshot.

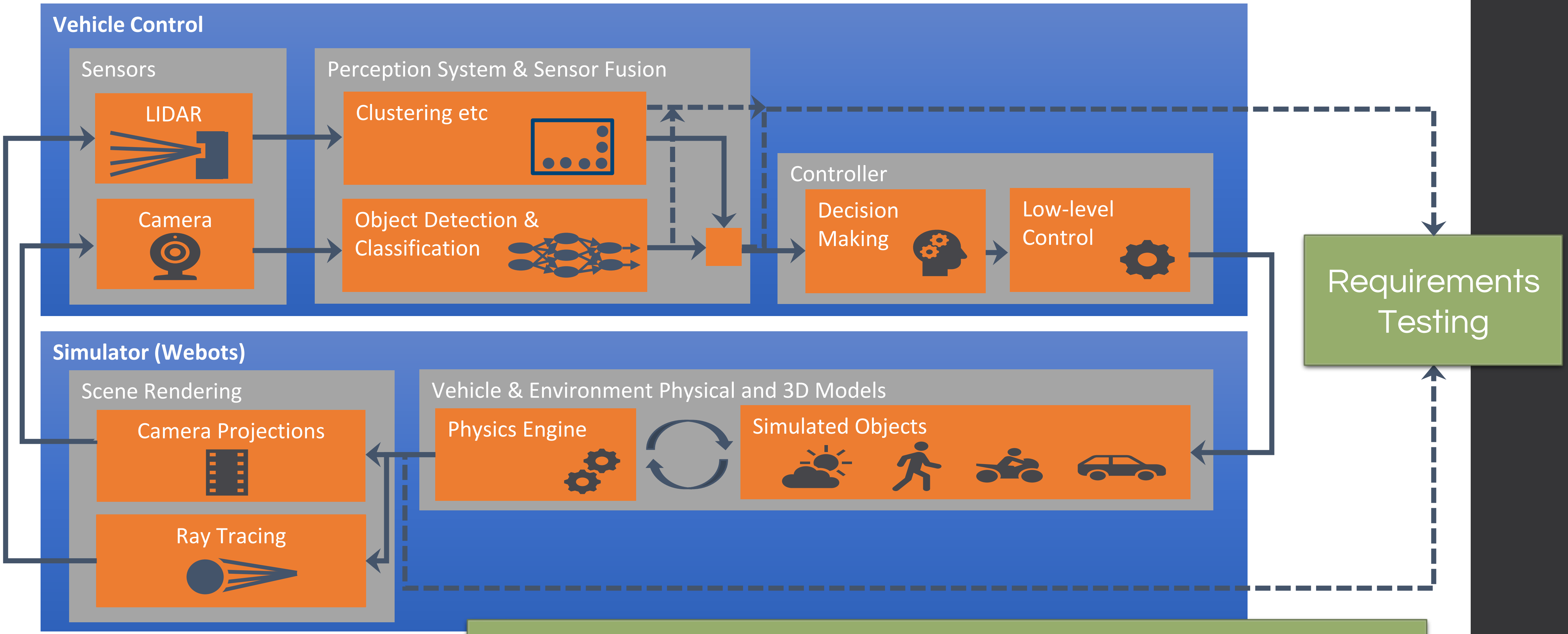
(d) third snapshot. (e) forth snapshot.

predicates	# of violations	description
$S_{b,f}^{lon}$	2	safe longitudinal distance
$S_{l,r}^{lat}$	3	safe lateral distance
$A_{l,maxAcc}^{lat}$	13	maximum allowed lateral acceleration
$A_{l,minBr}^{lat}$	0	minimum required lateral brake
$V_{l,stop}^{lat}$	0	zero $\mu-$lateral velocity
$V_{l,neg}^{lat}$	0	non-positive $\mu-$lateral velocity
$A_{b,maxAcc}^{lon}$	35	maximum allowed longitudinal acceleration
$A_{b,minBr}^{lon}$	5	minimum required longitudinal brake
Execution Statistics		
violation %	16.5%	falsified percentage using the RSS rules

Hekmatnejad, Hoxha, and Fainekos, Search-based Test-Case Generation by Monitoring Responsibility Safety Rules, IEEE ITSC 2020

TRINA TOYOTA RESEARCH INSTITUTE OF NORTH AMERICA

Perception and *System* level requirements



Main benefit of Requirements driven Simulation-based testing: We know the ground truth!

Requirements through temporal logics

Example:

If in the next 1 sec your localization error is LARGE and there exist objects which are visible but are NOT detected, then you should NOT crash within the next 5 sec.

How do you violate such a requirement?

If the localization error is large, an object is missed, and a collision occurred.

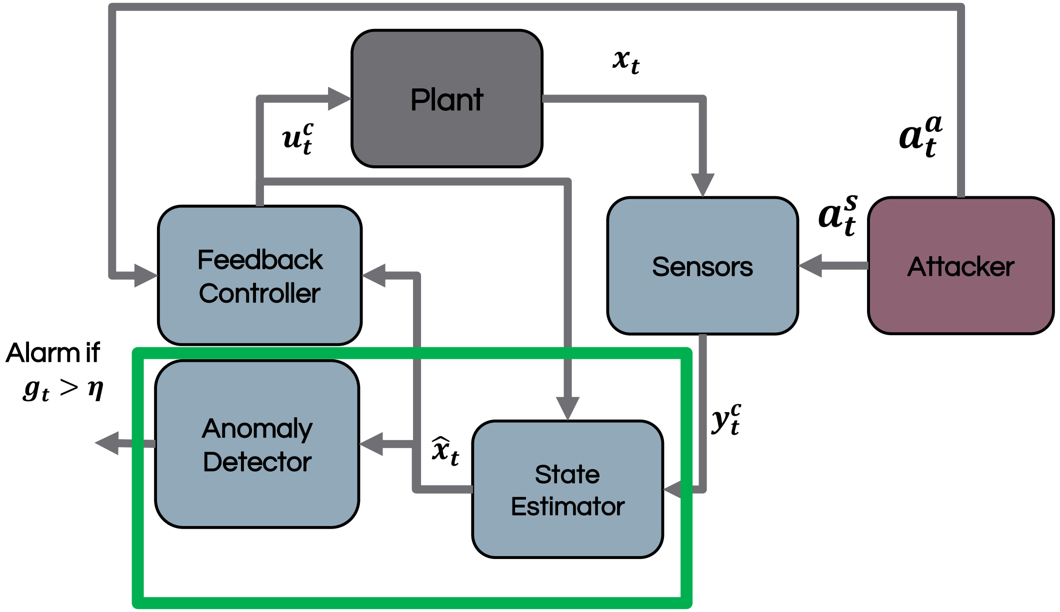
1. Search-based testing
2. Regression testing comparison
3. Online monitoring of requirements

The requirement can be assigned a numerical value of satisfaction ((+) safe; (-) unsafe).

TRINA TOYOTA RESEARCH INSTITUTE OF NORTH AMERICA

Tuncali et al, Requirements-driven test generation for autonomous vehicles with machine learning components, IEEE TIV 2019

What about security related violations?



For this work, we assume we have access to sensors and actuators

Security Requirements (φ)

Instantaneous:

$$\neg\left(\Box_{[0,\text{end}]}[\boldsymbol{P}(\boldsymbol{g_t} > \boldsymbol{\eta}) < \boldsymbol{\epsilon}] \wedge \Diamond_{[0,\text{end}]}\left(P(\|\boldsymbol{x}_t - \boldsymbol{x}_t^*\| \geq \alpha)\right) \geq \gamma\right)$$

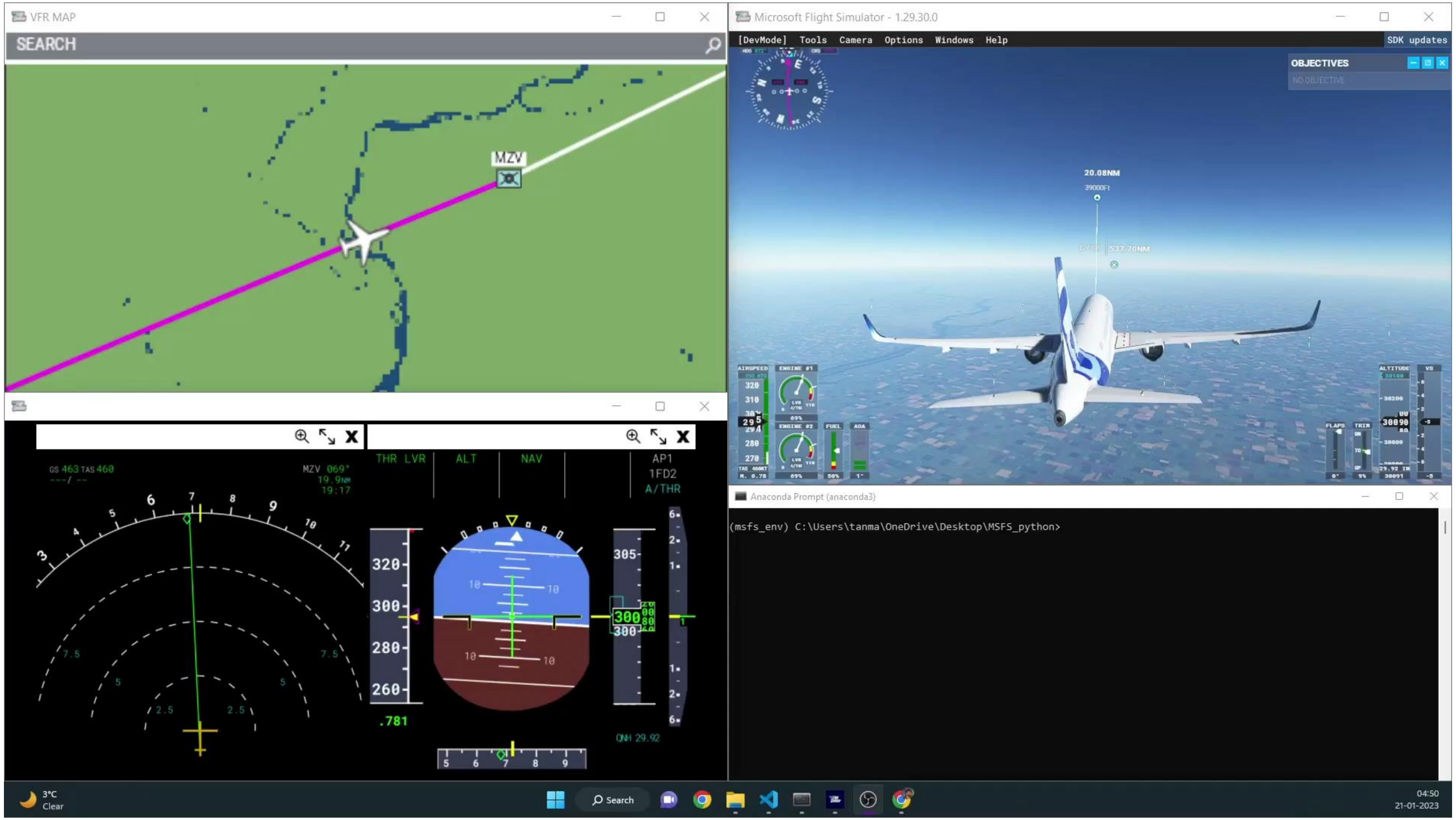
$\boldsymbol{\delta}$-sustained:

$$\neg\left([P(g_t > \eta) \leq \epsilon]\ \mathrm{U}_{[0,\text{end}]}\ \boxminus_{[0,\delta]}\left(P(\|\boldsymbol{x}_t - \boldsymbol{x}_t^*\| \geq \alpha)\right) \geq \gamma\right)$$

NOT (Probability of anomaly detector alarm going off $\leq \epsilon$ | Until | Historically | Probability of system state error exceeding threshold $\geq \gamma$)

Chandratre, et al, Stealthy attacks formalized as STL formulas for Falsification of CPS Security, ACM HSCC 2023

Falsification for discovering stealth attacks to CPS



TOYOTA RESEARCH INSTITUTE OF NORTH AMERICA

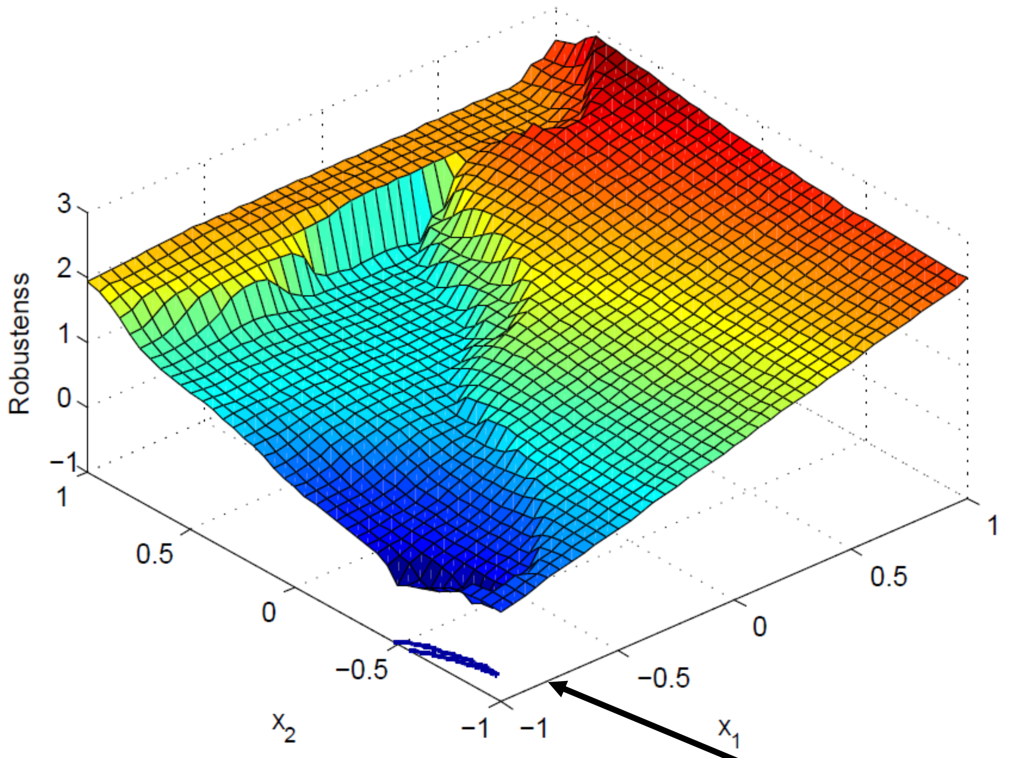
ARIZONA STATE UNIVERSITY

Guarantees

2nd Challenge

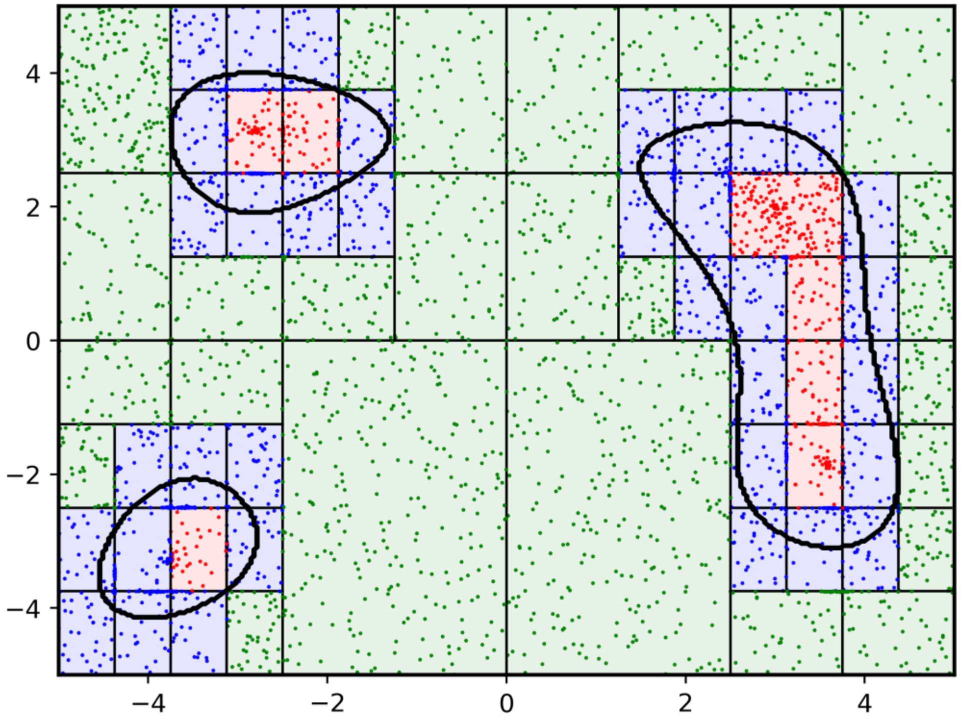
Guarantees on testing?

What if SBTG terminates, but no falsifying behavior has been detected?



What is the volume of the zero-sublevel set?

Part-X (algorithm for level set identification)



Part-X Level 6

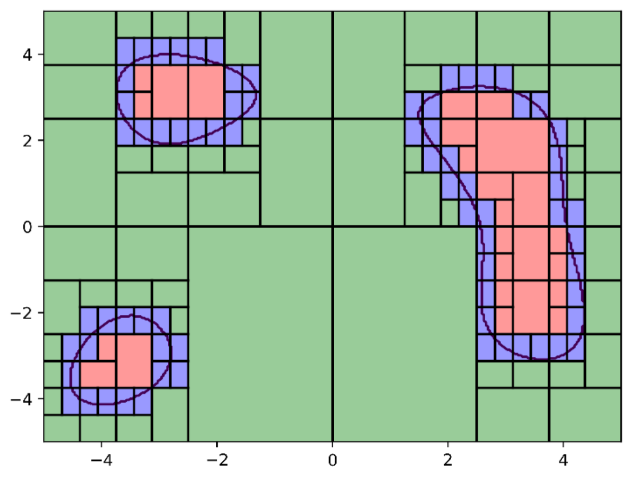
Himmelblau's Function

Headline Accomplishment

Part-X family of optimization algorithms with probabilistic guarantees.
Estimate the volume of the zero-level set within the desired confidence levels.

Benefits/advantages over Monte Carlo (MC) sampling

- Part-X identifies the regions in the search space which are the least safe with respect to the requirement
  - Part-X estimates what is the probability that a falsifying behavior exists even when no falsification is found
  - In contrast, if no falsifying behaviors are detected, then MC returns 0 as a probability of falsification
- Part-X is an importance sampling method
  - It is provably at least as good as a Monte Carlo approximation



Part-X has been released as an open-source Python package

- https://gitlab.com/bose1/part-x
- (it can also be integrated with Matlab)

TRINA TOYOTA RESEARCH INSTITUTE OF NORTH AMERICA

ASU Arizona State University

Pedrielli, et al, Part-X: A Family of Stochastic Algorithms for Search-Based Test Generation with Probabilistic Guarantees, IEEE TASE 2023

Part-X vs state of the art SBTG methods

- ARCH Falsification competition
  - Since 2017; 8 competing tools in 2021
  - Compares SBTG test generation tools on CPS models (build using Matlab/Simulink)
    - 7 different models (e.g., F16 GCAS), 1 to 10 different requirements for each model
    - All problems are falsifiable of different difficulty
    - Comparison only on how fast to detect falsifying points

- **No other tool submission provides probabilistic guarantees**
- **Part-X performs competitively even against tools designed specifically for fast detection of falsifying system behaviors**
  - In addition, Part-X provides an estimate of the falsification region

Pedrielli, et al, Part-X: A Family of Stochastic Algorithms for Search-Based Test Generation with Probabilistic Guarantees, IEEE TASE 2023

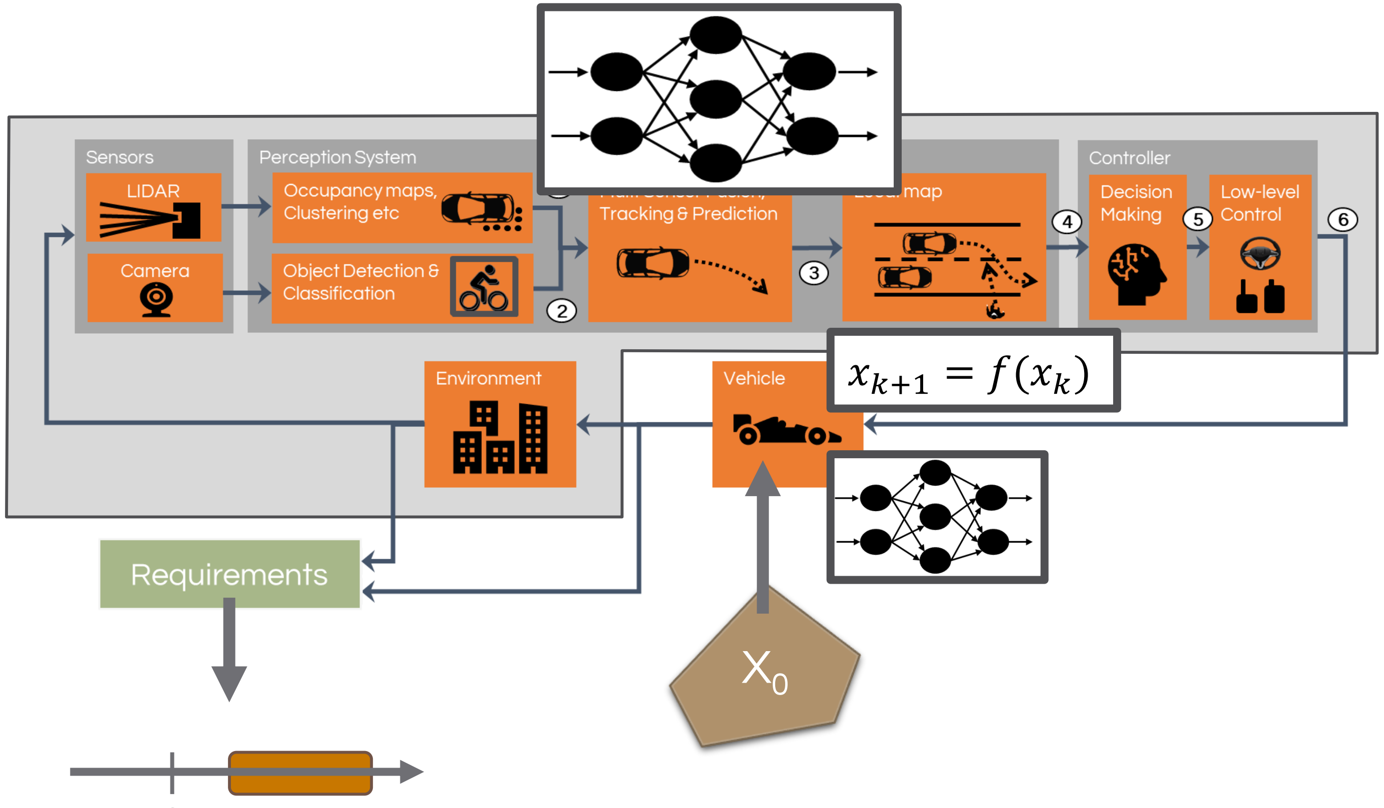
Bounded-time reachability analysis of NN-CPS

Assumptions:

- Control, perception and environment can be represented as (X)NN
- Vehicle can be model as NN or OΔE

Then:

- We can perform bounded time exhaustive verification

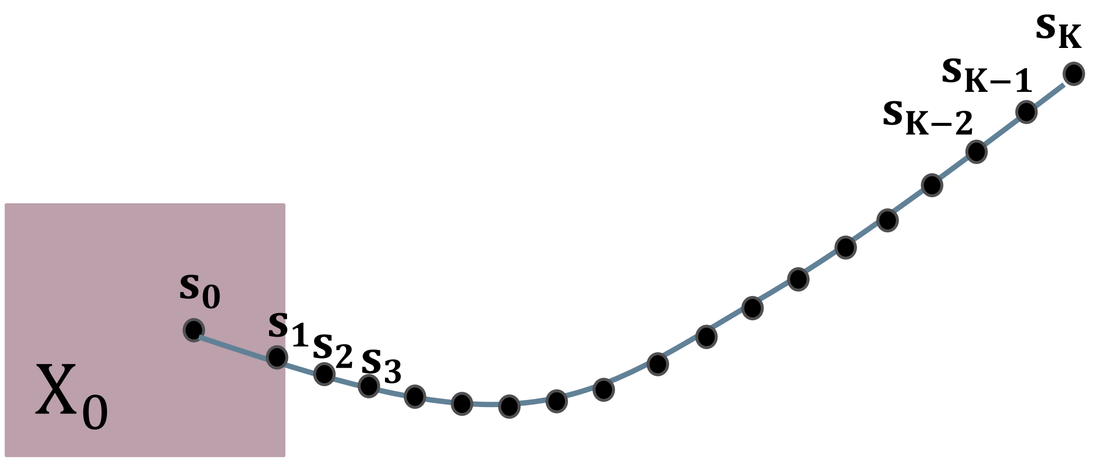


Specification robustness range up to time T

Hashemi, et al, *Neurosymbolic Approach to the Verification of Temporal Logic Properties of Learning-Enabled Control Systems, ACM/IEEE ICCPS 2023*

TRINA TOYOTA RESEARCH INSTITUTE OF NORTH AMERICA USC Viterbi

From requirements' robustness to NN



$\forall \; s_0 \in X_0$: $\{s_0, s_1, s_2, \dots, s_K\} \models$ Requirement

Theorem

$\rho(\varphi, \{s_0, s_1, \dots, s_K\}) \geq 0 \;\leftrightarrow$
$\text{STL2NN}(s_0, s_1, \dots, s_K) \geq 0 \leftrightarrow$
Temporal task is satisfied

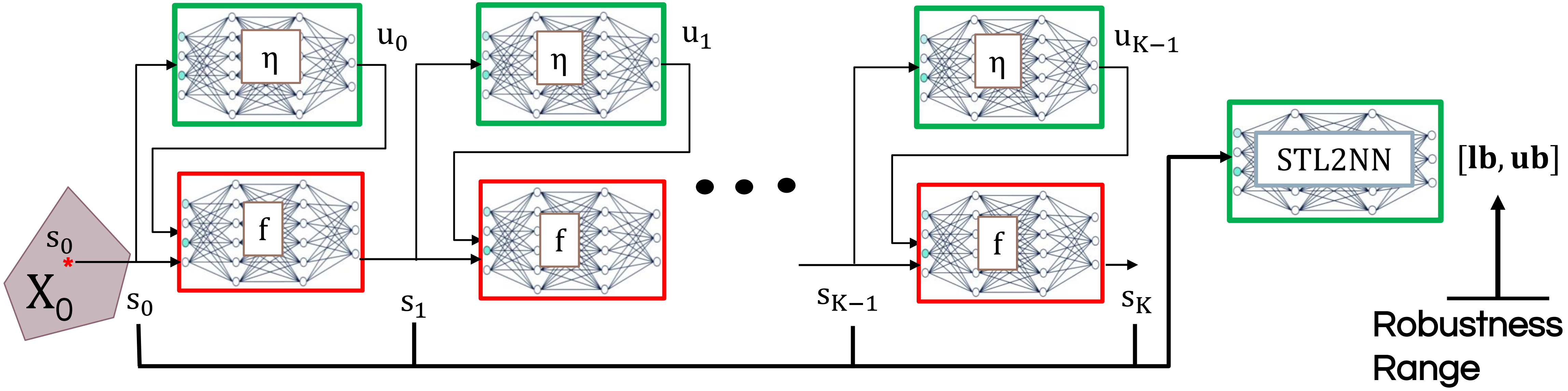
- Formulation of min/max of 2 variables in terms of ReLU neural network.

$$\min(a, b) = 0.5\,(\text{ReLU}(a+b) + \text{ReLU}(-a-b) - \text{ReLU}(b-a) - \text{ReLU}(a-b))$$

$$\max(a, b) = 0.5\,(\text{ReLU}(a+b) + \text{ReLU}(-a-b) + \text{ReLU}(b-a) + \text{ReLU}(a-b))$$

TRINA TOYOTA RESEARCH INSTITUTE OF NORTH AMERICA USC Viterbi

Benefit? Use standard NN reachability tools



Example reachability tools:
CROWN, POLAR, Sherlock, Veritex, NNV, ...

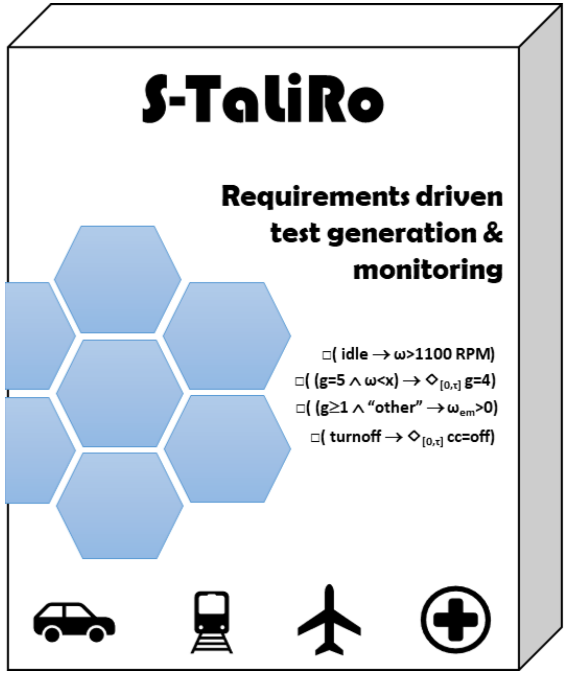
NNV
- Exact-star based reachability
- Approx-star based reachability

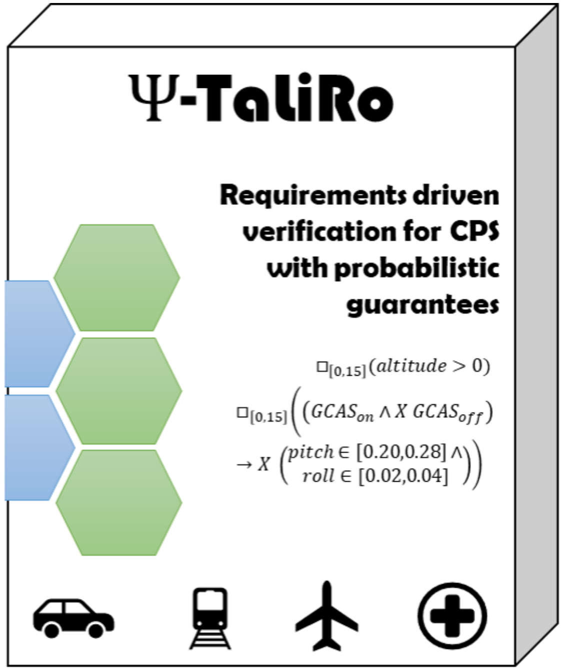
We can do requirements verification over NN-CPS using standard reachability analysis tools.

Conclusions

- To enable testing in “open” world environments we need assumptions and constraints on what reasonable tests are
  - Temporal logics can offer such a path
- Functional requirements can enable search-based testing as well as regression testing
  - Enabled through quantitative metrics
- The AI/ML verification problem may be ill defined without some physical embodiment to achieve some functionality
  - System level requirements may be better suited

Open-source software tools





S-TaLiRo (https://app.assembla.com/spaces/s-taliro_public/wiki)

A Matlab toolbox for falsification, specification mining, monitoring, and conformance testing of Cyber-Physical Systems.

See RV 2019

PSY-TaLiRo (https://github.com/cpslab-asu/psy-taliro)

A Python toolbox for falsification and verification with probabilistic guarantees of Cyber-Physical Systems.

See FMICS 2021

TRINA TOYOTA RESEARCH INSTITUTE OF NORTH AMERICA

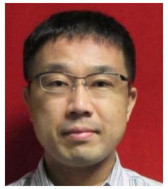
Acknowledgements (in random order)

Mohammad Hekmatnejad (ASU; now @NVIDIA)

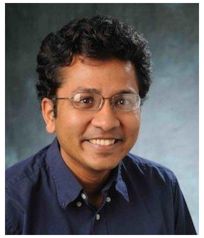
Erkan Tuncali (ASU; Now@Amazon)

Bardh Hoxha (Toyota)

Hisahiro Ito (Toyota; Now@Mathworks)



S. Sankaranarayanan (CU, Boulder)



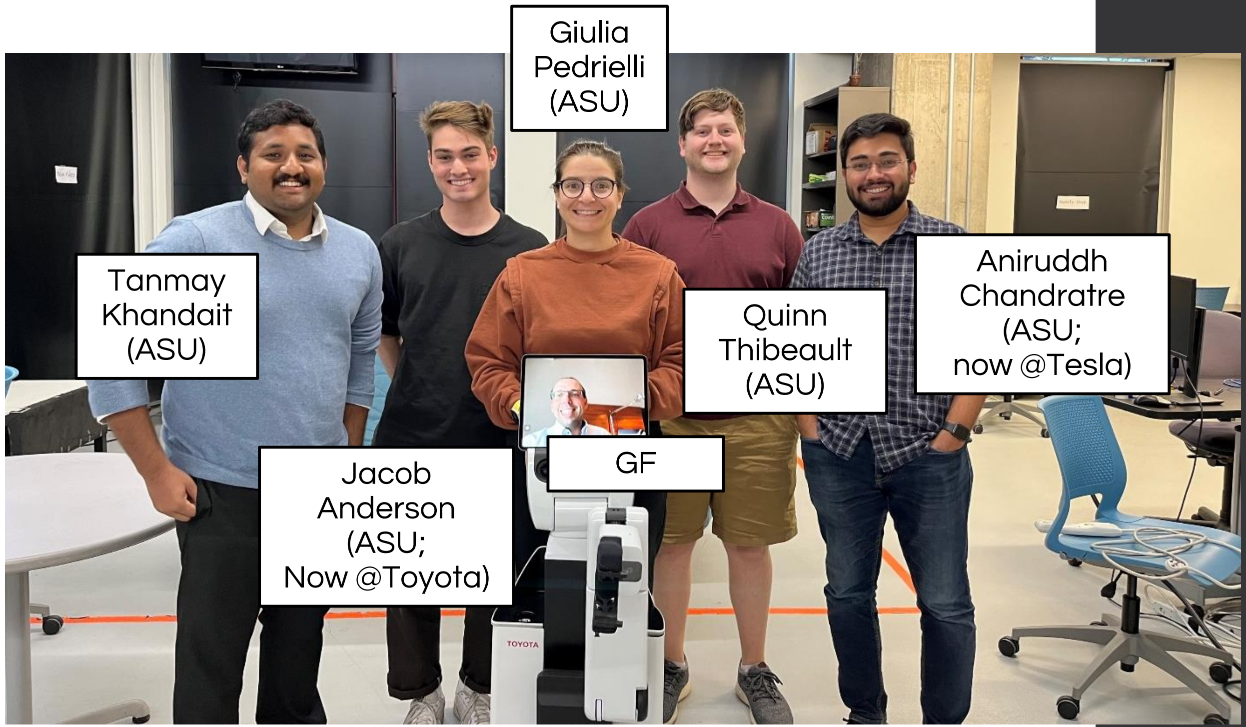
James Kapinski (Toyota Now@Amazon)

Keyvan Majd (ASU)

Danil Prokhorov (Toyota)

Geoffrey Clark (ASU; now@IHMC)

Hani Ben Amor (ASU)



Siyu Zhou (ASU; now @Microsoft)

Hao Huang (Yuan Ze University)

Mauricio Castillo-Effen (LM Co)

Surdeep Chotaliya (ASU)

TRINA TOYOTA RESEARCH INSTITUTE OF NORTH AMERICA