Robustness Guided Testing and Verification for Cyber-Physical Systems

PI: Georgios Fainekos

School of Computing, Informatics and Decision System Engineering
Arizona State University
fainekos at asu dot edu
http://www.public.asu.edu/~gfaineko

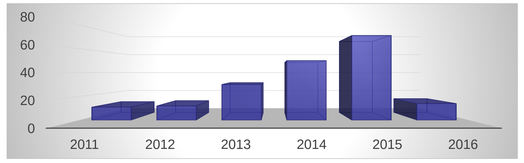
CPS Lab — ASU Arizona State University

NSF Award # 1350420

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Motivation

- The complexity of the software in Cyber-Physical Systems (CPS) is increasing almost exponentially with time.
- **Challenge:** The recent **multiple software related recalls** of automobiles and medical devices indicate that the current software development methods may be inadequate for safety critical software applications.



Automotive software recalls
Source: J.D. Power with NHTSA data (2016)

- **Foundational Challenge:** Can we develop automated driving systems (ADS) without clear requirements and specifications on system performance?
  - How do we verify safety and evaluate performance without formal requirements and standards?
  - How do we compare different designs?

Infamous case: The Uber ADS was never designed to expect jaywalking pedestrians.


ars TECHNICA SUBSCRIBE SIGN IN

SELF-DRIVING CARS

How terrible software design decisions led to Uber's deadly 2018 crash

NTSB says the system "did not include consideration for jaywalking pedestrians."

TIMOTHY B. LEE - 11/6/2019, 2:52 PM

- **Formalizing requirements for known safety critical software recalls**

When in 5th gear and RPM drops below x, then the system should always switch from 5th to 4th gear.	G((g=5 ∧ ω<x) → $F_{[0,τ]}$ g=4)
The electric motor should always rotate in the direction selected by the transmission.	G((g≥1 ∧ "other" → $ω_{em}$>0)
The cruise control should always disengage when the "turn off" button is pressed.	G(turnoff → $F_{[0,τ]}$ cc=off)

G: Always (Globally); F: Eventually (Future); X: (Next)

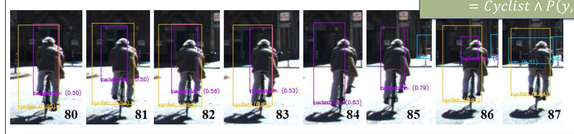
Requirements for perception systems

- A formal logic (TQTL) for reasoning about the assumptions and guarantees provided by perception systems in autonomous vehicles.

Example: Monitoring the quality of SqueezeDet object detection on KITTI data set

"At every time step, for all the objects (id) in the frame, if the object class is cyclist with probability more than 0.7, then in the next 5 frames the object (id) should still be classified as a cyclist with probability more than 0.6"

$$G(x, \forall id@x, (C(x,id) = Cyclist \wedge P(x,id) > 0.7) \rightarrow G(y, ((x \le y \wedge y \le x+5) \rightarrow (C(y,id) = Cyclist \wedge P(y,id) > 0.6)))$$



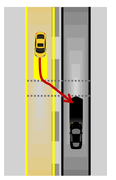
The data stream D does not contain an object classified as cyclist in Frames 84 and 85

Relaxed requirement: "… then in the next 5 frames the object (id) should still be classified as a cyclist … or a pedestrian should be detected within 40 pixels from where the cyclist was detected"

Requirements for open world testing

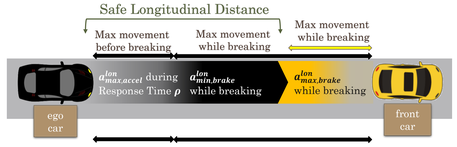
Challenge: How do we avoid obvious bad cases when testing ADS in open world environments?

One possibility: use *Responsibility Sensitive Safety (RSS) Rules developed by Intel Mobileye* to capture safe driver behavior for automated driving systems.*



Example: Longitudinal Safety Requirements in STL*

$$\varphi^{lon}_{resp} \equiv G\left(\left(S^{lon}_{b,f} \wedge X \neg S^{lon}_{b,f}\right) \rightarrow \circ\, \boldsymbol{p^{lon}}\right)$$
$$\boldsymbol{p^{lon}} \equiv \left(S^{lon}_{b,f}\, \bar{\mathcal{R}}_{[0,\rho)}\left(A^{lon}_{b,maxAcc} \wedge A^{lon}_{f,maxBr}\right)\right) \wedge \left(S^{lon}_{b,f}\, \bar{\mathcal{R}}_{[\rho,+\infty)}\left(A^{lon}_{b,minBr} \wedge A^{lon}_{f,maxBr}\right)\right)$$

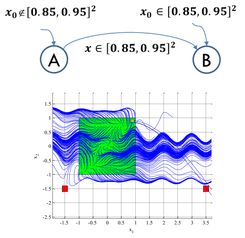


* Shalev-Shwartz et. al., "On a formal model of safe and scalable self-driving cars," arXiv:1708.06374v6, 2018

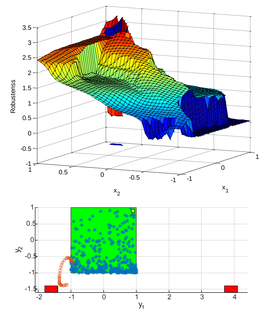
Falsification with model & code coverage

System:
If $\boldsymbol{x_0} \in [-1,1]^2 \backslash [0.85, 0.95]^2$ then
$\dot{x} = x - y + 0.1t$
$\dot{y} = y\cos(2\pi y) - x\sin(2\pi x) + 0.1t$
Else
$\dot{x} = x$
$\dot{y} = -x + y$

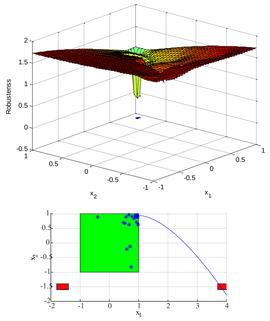
Initial conditions:
[-1,1] x [-1,1]



Specification:
$G_{[0,2]} \neg a \wedge G_{[0,2]} \neg b$
where
O(a) = [-1.6,-1.4] x [-1.6,-1.4]
O(b) = [3.4,3.6] x [-1.6,-1.4]

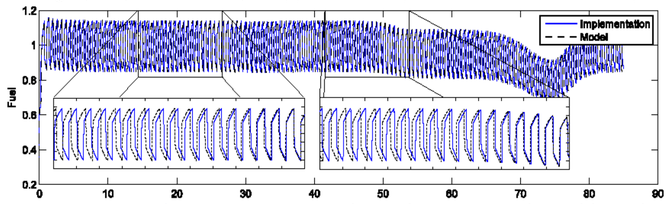


Specification:
$G_{[0,2]} \neg a \wedge G_{[0,2]} \neg b$
where
O(a) = [-1.6,-1.4] x [-1.6,-1.4] x {B}
O(b) = [3.4,3.6] x [-1.6,-1.4] x {B}



A generic conformance notion

In general, determining that the outputs of the Model and the Implementation are "close enough", i.e., conformant, is application-dependent and relies on expertise.



We propose $(T, J, \tau, \varepsilon)$-closeness as a generic conformance notion. This notion is appropriate for continuous-time, discrete-time, and hybrid-time systems.

$(\boldsymbol{T}, \boldsymbol{J}, \boldsymbol{\tau}, \boldsymbol{\varepsilon})$-closeness: Consider two trajectories $\boldsymbol{y}$, and $\boldsymbol{y}'$ of Σ and Σ', respectively. Given $T > 0$, $J > 0, \tau > 0$, and $\varepsilon > 0$, we say $\boldsymbol{y}$ and $\boldsymbol{y}'$ are $(T, J, \tau, \varepsilon)$–close if:

For all (t, j) in the support of $\boldsymbol{y}$ s.t. $t \le T$ and $j \le J$, there exists (s, j) in the support of $\boldsymbol{y}'$, such that $|t - s| < \tau$ and $||y(t,j) - y'(s,j)|| < \varepsilon$ (and the symmetric notion)

The largest (τ, ε) such that all trajectories of Σ and Σ' are $(T, J, \tau, \varepsilon)$–close is the conformance degree between Σ and Σ'.

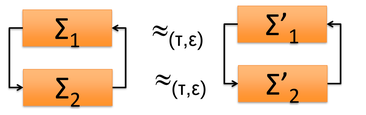
Property transfer between (τ, ε)-close systems

Model 1 $y_1 = H_1(x_1, u)$ $\quad \preccurlyeq_{\tau,\varepsilon}$ Model 2 $y_2 = H_2(x_2, u)$

$H_1 \models \varphi$'? $\qquad$ $H_2 \models \varphi$

Theorem: If $H_1 \preccurlyeq_{\tau,\varepsilon} H_2$ and $H_2 \models_O \varphi$, then $H_1^{\tau} \models_{O_\varepsilon} [\varphi]_\tau$

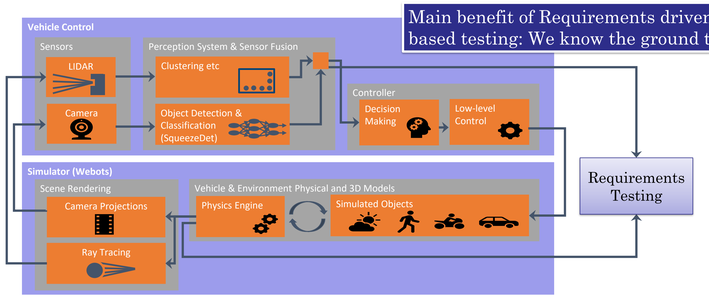
Compositionality



$\varphi \Rightarrow \quad \models \varphi'$

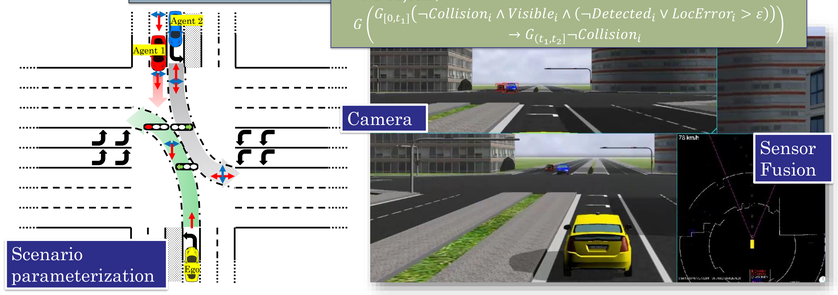
Auto-Test Generation for Perception in CPS

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



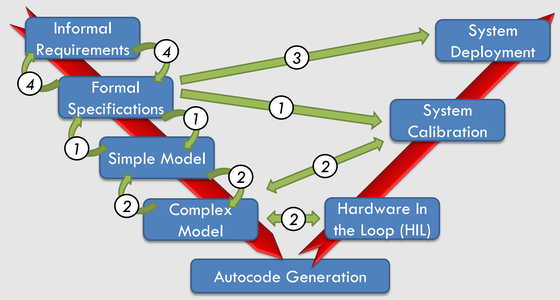
Req. 4 Search for a case where an object is visible, but it is not detected or its localization error is greater than ε and then there is a collision with the Ego.

$$\forall i \in \text{Objects}, G\left(G_{[0,t_1]}(\neg Collision_i \wedge Visible_i \wedge (\neg Detected_i \vee LocError_i > \varepsilon)) \rightarrow G_{(t_1,t_2]} \neg Collision_i\right)$$



Req. 2: Sensor should detect visible objects within a certain time.	$\forall i \in \text{Objects}, G\left((Visible_i \wedge \neg Detected_i) \rightarrow F_{[0,t_1]}(Visible_i \vee Detected_i)\right)$
Req. 3: Object localization error should not be too large for too long.	$\forall i \in \text{Objects}, G\left((Visible_i \wedge (\neg Detected_i \vee LocError_i > \varepsilon)) \rightarrow F_{[0,t_1]}(Visible_i \vee (Detected_i \wedge LocError_i \le \varepsilon))\right)$
Req. 5: The vehicle should not do excessive braking unnecessarily for too long or too often.	$G\left(G_{[0,t_1]}(HardBrake \wedge \neg Collision) \rightarrow \neg\left(BrRelease \wedge F_{(0,t_2]}(BrRelease \wedge F_{(0,t_2]} BrRelease)\right)\right)$

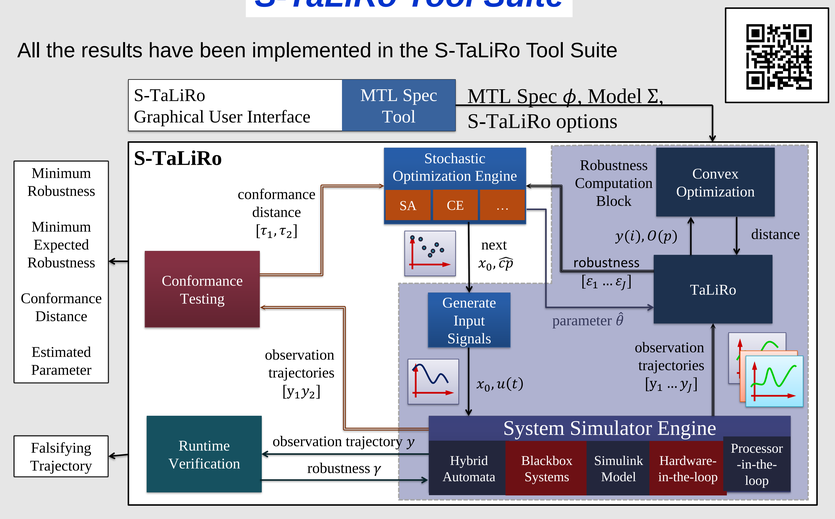
Vision: Supporting MBD at all stages



1. Testing formal specifications and specification mining
2. Conformance testing: models, HIL/PIL or tuned/calibrated model
3. Runtime monitoring of formal requirements
4. Specification analysis, visualization and debugging (inconsistencies, redundancy, etc.)

S-TaLiRo Tool Suite

All the results have been implemented in the S-TaLiRo Tool Suite



More information and publications: www.tinyurl.com/Staliro