Temporal Logic Testing and Verification for Cyber-Physical Systems

Georgios Fainekos

School of Computing, Informatics and Decision System Engineering
Arizona State University

fainekos at asu edu

http://www.public.asu.edu/~gfaineko
What do all these pictures have in common?

1. All of these are safety-critical Cyber-Physical Systems
2. Amazing engineering accomplishments!
Why so much urgency in CPS science/engineering?

*Paraphrased from Jeannette Wing’s quote

Can we really trust our lives on the current CPS?*
(designed with the current tools and practices)
What do all these pictures have in common?

Amazing engineering accomplishments!

Patriot Missile Incident (1991): Floating point issues

Therac-25 (1985-87): Computerized radiation therapy; Concurrency issues; human lives were lost

Near Earth Asteroid Rendezvous (1998): Problematic interaction between cyber and physical systems, missing software

Demonstration of Autonomous Rendezvous Technology (2005): Problematic interaction between cyber and physical systems, missing software

LAX (2004): Voice communication system crashed

BATS Global Markets (2012): software bug paused trading

2003 Northeast blackout: distributed systems issues; cost estimate $6.8 B - $10.3 B

Knight Capital (2012): automated trading system went erratic; cost $440,000,000

Ariane 5 (1996): Problematic interaction between cyber and physical systems; software that shouldn’t be there; cost about $500,000,000

Mars Pathfinder (1997): real-time embedded system issues

What ... this is just a small sample of systems with software bugs! 😞
• "A software error may prevent the transmission from downshifting, such as shifting from 5th to 4th gear when coasting," said NHTSA in its recalls summary of the problem. "This may result in decreased engine RPMs and possible engine stall, increasing the risk of a crash."

• ... the software that "allows the ECU to establish a 'handshake' with the engine is in error. The ECU monitors certain driving conditions, and when the engine is found to be out of tolerance, the software picks up an anomaly. When this happens, the ECU triggers a fault code. As the ECU tries to find an optimal driving condition outside its prescribed tolerances, a rough idle or stalling situation ensues."

• ... to update the software that controls the hybrid electric motor. Under certain circumstances, it is possible, according to the company, "...for the electric motor to rotate in the direction opposite to that selected by the transmission."

• If the fault occurs, cruise control can only be disabled by turning off the ignition while driving – which would mean a loss of some control and in many cars also disables power steering. Braking or pressing the cancel button will not work.

• ...
Unfortunately, these types of problems are only going to be more prevalent in our daily lives ...

Software and hardware are becoming more complex and development cycles are becoming shorter and shorter every year ...
Software complexity is only going to increase ...

it is a *general* trend
(all aerospace software – military, commercial, space)

[Source: Gerard Holzmann (NASA JPL)]
Where CPS Differs from the traditional embedded control systems problem:

**The traditional control problem:**
Design control laws such that a dynamical system is stable and/or tracks a desired trajectory optimally (under various forms of noise and inaccuracies).

**The traditional embedded systems problem:**
Embedded software is software on computers with limited resources. The challenge is on coping with limited resources and extracting performance.

**The CPS problem:**
Computation and networking integrated with physical processes. The technical problem is managing dynamics, time, and concurrency in networked computational + physical systems.

[Adopted and adapted from Lee & Seshia]
Why CPS design is hard?
Why is the problem challenging?

Near Earth Asteroid Rendezvous (NEAR)

\[
\frac{d\omega}{dt} = - (\omega + C_v)x + J(\omega C_v - C_v \frac{dv}{dt}) + u
\]

\[
\frac{d\varepsilon}{dt} = 1/2(\varepsilon x + \eta \omega)
\]

\[
\frac{d\eta}{dt} = - 1/2 \varepsilon T \omega
\]

Control laws
\[ u = h(x, w) \]

Autonomous system code

High-level “smart” system logic

“Continuous” system dynamics

“Discrete” system dynamics

Cyber-Physical System
Near Earth Asteroid Rendezvous (NEAR)*

- Dec. 20, 1998: 3 years en route to 433 Eros
- Executes a 15min main engine burn to place vehicle in orbit about the asteroid
- The software detects a transient in the lateral acceleration that exceeded the coded bounds in the software
  - the mechanical system could sustain the forces

Near Earth Asteroid Rendezvous (NEAR)

• The software shuts down the engine and uses thrusters to place NEAR in an earth-safe attitude

• After the thrusters, the software had to switch to reaction wheels for attitude control
  • Code for transition from thrusters to reaction wheels was missing!
  • The momentum was high so wheels were spinning faster than the limits set in software ⇒ wheels were ignored in the computation

• etc etc

• Bottom line: most of the mission’s fuel was wasted before earth gets control again! Mission was completed 13 months later.
Model Based Design to the Rescue...

**Modeling**

\[
\frac{Jd\omega}{dt} = -(\omega + Cv)xJ(\omega + Cv) + J(\omega x Cv - Cd\nu/dt) + u \\
\frac{d\varepsilon}{dt} = \frac{1}{2}(\varepsilon x \omega + \eta \omega) \\
\frac{d\eta}{dt} = -\frac{1}{2} \varepsilon T \omega
\]

When orbit maneuver is executed, then acceleration on the z axis should not exceed 0.1

**Control specifications**

High-level "smart" system logic

Unfortunately (despite the substantial progress) we are not there yet! 😞
In general, verifying a hybrid system is an undecidable problem!

What can be done?

- Previous approaches to the problem:
  - identifying decidable classes
  - Theorem proving (barrier certificates/invariants)
  - reachability algorithms
  - robust testing
  - systematic simulations / model based testing
  - statistical techniques

This talk ...
Overview

Motivation

Specification Language
- Metric Temporal Logic

Black-Box Testing
- Temporal Logic Testing
- Robust Temporal Logic
- Temporal Logic Robustness Guided Testing
- Simulated Annealing with Monte-Carlo sampling
- Temporal Logic Parameter Estimation

White\Gray Box Testing
- Gradient descent
- Local search using Robust Testing

Numerical Experiments

Conclusions & Future Research
“A design without specifications cannot be right or wrong, it can only be surprising!”*

**Challenge:** Force engineers to actually write formal requirements in any specification language

Powertrain Problem*

6 state var.

Simulink® Checkmate (CMU) model

Specification: For constant throttle and road grade the vehicle should not switch from gear 2 to gear 1 to gear 2

Metric Temporal Logic (MTL)

**Syntax:**

\[ \Phi ::= T \mid \bot \mid p \mid \neg p \mid \Phi_1 \lor \Phi_2 \mid \Phi_1 \land \Phi_2 \mid \Phi_1 \triangleright I \Phi_2 \mid \Phi_1 \triangleright I \Phi_2 \]

* I can be of any bounded or unbounded interval of \([0,+)\), but \(I \neq \emptyset\)
  
  i.e. \(I = [0,+)\), \(I = [2.5,9.8]\)

**Derived operators:**

- Eventually (in the future) \(F_I \Phi := T \triangleright I \Phi\)
- Always (globally) \(G_I \Phi := \bot \triangleright I \Phi\)

Koymans '90, Specifying real-time properties with metric temporal logic
LTL intuition

\( a - a \text{ now} \)

\( \mathbf{G} a - \text{always } a \)

\( \mathbf{F} a - \text{eventually } a \)

\( \mathbf{X} a - \text{next state } a \)

\( a \mathbf{U} b - a \text{ until } b \)

\( a \mathbf{B} b - a \text{ before } b \)
MTL: An example for signals

![Diagram of signals with time and Boolean abstraction](image-url)
MTL: More complicated examples

- $F_I a$
- $G_{I'} \neg a$
- $(\neg a) \mathcal{U}_I b$

Boolean abstraction
Formalizing Complex Specifications

1. Find values for the initial parameters such that starting from 0 speed, the gear transitions from second to first to second.

   \[ \varphi_1 = \neg F(\text{gear}_2 \land F(\text{gear}_1 \land F\text{gear}_2)) \]

2. A more “useful” property is to find constrain the gear change from second to first to second not happen within 2.5 sec.

   \[ \varphi_2 = G((\neg \text{gear}_1 \land X \text{gear}_1) \rightarrow G_{[0,2.5]} \neg \text{gear}_2) \]

3. Verify that the jitter is within acceptable limits

   \[ \varphi_3 = G(\text{gear}_{21} \rightarrow b) \]
Overview

Motivation

Specification Language
- Metric Temporal Logic

Black-Box Testing
- Temporal Logic Testing
- Robust Temporal Logic
- Temporal Logic Robustness Guided Testing
- Simulated Annealing with Monte-Carlo sampling
- Temporal Logic Parameter Estimation

White/Gray Box Testing
- Gradient descent
- Local search using Robust Testing

Numerical Experiments

Conclusions & Future Research
Temporal Logic Testing

\[ \Phi = G p_1 \land F_{[0,T]} G p_2 \]

Monitoring Algorithm

[\text{Maler and Nickovic '04}]
[\text{Thati and Rosu '04}]
[\text{Rosu and Havelund '05}]
[\text{Geilen '01}]
others …
Motivating Example: Automatic Transmission Simulink Demo

Remark: Throttle percentage is the user input

Specification: The following 2 conditions should not occur during the first 60 sec of system operation:
1. the vehicle speed $v$ exceeds 120km/h, and
2. the engine speed $\omega$ exceeds 4500RPM
Automatic Transmission Example

- Throttle
- RPM
- Speed
What test methods are out there?

Assume constant inputs

Spec: \( \neg (F(v \geq 120\text{km/h}) \land F(\omega \geq 4500\text{RPM})) \)

Systematic testing

Monte Carlo testing
Both approaches manage to falsify the specification

Systematic testing

Monte Carlo testing

What if either method cannot find a bad trajectory? We may only be able to provide statistical guarantees, but no other insight. See for example “Statistical Model Checking”.
Overview

Motivation

Specification Language
- Metric Temporal Logic

Black-Box Testing
- Temporal Logic Testing
- Robust Temporal Logic
  - Temporal Logic Robustness Guided Testing
  - Simulated Annealing with Monte-Carlo sampling
  - Temporal Logic Parameter Estimation

White\Gray Box Testing
- Gradient descent
- Local search using Robust Testing

Numerical Experiments

Conclusions & Future Research
Two signals that satisfy the same spec, but ...

MTL Spec:
\[ G(p_1 \rightarrow F_{\leq 2} p_2) \]
Robustness of Temporal Logics

LTL / MTL
\( \Phi = G(p_1 \rightarrow F_{\leq 2} p_2) \)

Monitor/Tester

Robustness parameter
\( \varepsilon \in \mathbb{R} \cup \{ \pm \infty \} \)

Fainekos and Pappas, Robustness of temporal logic specifications for continuous-time signals, Theoretical Computer Science, 2009
**Theorem:** Let $\Phi$ be an MTL formula, $s$ be a (continuous or discrete time) signal and $|\varepsilon|>0$ be the *robustness parameter* of $\Phi$ with respect to $s$, then for all $s'$ in $B_\rho(s,\varepsilon)$ we have that $s \models \Phi$ iff $s' \models \Phi$.

---

*Fainekos and Pappas, Robustness of temporal logic specifications for continuous-time signals, Theoretical Computer Science, 2009*

*Abbas et al., Probabilistic Temporal Logic Falsification of Cyber-Physical Systems, ACM TECS 2013*
(Signed) Distance

Let $x \in X$ be a point, $C \subseteq X$ be a set and $d$ be a generalized quasi metric. Then we define

\[
\text{dist}_d(x,C) = \inf \{ d(x,y) \mid y \in C \} \\
\text{depth}_d(x,C) = \inf \{ d(x,y) \mid y \in X \setminus C \}
\]

\[
\text{Dist}_d(x,C) = \begin{cases} 
-\text{dist}_d(x,C) & \text{if } x \not\in C \\
\text{depth}_d(x,C) & \text{if } x \in C 
\end{cases}
\]
But the requirements are not only on the continuous state space ...

e.g. the vehicle speed should not exceed 120km/h only in third gear

How do we define distances and robustness?
But the requirements are not only on the continuous state space ...

Concurrent FSM

Define “hybrid” robustness values: $(\varepsilon, \delta)$

Distance on the graph

Distance on the continuous state
Discrete-time Robust Semantics for MTL

\[
[c]_D(\mu, i) := c
\]
\[
[p]_D(\mu, i) := \text{Dist}_d(\sigma(i), \mathcal{O}(p))
\]
\[
[\neg \phi_1]_D(\mu, i) := -[\phi_1]_D(\mu, i)
\]
\[
[\phi_1 \lor \phi_2]_D(\mu, i) := [\phi_1]_D(\mu, i) \cup [\phi_2]_D(\mu, i)
\]
\[
[\phi_1 U_I \phi_2]_D(\mu, i) := \bigcup_{j \in \tau^{-1}(\tau(i) + RI)} \left( [\phi_2]_D(\mu, j) \cap \bigcap_{i \leq k < j} [\phi_1]_D(\mu, k) \right)
\]

Algorithm I
- Based on formula re-writing
- Suitable for runtime monitoring algorithms
- Details Fainekos & Pappas, TCS 2009

Algorithm II
- Based on dynamic programming
- Suitable for offline testing
- MTL formulas: \(O(|\phi| \cdot |\tau| \cdot c)\), where \(c = \max_{0 \leq j \leq |\tau|, I \in T(\phi)} |[j, \max J(j, I)]|\)
- Details Fainekos et al ACC 2012

Algorithms adopted and adapted from prior results by Thati, Rosu and Havelund
Running time* in sec

*Laptop Intel i7 2.7GHz 8GB RAM

| Specification | |τ| |
|---------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| G(p₁ → F ¬p₁) | 0.029925 | 0.244558 | 0.465588 | 0.705208 | 0.915293 |
| G(p₁ → F_{5,∞} G_{5,∞} ¬p₁) | 0.040668 | 0.38562 | 0.735401 | 1.045253 | 1.53955 |
| G(p₁ → F_{[0,5]} G_{[0,5]} ¬p₁) | 0.049375 | 0.512974 | 0.948382 | 1.400167 | 1.851488 |
| ¬F(g₂ ∧ F(g₁ ∧ Fg₂)) | 0.0447 | 0.35336 | 0.671639 | 0.988415 | 1.308291 |
| G((¬g₁ ∧ X g₁) → G_{[0,2.5]} ¬g₂) | 0.045691 | 0.467144 | 0.920518 | 1.323044 | 1.692781 |
| G(gear₂₁ → b) | 0.020956 | 0.191298 | 0.33604 | 0.481997 | 0.648962 |
Overview

Motivation

Specification Language
- Metric Temporal Logic

Black-Box Testing
- Temporal Logic Testing
- Robust Temporal Logic
- Temporal Logic Robustness Guided Testing
- Simulated Annealing with Monte-Carlo sampling
- Temporal Logic Parameter Estimation

White\Gray Box Testing
- Gradient descent
- Local search using Robust Testing

Numerical Experiments

Conclusions & Future Research
Verifying Temporal Logic Properties

System $\Sigma$

$\sigma = \Delta(x_0, u) \quad x_0 \in X_0 \quad u \in U$

$LTL / MTL$

$\Phi = G p_1 \land F_{[0, T]} G p_2$

Verifier

Yes

No

Counterexample for debugging?
Temporal Logic falsification as robustness minimization

Spec : $Fp$

$B_\rho(s, \varepsilon) \subseteq L(\Phi)$
Testing Temporal Logic Robustness

System $\Sigma$

$$y = \Delta(x_0, u) \quad x_0 \in X_0 \quad u \in U$$

LTL / MTL

$$\Phi = G p_1 \land F_{[0,T]} G p_2$$

Optimizer

$$\delta = \inf_y \text{Dist}_\rho(y, \mathcal{L}(\Phi)) = ?$$

if $\delta < 0$, then system can be falsified

Minimizing Temporal Logic Robustness

• We need to solve an optimization problem:

\[
\min \text{Dist}_\rho(y, \mathcal{L}(\Phi)) \\
y \in Y \text{ is the set of all observable trajectories of the hybrid system}
\]

• Challenges:
  • Non-linear system dynamics
  • Unknown input signals
  • Unknown system parameters
  • Non-differentiable cost function
    • not known in closed form
    • needs to be computed

Example of Robustness Landscape

System:
\[
\frac{dx}{dt} = x - y + 0.1t \\
\frac{dy}{dt} = y\cos(2\pi y) - x\sin(2\pi x) + 0.1t
\]

Initial conditions:
\([-1, 1] \times [-1, 1]\)

Specification:
\[G_{[0, 2]} \neg a\]
where \(O(a) = [-1.6, -1.4] \times [-1.1, -0.9]\)
Minimizing Temporal Logic Robustness

• Classical solution to difficult engineering problems: Stochastic optimization algorithms

• S-TaLiRo* supports:
  • Simulated Annealing hit-and-run Sampling
  • Extended Ant Colony Optimization
  • Genetic Algorithms
  • Cross-Entropy methods

Goal: Falsify the system or find non-robust behaviors or the worst possible behavior

Negative robustness  Small positive robustness

* Annapureddy, Liu, Fainekos, Sankaranarayanan, S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems, TACAS '11
S-TaLiRo: Systems TemporAl LogIc ROBustness

Minimum Robustness
Falsifying Trajectory

S-TaLiRo

Convex Optimization

Stochastic Optimization

TaLiRo

Simulink/Stateflow Simulation Engine (or hybrid automata simulator)

MTL spec
Model

https://sites.google.com/a/asu.edu/s-taliro/
Overview

Motivation

Specification Language
- Metric Temporal Logic

Black-Box Testing
- Temporal Logic Testing
- Robust Temporal Logic
- Temporal Logic Robustness Guided Testing
- Simulated Annealing with Monte-Carlo sampling
- Temporal Logic Parameter Estimation

White\Gray Box Testing
- Gradient descent
- Local search using Robust Testing

Numerical Experiments

Conclusions & Future Research
Vector Simulated Annealing

Start with \(x_0 \in X_0\), \(\tau_0 > 0\), \(k = 1\)

Draw candidate \(y_k \sim R(x_{k-1}, B)\)

Accept based on \(V_1\)

\[\begin{align*}
\text{yes} & \quad \tau_k \to 0 \\
\text{no} & \quad \text{in probability } \propto \exp\left[-\frac{V_1(y_k) - V_1(x_{k-1})}{\tau_{k-1}}\right]
\end{align*}\]

Closed and bounded
Positive density

\(x_k = y_k\) with probability

\(\tau_0 > 0\)
Vector Simulated Annealing

Start with $x_0$ in $X_0$, $\tau_0 > 0$, $k = 1$

Draw candidate $y_k \sim R(x_{k-1}, B)$

Accept based on $V_2$

Does it converge to the global minimum of $V$?

$x_k = y_k$ with probability $\propto \exp\left[\frac{V_2(y_k) - V_2(x_{k-1})}{\tau_{k-1}}\right]$

Accept based on $V_2$
If $H$ is “simulatable”, then the hybrid system trajectories are “continuous” with respect to the initial conditions.

**Proposition:** Let $H$ be a hybrid automaton, and let $P$ be the partition of the initial conditions induced by the equivalence relation $x \equiv x_0$ iff $\text{loc}(x) = \text{loc}(x_0)$. Let $S \in P$ be a partition such that $\mu(S) \neq 0$. If $H$ satisfies the conditions for accurate simulation over $S$, then for any $x_0 \in S$ and every $\varepsilon > 0$, there exists $\delta > 0$ with the following property: for every trajectory $\eta_x(.)$ with initial point $x$ in $B(x_0) \cap S$, and every $t < T$, there exists $t'$ such that $|t-t'| < \varepsilon$, $s(x, t')$ and $s(x_0, t)$ are in the same location, and $\|s(x,t')-s(x_0,t)\| < \varepsilon$.

Conditions on the Hybrid System

Consider a hybrid system*: \( H = (C,F,G,D) \)

- \( C \subseteq \mathbb{R}^d \) is called the flow set
- \( F : \mathbb{R}^n \rightarrow \mathcal{P}(\mathbb{R}^n) \) is called the flow map
- \( D \subseteq \mathbb{R}^n \) is called the jump set
- \( G : \mathbb{R}^n \rightarrow \mathcal{P}(\mathbb{R}^n) \) is called the jump map

System dynamics

- \( \dot{\xi} \in F(\xi), \xi \in C \)
- \( \xi^+ \in G(\xi), \xi \in D \)

Conditions on the Hybrid System

Conditions for a system to have accurate simulations*

1. C, D are closed sets
2. $F : \mathbb{R}^n \rightarrow \mathcal{P} (\mathbb{R}^n)$ is outer semi-continuous and locally bounded, and $F(\xi)$ is nonempty and convex for all $\xi \in C$
3. $G : \mathbb{R}^n \rightarrow \mathcal{P} (\mathbb{R}^n)$ is outer semi-continuous and locally bounded, and $F(\xi)$ is nonempty for all $\xi \in D$

- A set-valued map $F : \mathbb{R}^n \rightarrow \mathcal{P} (\mathbb{R}^n)$ is o.s.c. iff for all sequences $(\xi_i) \in \mathbb{R}^n$ converging to $\xi$ and all sequences $(\omega_i) \in F(\omega_i)$ converging to $\omega$, it holds that $\omega \in F(\xi)$.

---

Convergence under conditions

Convergence of traditional SA over continuous domains established by Belisle in 1992 under these conditions:

C0 – V is continuous over its domain Init

C8 – Automaton admits a simulator over a subset S* of Init \( \Rightarrow \) trajectories are continuous in \( x_0 \) over S*
\( \Rightarrow V_2 \) is continuous over S*.

C1 – Init is closed and bounded

Physical parameters are bounded. Closure is a standard assumption.

C2 – V achieves its minimum at some \( x^* \) in Init

Necessary to minimize by simulation. Holds for \( V_1 \), implied by C0 and C1 for \( V_2 \).
Convergence under conditions

C3 - For any $\varepsilon > 0$, $B_\varepsilon(x^*) \cap \text{Init}$ not empty
This allows the SA stochastic sampling to approach $x^*$.

C7 - Measure of $B_\varepsilon(x^*) \cap S^* > 0$ for all $\varepsilon > 0$.

C4 - The selection Markov kernel $R(x, B)$ is absolutely continuous w.r.t. the Lebesgue measure, with density $r(x,y)$ bounded away from 0

C5 - $R(x,B)$ is continuous in $x$ for every open set $B$

C6 - The sequence of temperatures $\tau_k$ converges in probability to 0 for any initial choice $\tau_0$.

We implement the kernel using Hit-and-Run, shown to satisfy C4, C5, and temperatures are chosen to satisfy C6.
**Theorem:** Let \( x_1, x_2, \ldots \) be the sequence of states generated by vector SA when solving the Falsification Problem. Assume that conditions \( C1, C2, C4-C8 \) hold. Let \( V^* \) denote the global minimum of \( V \) on \( X_0 \). Then, for any pair of initial conditions \((x_0, \tau_0)\) for SA, the sequence of function values \((V(x_k)), k \geq 0\), converges in probability to \( V^* \).

Overview

Motivation
Specification Language
- Metric Temporal Logic

Black-Box Testing
- Temporal Logic Testing
- Robust Temporal Logic
- Temporal Logic Robustness Guided Testing
- Simulated Annealing with Monte-Carlo sampling
- Temporal Logic Parameter Estimation

White\Gray Box Testing
- Gradient descent
- Local search using Robust Testing

Numerical Experiments
Conclusions & Future Research
Temporal Logic Parameter Estimation

\[ y = \Delta(x_0, u) \quad x_0 \in X_0, \quad u \in U \]

**CPS \( \Sigma \)**

**LTL / MTL**

\[ \Phi[\tau, \theta] = G(x < -10) \land F_{[0, \tau]} G(x > \theta) \]

**Parameter Estimator**

\[ T = [\tau_1, \tau_2], \quad \Theta = [\theta_1, \theta_2] \]

s.t. \( \forall \tau \in T, \forall \theta \in \Theta, \Sigma \not\models \Phi[\tau, \theta] \)

Minimizing Temporal Logic Robustness

• We need to solve an optimization problem:

\[
\begin{align*}
\text{optimize} & \quad \theta \\
\text{subject to} & \quad \theta \in \Theta \text{ and } \left[ \phi[\theta] \right](\Sigma) = \min_{\mu \in \mathcal{L}_\tau(\Sigma)} \left[ \phi[\theta] \right](\mu) \leq 0
\end{align*}
\]

• Challenges:
  • Non-linear system dynamics
  • Unknown input signals
  • Unknown system parameters
  • Non-differentiable cost function
    • not known in closed form
    • needs to be computed

Yang, Hoxha and Fainekos, Querying Parametric Temporal Logic Properties on Embedded Systems, ICTSS 2012
How does our cost function look like?

Spec: $G_{[0,\theta]}(\omega<4500\text{RPM})$

Throttle % parameterization with 1 variable
Overview

Motivation

Specification Language
  - Metric Temporal Logic

Black-Box Testing
  - Temporal Logic Testing
  - Robust Temporal Logic
  - Temporal Logic Robustness Guided Testing
  - Simulated Annealing with Monte-Carlo sampling
  - Temporal Logic Parameter Estimation

White\Gray Box Testing
  - Gradient descent
  - Local search using Robust Testing

Numerical Experiments
Conclusions & Future Research
Falsification through “descent”

\[
d = A(t^*, x)^{-1} n_s(x)(t^*)
\]

Abbas and Fainekos, Computing Descent Direction of MTL Robustness for Non-Linear Systems, ACC, 2013
Examples

\[ G(\neg\text{red}) \]

\[ G(\text{small\_red} \Rightarrow G_{[0,1]} \neg\text{big\_red}) \]
Reducing Robustness via Gradient Descent

\[ \bar{s}(t; w) = s_{x_0}(t; u) \]

\[ z(t; w) = \arg \min_{z \in \mathcal{O}(p)} \| \bar{s}(t; w) - z \| \]

\[ J(w) = \| z(t_{r,1}; w_1) - \bar{s}(t_{r,1}; w) \| \]

\( t_{r,1} \) is the critical time for \( \bar{s}(t, w_1) \)

*Abbas, Winn, Fainekos and Julius, Functional Gradient Descent Method for Metric Temporal Logic Specifications, ACC 2014*
Reducing Robustness via Gradient Descent

Update $w$ via gradient descent on $J(w)$

If $J(w_2) < J(w_1)$ then robustness is reduced

Overview

Motivation

Specification Language
- Metric Temporal Logic

Black-Box Testing
- Temporal Logic Testing
- Robust Temporal Logic
- Temporal Logic Robustness Guided Testing
- Simulated Annealing with Monte-Carlo sampling
- Temporal Logic Parameter Estimation

White/Gray Box Testing
- Gradient descent
- Local search using Robust Testing

Numerical Experiments

Conclusions & Future Research
Falsification through “descent”: Hybrid systems

Abbas and Fainekos, Linear Hybrid System Falsification Through Local Search, ATVA, 2011
Falsification through “descent” within the robustness neighborhood

Abbas and Fainekos, Linear Hybrid System Falsification Through Local Search, ATVA, 2011
Computing bisimulation functions

Quadratic Bisimulation Functions for Deterministic Linear Systems

\[ \frac{dx}{dt} = Ax \]
\[ y = Cx \]

\[ V(x) = \sqrt{x^T M x} \]
is a bisimulation function if
\[ M \geq C^T C \]
\[ A^T M + M A \leq 0 \]

Approximate Bisimulations for Constrained Linear Systems
Antoine Girard and George J. Pappas

Bisimulation Functions using Sum Of Squares Relaxation

\[ \frac{dx}{dt} = f(x) \]
\[ y = g(x) \]

\[ V(x_1, x_2) = \sqrt{q(x_1, x_2)} \]
is a bisimulation function if
\[ q(x_1, x_2) - \left\| g_1(x_1) - g_2(x_2) \right\|^2 \] is SOS
\[ - \frac{\partial q(x_1, x_2)}{\partial x_1} f_1(x_1) - \frac{\partial q(x_1, x_2)}{\partial x_2} f_2(x_2) \] is SOS

Approximate Bisimulations for Nonlinear Dynamical Systems
Antoine Girard and George J. Pappas

[For more details and possibilities see Tabuada 2009]
Overview

Motivation

Specification Language
- Metric Temporal Logic

Black-Box Testing
- Temporal Logic Testing
- Robust Temporal Logic
- Temporal Logic Robustness Guided Testing
- Simulated Annealing with Monte-Carlo sampling
- Temporal Logic Parameter Estimation

White/Gray Box Testing
- Gradient descent
- Local search using Robust Testing

Numerical Experiments

Conclusions & Future Research
Goals

• Show that stochastic optimization methods (heuristics) work faster than uniform random sampling
  - Simulations of complex models are expensive!

• TL Robustness provides a physical notion of system robustness with respect to formal requirements
  - Boolean based testing does not

• Demonstrate that TL testing & verification provides useful insight to system design
## Experimental Results

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Model Type</th>
<th>Property Type</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mod1-3</td>
<td>Third order Delta-Sigma Modulator [12] with varying initial conditions</td>
<td>S/S diagram</td>
<td>$\square a$</td>
<td>1000</td>
</tr>
<tr>
<td>IG1-3</td>
<td>Insulin Glucose control (Cf. Section 3.2) with varying initial conditions</td>
<td>ODE (matlab function)</td>
<td>$\square [0,20.0] P \land \square [20,200.0] Q$</td>
<td>1000</td>
</tr>
<tr>
<td>AT1</td>
<td>Auto Transmission Simulink demo [37]</td>
<td>S/S diagram</td>
<td>$\neg (\diamond p_1 \land \diamond [0,10] p_3)$</td>
<td>1000</td>
</tr>
<tr>
<td>AT2</td>
<td></td>
<td></td>
<td>$\neg (\diamond (p_1 \land \diamond [0,7.5] p_3))$</td>
<td>1000</td>
</tr>
<tr>
<td>AT3-5</td>
<td></td>
<td></td>
<td>$\neg (\diamond \forall i (q_{1,i} \land \diamond q_{2,i} \land \diamond q_{3,i})$</td>
<td>1000</td>
</tr>
<tr>
<td>PT1</td>
<td>Power train model [9]</td>
<td>Checkmate model [35]</td>
<td>$\neg \diamond (g_2 \land \diamond (g_1 \land \diamond g_2))$</td>
<td>1000</td>
</tr>
<tr>
<td>PT2</td>
<td></td>
<td></td>
<td>$\square ((\neg g_1 \land X g_1) \Rightarrow \square [0,2.5] \neg g_2)$</td>
<td>1000</td>
</tr>
<tr>
<td>Air1</td>
<td>Aircraft model [26]</td>
<td>ODE (matlab function)</td>
<td>$\neg (\square [5,1.5] a \land \diamond [3,4] b)$</td>
<td>500</td>
</tr>
<tr>
<td>Air2</td>
<td></td>
<td></td>
<td>$\neg (\square [0.4] a \land \diamond [3.5,4] d)$</td>
<td>1000</td>
</tr>
<tr>
<td>Air3</td>
<td></td>
<td></td>
<td>$\neg \diamond [1.3] e$</td>
<td>2000</td>
</tr>
<tr>
<td>Air4</td>
<td></td>
<td></td>
<td>$\neg \square [0.5] h$</td>
<td>2500</td>
</tr>
<tr>
<td>Air5</td>
<td></td>
<td></td>
<td>$\neg \square [2,2.5] i$</td>
<td>2500</td>
</tr>
</tbody>
</table>


# Experimental Results

<table>
<thead>
<tr>
<th>Bench.</th>
<th># Runs</th>
<th>Cross Entropy (CE)</th>
<th>Monte Carlo (MC)</th>
<th>Unif. Rand. (UR)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#F</td>
<td>Time (av,lb,ub)</td>
<td>#Rnds (lb,ub)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mod3</td>
<td>100</td>
<td>13</td>
<td>41,[0,50]</td>
<td>[1,10]</td>
</tr>
<tr>
<td>AT1</td>
<td>100</td>
<td>36</td>
<td>102,[43,116]</td>
<td>[5,10]</td>
</tr>
<tr>
<td>AT4</td>
<td>100</td>
<td>97</td>
<td>53,[1,116]</td>
<td>[1,10]</td>
</tr>
<tr>
<td>Air3</td>
<td>100</td>
<td>99</td>
<td>316,[120,984]</td>
<td>[3,20]</td>
</tr>
</tbody>
</table>
# Experimental Results

<table>
<thead>
<tr>
<th>Bench.</th>
<th># Runs</th>
<th>Cross Entropy (CE)</th>
<th>Monte Carlo (MC)</th>
<th>Unif. Rand. (UR)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#F</td>
<td>Time (av, lb, ub)</td>
<td>#Rnds (lb, ub)</td>
</tr>
<tr>
<td>IG1</td>
<td>25</td>
<td>25</td>
<td>47,[3,91]</td>
<td>[1,4]</td>
</tr>
<tr>
<td>IG2</td>
<td>25</td>
<td>25</td>
<td>65,[20,124]</td>
<td>[1,6]</td>
</tr>
<tr>
<td>IG3</td>
<td>25</td>
<td>25</td>
<td>141,[81,286]</td>
<td>[3,7]</td>
</tr>
</tbody>
</table>
## Experimental Results

<table>
<thead>
<tr>
<th>Bench.</th>
<th># Runs</th>
<th>Cross Entropy (CE)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#F Time (av,lb,ub)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#Rnds (lb,ub)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Monte Carlo (MC)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#F Time (av,lb,ub)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unif. Rand. (UR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#F Time (av,lb,ub)</td>
</tr>
<tr>
<td>PT2</td>
<td>25</td>
<td>24 1013,[32,6381]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1,10] dnf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dnf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25 689,[21,2392]</td>
</tr>
</tbody>
</table>
Formalizing Complex Specifications

1. Find values for the initial parameters such that starting from 0 speed, the gear transitions from second to first to second.

   \[ \varphi_1 = \neg F(gear_2 \land F(gear_1 \land F\neg gear_2)) \]

2. A more “useful” property is to find constrain the gear change from second to first to second not happen within 2.5 sec.

   \[ \varphi_2 = G((\neg gear_1 \land X gear_1) \rightarrow G_{[0,2.5]} \neg gear_2) \]

3. Verify that the jitter is within acceptable limits

   \[ \varphi_3 = G(gear_{21} \rightarrow b) \]
Powertrain Example: Specifications

Falsifying $\phi_1$

Falsifying $\phi_2$

Falsifying $\phi_3$

Torque

Shift Schedule

Throttle $\approx 93.9$, 
Grade $\approx 0.2453$
Robustness cost function

\[ \varphi_2 = G(\neg \text{gear}_1 \land X \text{ gear}_1) \rightarrow G_{[0,2.5]} \neg \text{gear}_2) \]
Modified Spec on Enginuity Engine model

\[ \phi_1^S[\lambda] = \Box_{[0,100]}((g_2 \land X g_1) \rightarrow \Box_{(0,2.5]}((\tau \leq 2.5) \rightarrow g_1) \]
I.e. for any parameter \( \geq 0.4273 \), it is guaranteed that the system does not satisfy \( \varphi_2 \).
Overview

Motivation

Specification Language
- Metric Temporal Logic

Black-Box Testing
- Temporal Logic Testing
- Robust Temporal Logic
- Temporal Logic Robustness Guided Testing
- Simulated Annealing with Monte-Carlo sampling
- Temporal Logic Parameter Estimation

White\Gray Box Testing
- Gradient descent
- Local search using Robust Testing

Numerical Experiments

Conclusions & Future Research
S-Taliro support in the V-process

1. Informal Requirements
2. Model Design
3. Formal Specifications
4. System Deployment

Support

Autocode Generation (with multi-core in mind)

Hardware in the Loop (HIL)
System Calibration

System Deployment

Informal Requirements

Formal Specifications
Model Design

S-Taliro support
S-Taliro support in the V-process

1. Testing formal specifications and modifying formal specifications and model
2. Conformance testing between model and HIL/PIL or tuned/calibrated model
3. Testing formal specifications on the HIL/PIL calibrated system
4. Runtime monitoring of formal requirements
S-Taliro Status

Graphical User Interface for S-TaLiRo

MTL Specification Tool

S-TaLiRo

Descent computation

Convex Optimization

TaLiRo

1. Any simulator interfaced with Matlab
2. Hardware and/or processor in the loop
Future S-Taliro support in the V-process

1. Informal Requirements
2. Formal Specifications
3. Model Design
4. System Calibration
5. System Deployment

- Continuous improvement of 1-4
5. Extracting formal requirements for CPS and checking specification inconsistencies
Further in the Future: Automatic Synthesis

Modeling

\[
\begin{align*}
J \frac{d\omega}{dt} &= -(\omega + C_v) x J (\omega + C_v) + J (\omega x C_v - C_v \frac{dv}{dt}) + u \\
\frac{d\varepsilon}{dt} &= \frac{1}{2} (\varepsilon x w + \eta w) \\
\frac{d\eta}{dt} &= -\frac{1}{2} \varepsilon T \omega
\end{align*}
\]

Control specifications

When orbit maneuver is executed, then acceleration on the z axis should not exceed 0.1

High-level “smart” system logic

Control laws

\[ u = h(x,w) \]

Autocode generation
Acknowledgements

Students
• Houssam Abbas - PhD
• Adel Dokhanchi - PhD
• Bardh Hoxha - PhD

Former Students
• Y. Annapureddy - MS
• Hengyi Yang - MS

Recent collaborators
• UC, Boulder: S. Sankaranarayanan
• RPI: Agung Julius
• ASU: Y. Kobayashi, Y-H Lee,
• NEC Labs: A. Gupta, F. Ivancic
• Toyota: J. V. Deshmukh, J. Kapinski, K. Ueda, H. Yazarel

Tools at: https://sites.google.com/a/asu.edu/s-taliro/

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.