Temporal Logic Testing for Hybrid Systems

Georgios Fainekos

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

fainekos at asu edu

http://www.public.asu.edu/~gfaineko
“A design without specifications cannot be right or wrong, it can only be surprising!“*

Motivating Example: Automatic Transmission Simulink Demo

Specification: The following 3 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, and
3. all states are reached in the switching logic

[Zhao, Krogh, Hubbard, IEEE CSM Aug. 2003]
Why is this problem challenging?

Concurrent FSM

- Discrete & Continuous
- Look-up tables
Why is this problem challenging?

Switching guards that depend in inputs

In other models some guards may also depend on initial conditions
Why is this problem challenging?

Nonlinear operations on state variables
In summary

• Nonlinear ODEs with 2 state variables
• 3 look-up tables and 3 look-up 2D tables
• A Stateflow chart: two concurrent FSM with 4 and 3 states

If you would like to get real, in simplified models you will expect ~100 state variables (both continuous & discrete time)
Properties?

**Specification:** The following 3 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, and
3. all states are reached in the switching logic

What if you would like to verify some more complicated property? For example:

- **“when the road grade and the throttle remain constant, then the system should not change from gear 1 to gear 2 and then back to gear 1”**
- **“whenever the system enters the first gear, then it should not enter the second gear within 2.5 sec”**
- **“whenever the system is in transition from gear 2 to gear 1, then the derivative of the torque is less than 450’”**
Linear Temporal Logic Intuition

- \( a \) - a now
- \( G a \) - always a
- \( F a \) - eventually a
- \( X a \) - next state a
- \( a U b \) - a until b
- \( a B b \) - a before b
Metric Temporal Logic: An example

\[ \text{Boolean abstraction} \]

\[ F_I a \]
Automatic Transmission Example

Not:
1. the vehicle speed $v$ exceeds 120km/h and
2. the engine speed $!$ exceeds 4500RPM and
3. all states are reached in the switching logic

$$
\varphi = \neg(\bigwedge_{i=1}^{9} F p_i)
$$

$$
O(p_1) = Q \times [120,\infty) \times R
$$

$$
O(p_2) = Q \times R \times [4500,\infty)
$$

$$
O(p_3) = \{\text{first}\} \times R^2
$$

::
More Complex Specifications

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

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

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

\[ \varphi_3 = G(gear_{21} \rightarrow b) \]
Verifying Temporal Logic Properties

System $\Sigma$

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

$$u \in U$$

LTL / MTL

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

Verifier

Yes

No

Counterexample for debugging?
Back to the Automatic Transmission Example

Stateflow state coverage was verified with Matlab diagnostics tool
What can be done?

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

This talk ...
What test methods are out there?

Robust testing

Random Rapidly exploring Trees

Not easy to test real time specifications using these methods.
What test methods are out there?

Assume constant inputs

\[ u \]

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

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 can only provide probabilistic guarantees, but no other insight.
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*, TCS, 2009
Discrete-time Robust Semantics for MTL

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

Algorithm II
- Based on dynamic programming
- Suitable for offline testing
- Details ACC 2012

\[
\begin{align*}
[c]_D(\mu, i) & := c \\
[p]_D(\mu, i) & := \text{Dist}_d(\sigma(i), \mathcal{O}(p)) \\
[\neg \phi_1]_D(\mu, i) & := -[\phi_1]_D(\mu, i) \\
[\phi_1 \lor \phi_2]_D(\mu, i) & := [\phi_1]_D(\mu, i) \cup [\phi_2]_D(\mu, i) \\
[\phi_1 \mathcal{U}_I \phi_2]_D(\mu, i) & := \bigsqcup_{j \in \tau^{-1}(\tau(i) + R I)} ([\phi_2]_D(\mu, j) \cap \prod_{i \leq k < j} [\phi_1]_D(\mu, k))
\end{align*}
\]

timed trace \( \mu = (\sigma, \tau) \)
Temporal Logic falsification as robustness minimization

\[ B_\rho(s,\varepsilon) \subseteq \mathcal{L}(\Phi) \]
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

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

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

Nghiem, Sankaranarayanan, Fainekos, Ivancic, Gupta, Pappas, Monte-Carlo Techniques for Falsification of Temporal Properties of Non-Linear Hybrid Systems, HSCC 2010
Minimizing Temporal Logic Robustness

• We need to solve an optimization problem:

\[
\min \text{Dist}_\rho(y, \mathcal{L}(\Phi))
\]

Where \( y \in Y \) 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
How does our cost function look like?

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

Throttle % parameterization with 1 variable

Throttle % parameterization with 2 variables
S-TaLiRo: Systems TemporAl LogIc ROBustness

S-TaLiRo

Convex Optimization

TaLiRo

Stochastic Optimization

Simulink/Stateflow Simulation Engine (or hybrid automata simulator)

Minimum Robustness

Falsifying Trajectory

Robustness ε

next $x_0$, $u(t)$, $w$

observation trajectory $y$

MTL spec

Model

https://sites.google.com/a/asu.edu/s-taliro/
Minimizing Temporal Logic Robustness

- Classical solution to difficult engineering problems: Stochastic optimization algorithms

- S-TaLiRo* supports:
  - Monte-Carlo (Simulated Annealing)
  - 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
Why Cross-Entropy?

• Main point: Sound underlying theory
  • Important properties, e.g., convergence, can be studied

• Way fewer parameters than other stochastic optimization heuristics
  • Very important: It is not be possible to tune parameters on industrial size models

• Industrial size models can take several seconds to simulate
  • We need stochastic optimization algorithms that can be readily parallelized and actually achieve better performance

• Get for free optimization of expected value of noisy cost functions
  • Handle stochastic hybrid systems
Cross Entropy (CE) Method*

Consider the optimization problem:

\[
\min \ \{\text{Dist}_\rho(y, L(\Phi)) \mid y = \Delta(x_0,u), \ x_0 \in X_0, \ u \in U\}
\]

remark: \(U\) is the set of all possible input signals

and its associated stochastic problem:

\[
P\{\text{Dist}_\rho(\Delta(X,V), L(\Phi)) \leq \varepsilon\} = \mathbb{E}_{\{\text{Dist}_\rho(\Delta(X,V), L(\Phi)) \leq \varepsilon\}}\]

where \(X, V\) are random vectors with pdf \(f(.,\theta)\).

*For falsification, we need to estimate \(p = P\{\text{Dist}_\rho(\Delta(X,V), L(\Phi)) \leq 0\}\)

Cross Entropy Method

Challenge: $\text{Dist}_\rho(\Delta(X,V), L(\Phi)) \leq \varepsilon$ is a rare event

Solution: Generate a sequence of pdfs $f(., \theta_0), f(., \theta_1), f(., \theta_2), ...$
using the Kullback-Leibler divergence

$$D(\Omega, f(., \theta)) = \int \log \left( \frac{\Omega(x)}{f(x, \theta)} \right) \Omega(x) dx$$

between the distribution $\Omega$ defined as

$$\Omega(x_0, u) = 1/W \exp(-K \text{Dist}_\rho(\Delta(x_0, u), L(\Phi)))$$

and $f(., \theta_i)$. 
Cross Entropy Method

Use $n_b$ best samples to update $\mu$ and $\sigma$

E.g. Assume normal distribution

Parameters: $\mu$ and $\sigma$
Cross Entropy Method

Repeat until $\sigma$ is small enough
Cross Entropy Algorithm
Example: update rules for Normal distribution

1. Choose some $\theta_0$ and set $i=0$
   - E.g. $\theta_0 = (\mu_0, \sigma_0)$ for $N(\mu_0, \sigma_0)$

2. While stopping criterion is not met
   - E.g. while $||\sigma_i||_\infty > \delta$
     1. $i = i + 1$
     2. Generate samples $X_1, ..., X_n$ according to pdf $f(., \theta_{i-1})$
        - E.g., Assume independent components in $X$ and $N(\mu_{i-1}, (\sigma_{i-1})^2)$
     3. Use $n_b \leq n$ best samples to update $\theta_i$
        - E.g. $\mu_{ij} = \Sigma_k X_{kj}/n_b$ and $\sigma_{ij} = (\Sigma_k (X_{kj} - \mu_{ij})/n_b)^{1/2}$
     4. Smooth
        - E.g. $\mu_i = a\mu_i + (1-a)\mu_i$ and $\sigma_i = a\sigma_i + (1-a)\sigma_i$

Update rules for Piecewise Uniform Distributions

• Important in the case of constraint optimization
  • Well suited for search spaces that are hyper-rectangles
  • Well suited for correlated variables

• Split each search interval $C_j$ into a number of cells $m$
• Associate a probability $P_{jk}$ with each cell $C_{jk}$
• Then, the update rule becomes

$$P_{jk} = \frac{\sum_{l=0}^{n_b-1} \gamma_l \{x_{lj} \in C_{jk}\}}{\sum_{l=0}^{n_b-1} \gamma_l}$$

where

$$\gamma_l = \frac{W \Omega(x_l)}{f(x_l, \theta_i)}$$
Example: Insulin infusion

\[
\begin{align*}
\frac{dG}{dt} &= -p_1 G - X(G + G_b) + P(t) \\
\frac{dX}{dt} &= -p_2 X + p_3 I \\
\frac{dI}{dt} &= -n(I + I_b) + u(t)/V_I \\
\end{align*}
\]

\[\varphi : \neg (\square_{[0,20.0]}(G \in [-2, 10]) \land \square_{[20,200.0]}(G \in [-1, 1]))\]
# 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>□_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>□<em>[0,20.0]p ∧ □</em>[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>¬(□p_1 ∧ □[0,10]p_3) ¬(□(p_1 ∧ □[0,7.5]p_3)) ¬(□q_1,i ∧ □q_2,i ∧ □q_3,i)</td>
<td>1000</td>
</tr>
<tr>
<td>AT2</td>
<td>different predicates q_{i,j}</td>
<td></td>
<td></td>
<td>1000</td>
</tr>
<tr>
<td>AT3-5</td>
<td></td>
<td></td>
<td></td>
<td>1000</td>
</tr>
<tr>
<td>PT1</td>
<td>Power train model [9]</td>
<td>Checkmate model [35]</td>
<td>¬(□(g_2 ∧ □(g_1 ∧ □g_2)) ⊨(¬g_1 ∧ Xg_1) ⊨□[0,2.5]¬g_2)</td>
<td>1000</td>
</tr>
<tr>
<td>PT2</td>
<td></td>
<td></td>
<td></td>
<td>1000</td>
</tr>
<tr>
<td>Air2</td>
<td></td>
<td></td>
<td></td>
<td>1000</td>
</tr>
<tr>
<td>Air3</td>
<td></td>
<td></td>
<td></td>
<td>2000</td>
</tr>
<tr>
<td>Air4</td>
<td></td>
<td></td>
<td></td>
<td>2500</td>
</tr>
<tr>
<td>Air5</td>
<td></td>
<td></td>
<td></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>#Fds</td>
</tr>
<tr>
<td>IG3</td>
<td>25</td>
<td>25</td>
<td>141, [81, 286]</td>
<td>0</td>
</tr>
<tr>
<td>Mod1</td>
<td>100</td>
<td>97</td>
<td>15, [1, 43]</td>
<td>84</td>
</tr>
<tr>
<td>Mod2</td>
<td>100</td>
<td>87</td>
<td>25, [1, 43]</td>
<td>58</td>
</tr>
<tr>
<td>Mod3</td>
<td>100</td>
<td>13</td>
<td>41, [0, 50]</td>
<td>21</td>
</tr>
<tr>
<td>AT1</td>
<td>100</td>
<td>36</td>
<td>102, [43, 116]</td>
<td>51</td>
</tr>
<tr>
<td>AT2</td>
<td>100</td>
<td>0</td>
<td>111, [97, 114]</td>
<td>0</td>
</tr>
<tr>
<td>AT3</td>
<td>100</td>
<td>99</td>
<td>34, [0, 114]</td>
<td>93</td>
</tr>
<tr>
<td>AT4</td>
<td>100</td>
<td>97</td>
<td>53, [1, 116]</td>
<td>94</td>
</tr>
<tr>
<td>AT5</td>
<td>100</td>
<td>0</td>
<td>111, [98, 121]</td>
<td>0</td>
</tr>
<tr>
<td>PT2</td>
<td>25</td>
<td>24</td>
<td>1013, [32, 6381]</td>
<td>dnf</td>
</tr>
<tr>
<td>Air1</td>
<td>100</td>
<td>100</td>
<td>8, [0, 29]</td>
<td>100</td>
</tr>
<tr>
<td>Air2</td>
<td>100</td>
<td>100</td>
<td>134, [3, 386]</td>
<td>100</td>
</tr>
<tr>
<td>Air3</td>
<td>100</td>
<td>99</td>
<td>316, [120, 984]</td>
<td>78</td>
</tr>
<tr>
<td>Air4</td>
<td>100</td>
<td>100</td>
<td>2, [0, 10]</td>
<td>100</td>
</tr>
<tr>
<td>Air5</td>
<td>100</td>
<td>100</td>
<td>10, [0, 53]</td>
<td>100</td>
</tr>
</tbody>
</table>
Powertrain benchmark*

Powertrain Example: Specifications

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

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

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 $\varphi_1$

Falsifying $\varphi_2$

Falsifying $\varphi_3$

Throttle \( \cong 93.9 \),
Grade \( \cong 0.2453 \)
Robustness cost function

\[ \phi_2 = G((\neg \text{gear}_1 \land X \text{gear}_1) \rightarrow G_{[0,2.5]} \neg \text{gear}_2) \]
Satisfying $\varphi_2$
Throttle $\cong 93.9$, Grade $\cong 0.24$
Falsifying $\varphi_2$

Throttle $\approx 93.9$, Grade $\approx 0.2453$
Satisfying $\varphi_2$
Throttle $\approx 93.9$, Grade $\approx 0.25$
Conclusions

• Functional verification of arbitrary hybrid systems remains a challenging problem

• Robustness minimization for functional falsification offers a practical alternative

• Future work:
  • Study convergence properties
  • Demonstrate applicability over stochastic hybrid systems
Acknowledgements

Students
• Houssam Abbas - PhD (ECEE)
• Bardh Hoxha - PhD (CIDSE)
• Hengyi Yang - MS (CIDSE)

Former Students
• Y. Annapureddy - MS (CIDSE)
• Che Liu - MS (CIDSE)

Recent collaborators
• Aarti Gupta (NEC Labs)
• Franjo Ivancic (NEC Labs)
• Truong Nghiem (UPenn)
• George Pappas (UPenn)
• Sriram Sankaranarayanan (U of Colorado)
• Koichi Ueda (Toyota)
• Hakan Yazarel (Toyota)

Sponsors

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