Automatic Trace Generation for Signal Temporal Logic

Pavithra Prabhakar    Ratan Lal    James Kapinski

Department of Computer Science
Kansas State University

39th IEEE Real-Time Systems Symposium (RTSS)
December 13, 2018
Cyber Physical Systems and Requirement Engineering
Cyber Physical Systems

- Software controlled physical systems (cyber-physical systems (CPS))
  - Automotive cruise control, autonomous robots, and medical devices

\[ \begin{align*}
  \dot{x} &= f(x, u) \\
  y &= h(x) \\
  u &= g(y)
\end{align*} \]

- Model-based design is a popular paradigm for cyber physical system design

Cyber Physical Systems
Systems with mixed continuous and discrete behaviors
Collecting requirements is the first crucial step in model-based design. Requirements are used to develop control system models and specify expected behaviors of models. Quality/correctness of design depends on the quality of the requirements.
Formal Specification

- Formal specifications play a crucial role towards checking inconsistencies/redundancies in requirements
- Needed to exploit formal verification/synthesis techniques
Challenges towards Formal Specification

- Under-specified or over-specified requirements
- Co-design of the physical environment
- Complex requirements including timing constraints and real valued
- Need experts for writing formal specifications
Validating Formal Specification

How to assist control engineers to write correct formal specifications?

Overview of our framework
Signal Temporal Logic
Signal Temporal Logic (STL)

- Specify properties of reactive systems, specially cyber physical systems
- Linear temporal logic (LTL) expresses properties about temporal ordering of behaviors
- STL extends LTL with real timing and real valued constraints
- Example: LTL formula
  - Request-grant specification: \( \Box [r \Rightarrow \Diamond g] \)
    - \( \Box \) - Always
    - \( \Diamond \) - Eventually
  - At any step if there is a request, then that will be eventually granted
- Example: STL formula
  - Value High specification: \( \Box_{[0,5]}(out < 10) \)
  - Always in the interval \([0, 5]\), output signal value is less than 10
**Specification:** Always in the interval $[0, 5]$, the output signal value is less than 10

$$\text{Value}_{\text{High}} = \left[ \square_{[0,5]}(\text{out} < 10) \right]$$

**Satisfying Signal**

**Violating Signal**
Specification: Steady State Error High

**Specification:** At any point in the interval [0,10] if there is no step in the input signal for 2 time units, then the output signal will be less than 5% more than the input signal.

\[
\text{SS}_{\text{Error High}}[\square_{[0,10]}(\square_{[0,2]}(\neg \text{Step}) \Rightarrow \square_{[2,2]}(\text{SS}_{\text{Error Under Limit}}))] \\
\text{SS}_{\text{Error Under Limit}} = [\text{out} < 1.05\text{in}]
\]

---

**Satisfying Signal**

**Violating Signal**
**Specification: Rise Time**

**Specification:** At any point in the interval $[0, 10]$, if there is a step up in the input signal, the output signal will be greater than 90% of the input signal at some point within the interval $[1, 2]$.

$$\text{Rise Time} = \Box_{[0,10]}(\text{Step Up} \implies \Diamond_{[1,2]}(\text{Over Rise Limit}))$$

$$\text{Over Rise Limit} = [\text{out} > 0.9\text{in}]$$
**Specification: Overshoot**

**Specification:** At any point in the interval $[0, 10]$, if there is step up in the input signal, the output signal will be less than 10% more than the input signal at all points within the interval $[0, 2]$.

\[
\text{Overshoot} = \left[ \square_{[0,10]}(\text{Step}_\text{Up} \Rightarrow \square_{[0,2]}(\text{Overshoot}_\text{Limit})) \right] \\
\text{Overshoot}_\text{Limit} = [\text{out} < 1.1\text{in}]
\]
**Specification: Overshoot**

**Specification:** At any point in the interval [0, 10], if there is step up in the input signal, the output signal will be less than 10% more than the input signal at all points within the interval [0, 2].

\[
\text{Overshoot} = [\square_{[0,10]}(\text{Step-Up} \Rightarrow \square_{[0,2]}(\text{Overshoot Limit}))]
\]

\[
\text{Overshoot Limit} = [\text{out} < 1.1\text{in}]
\]

- Signals satisfy the engineer’s intended specification
  - The output signal is less than 10% more than the input signal
- Formal specification is not satisfied
  - The input signal is negative
    - \(1.1\text{in} < \text{in}\)
- Fixing the formal specification
  - Overshoot Limit = \([\text{out} < \text{in} + 0.1|\text{in}|]\)
Trace Generation Problem
Trace Generation Problem

- Find traces that satisfy/violate a given STL formula
- The problem is undecidable
- Focus on bounded varying signals
- Restrict to piecewise linear signals

**Single trace generation problem:** Given an STL formula $\varphi$ and a bound $k$, find a $k$-varying piecewise linear signal satisfying $\varphi$.

**Multi trace generation problem:** Given an STL formula $\varphi$ and a bound $k$, and a number $m$, generate $m$ $k$-varying piecewise linear signals satisfying $\varphi$. 
**k-varying Piecewise Linear Signals**

**Piecewise Linear Signal:**
- Has a finite number of non-differentiable (switching) points
- Has linear trajectory between any two switching points

**k-varying Piecewise Linear Signal:**
- There exist \( t_1 < \ldots < t_k < t_{k+1} = \infty \) such that for each sub-formula \( \psi \) of \( \varphi \), switching points of \( \sigma \) and \( \sigma^\psi \) are contained in \( \{t_1, \ldots, t_k\} \).

**Example:**
- \( \sigma \) is 7-varying piecewise linear signal
Approach

* Similar encoding as in Bounded Model Checker
* Reduce the trace generation problem into a satisfiability checking problem
* Encode k-varying piecewise linear signal for an STL specification into an SMT formula over finite number of boolean and real variables

\[ \varphi \xrightarrow{k} \text{Encoder} \xrightarrow{Enc(\varphi, k)} \text{SMT Solver} \xrightarrow{} \text{Satisfying Trace} \]

* Satisfying assignment of the formula \( Enc(\varphi, k) \) provides a satisfying trace
* Satisfying assignment of the formula \( Enc(\neg \varphi, k) \) provides a violating trace
Overview of Encoding

STL formula \( \varphi \) and its sub-formula \( \psi \)

- Signals corresponding to sub-formula are captured using finite number of variables
- Establish relation between variables such that they are consistent with respect to signal \( \sigma \)
- Encode all relations between variables into an SMT formula

Soundness: Satisfiability of the SMT formula generates satisfying k-varying piecewise linear signal
Example: Encoding

- **Specification:** Always in the interval $[2, 5]$, Output signal value is less than or equal to 10

- **Sub-formulas:** $\psi_1 = \lbrack out > 10 \rbrack$, $\psi_2 = \lnot \psi_1$, $\varphi = [\Box_{[2,5]} \psi_2]$

Signals corresponding to the sub-formulas and SMT encoding variables:
Step 1: Encoding Predicate

**Encoding:** Relationship between variables for signal \( out \) and \( out^{\psi_1} \)

\[
\psi_1 = [out > 10]
\]

**Relationship:**
- \( \psi_1 \) is true at \( t_i \) iff \( x_i \) satisfies the predicate \( x_i > 10 \)

\[
a_i^{\psi_1} = \top \iff x_i > 10
\]

- If \( \psi_1 \) is true in the interval \( (t_i, t_{i+1}) \), then all values \( x \) on the line between points \( x_i \) and \( x_{i+1} \) will satisfy the predicate \( x > 10 \)

\[
b_i^{\psi_1} = \bot \Rightarrow [x_i \leq 10 \land x_{i+1} \leq 10]
\]

- If \( \psi_1 \) is false in the interval \( (t_i, t_{i+1}) \), then all values \( x \) on the line between points \( x_i \) and \( x_{i+1} \) will violate the predicate \( x > 10 \)

\[
b_i^{\psi_1} = \top \Rightarrow [x_i > 10 \land x_{i+1} \geq 10] \lor [x_i \geq 10 \land x_{i+1} > 10]
\]
**Step 2: Encoding Negation**

**Encoding:** Relationship between variables for signal $out^{\psi_2}$ and $out^{\psi_1}$

- If $\psi_2$ is false in the interval $(t_i, t_{i+1})$, then $\psi_1$ will be true in the interval $(t_i, t_{i+1})$

**Relationship:**

- If $\psi_2$ is true at $t_i$, then $\psi_1$ will be false at $t_i$

- If $\psi_2$ is false at $t_i$, then $\psi_1$ will be true at $t_i$

- If $\psi_2$ is true in the interval $(t_i, t_{i+1})$, then $\psi_1$ will be false in the interval $(t_i, t_{i+1})$
Step 3: Encoding Always

**Encoding:** Relationship between variables for signal $\text{out}^\varphi$ and $\text{out}^{\psi_2}$

\[
\varphi = \Box_{[2,5]} \psi_2
\]

**Relationship:**

- If $\varphi$ is true at $t_i$, then $\psi_2$ will be true in the interval $[t_i + 2, t_i + 5]$.
- If $\varphi$ is false at $t_i$, then $\psi_2$ will be false at some point in the interval $[t_i + 2, t_i + 5]$.

$C_{\psi_2}(I)$ is true if and only if $\psi_2$ is true in the interval $I$

\[
\bigwedge_{1 \leq i \leq 5} (a_i^\varphi = \top \iff C_{\psi_2}([t_i + 2, t_i + 5]))
\]
Step 3: Encoding Always

**Encoding:** Relationship between variables for signal $\text{out}^\varphi$ and $\text{out}^{\psi_2}$

\[ \varphi = \square^{[0,5]} \psi_2 \]

**Relationship:**

- If $\varphi$ is true in the interval $(t_i, t_{i+1})$, then for all $t \in (t_i, t_{i+1})$, $\psi_2$ will be true in the interval $[t + 2, t + 5]$

\[ \bigwedge_{1 \leq i \leq 5} \left( b_i^\varphi = \top \Rightarrow \mathbb{C}_{\psi_2}(t_i + 2, t_{i+1} + 5) \right) \]

- If $\varphi$ is false in the interval $(t_i, t_{i+1})$, then for all $t \in (t_i, t_{i+1})$, $\psi_2$ will be false in the interval $[t + 2, t + 5]$

\[ \bigwedge_{1 \leq i \leq 5} \left( b_i^\varphi = \bot \Rightarrow \mathbb{N}_{\psi_2}((t_i, t_{i+1}), [2, 5]) \right) \]
Experiments
Experimental Setup

- Evaluate the performance of our approach along:
  - Varying complexity of STL formula specifications
  - Increasing value of k for k-varying signals
  - Increasing the number m of satisfying and violating traces returned
- STL formula considered: Value High, Steady State Error High, Overshoot, and Rise Time
- Our method is implemented in MATLAB using Breach STL toolbox and Z3 SMT solver
  - Breach STL toolbox is used for parsing STL formula
  - Z3 SMT solver is used for solving the encoding
## Experimental Results

<table>
<thead>
<tr>
<th>Specification</th>
<th>k</th>
<th>No. of ExamplesReturned (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td><strong>Value_High</strong></td>
<td>1</td>
<td>2.0484</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.410</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.506</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0.795</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>2.176</td>
</tr>
<tr>
<td><strong>SE_Error_High</strong></td>
<td>1</td>
<td>1.174</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2.169</td>
</tr>
<tr>
<td><strong>Overshoot</strong></td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.961</td>
</tr>
<tr>
<td><strong>Rise_Time</strong></td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.906</td>
</tr>
</tbody>
</table>

- Value High specification is a simple STL formula
- Other specifications are nested STL formula
- Scalable with respect to k for simple structure of STL formula
## Experimental Results

<table>
<thead>
<tr>
<th>Specification</th>
<th>k</th>
<th>No. of Examples Returned (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Value_High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2.0484</td>
<td>2.032</td>
</tr>
<tr>
<td>4</td>
<td>0.410</td>
<td>1.008</td>
</tr>
<tr>
<td>8</td>
<td>0.506</td>
<td>1.472</td>
</tr>
<tr>
<td>16</td>
<td>0.795</td>
<td>2.810</td>
</tr>
<tr>
<td>32</td>
<td>2.176</td>
<td>9.082</td>
</tr>
<tr>
<td>SE_Error_High</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.174</td>
<td>1.888</td>
</tr>
<tr>
<td>4</td>
<td>2.169</td>
<td>6.481</td>
</tr>
<tr>
<td>Overshoot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>0.961</td>
<td>2.498</td>
</tr>
<tr>
<td>Rise_Time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>0.906</td>
<td>2.338</td>
</tr>
</tbody>
</table>

* Scalable with respect to number of traces m for simple structure of STL formula
Conclusion and Future Works

Conclusions:
- Presented a method for automatic satisfying/violating trace generation from a formal specification
- Provide a framework to control engineers for debugging formal specifications

Future Works:
- Investigate more efficient encoding of STL formula into SMT formula such that less number of variables are required
- Consider experimentation on a large set of industrial STL specifications
- Explore strategies for generating multiple satisfying/violating traces and ensure coverage