Specification Mining of Industrial-scale Control Systems

Alexandre Donzé

Joint work with
Xiaoqing Jin, Jyotirmoy V. Deshmuck, Sanjit A. Seshia

University of California, Berkeley

May 14, 2013
Technically, hybrid systems integrating continuous dynamics, switching logics, computations, etc.
The model-based design (MBD) V design process.
Model-Based Design

The actual design process.
Model-Based Design

The actual design process.

- Alternation between specification and design,
- A flavor of *chicken and egg* problem...
Motivations for Specification Mining

Specification should be objects of equal importance as the design itself.
Motivations for Specification Mining

Specification should be objects of equal importance as the design itself.

This enables

- co-developement of design and specification
- automatization of verification and testing
Motivations for Specification Mining

Specification should be objects of equal importance as the design itself.

This enables

- co-developement of design and specification
- automatization of verification and testing

However

- this is not (yet) the case: specification are often high level, vague textual/oral/implicit requirements
- this was not the case: problems of reusability of older component (legacy code).
Motivations for Specification Mining

Specification should be objects of equal importance as the design itself.

This enables

- co-developement of design and specification
- automatization of verification and testing

However

- this is not (yet) the case: specification are often high level, vague textual/oral/implicit requirements
- this was not the case: problems of reusability of older component (legacy code).

To construct/reconstruct formal and usable specifications, there is a need specification mining techniques.
Challenges

Closed-loop setting very complex

- nonlinear dynamics
- look-up tables
- large amounts of switching
- components with no models
- unclear semantics of modeling language

What can we do, as formally as possible if all we have is

- the ability to simulate the system
- some vague idea of what the system should satisfy
- the ability to check if simulation traces satisfy properties
1. Specification Using Signal Temporal Logics

2. Mining Algorithm

3. Experimental Results
Temporal logics in a nutshell

Temporal logics allow to specify patterns that timed behaviors of systems may or may not satisfy.

The most intuitive is the Linear Temporal Logic (LTL), dealing with discrete sequences of states.

Based on logic operators ($\neg$, $\land$, $\lor$) and temporal operators: “next”, “always” (alw), “eventually” (ev) and “until” (U)
From LTL to STL

Extension of LTL with real-time and real valued constraints
From LTL to STL

Extension of LTL with real-time and real valued constraints

LTL $G( a \Rightarrow F b)$
Boolean predicates, discrete-time
From LTL to STL

Extension of LTL with real-time and real valued constraints

**LTL** $G( a \implies F b)$
Boolean predicates, discrete-time

**MITL** $G( a \implies F_{[0,5s]} b )$
Boolean predicates, real time
From LTL to STL

Extension of LTL with real-time and real valued constraints

LTL $\mathcal{G}(a \implies Fb)$
Boolean predicates, discrete-time

MITL $\mathcal{G}(a \implies F_{[0,5s]}b)$
Boolean predicates, real time

STL $\mathcal{G}(f(x[t]) > 0 \implies F_{[0,5s]}g(y[t]) > 0)$
Predicates over real values, real time
Formal Definitions

Definition (STL Syntax)

\[ \varphi := \mu \mid \neg \varphi \mid \varphi \land \psi \mid \varphi \ U_{[a,b]} \psi \]

where \( \mu \) is a predicate of the form \( \mu : \mu(x[t]) > 0 \)
Formal Definitions

Definition (STL Syntax)

\[ \varphi ::= \mu \mid \neg \varphi \mid \varphi \land \psi \mid \varphi \ U_{[a,b]} \psi \]

where \( \mu \) is a predicate of the form \( \mu : \mu(x[t]) > 0 \)

Definition (STL Semantics)

The validity of a formula \( \varphi \) with respect to a signal \( x \) at time \( t \) is

\[
\begin{align*}
(x, t) \models \mu & \iff \mu(x[t]) > 0 \\
(x, t) \models \varphi \land \psi & \iff (x, t) \models \varphi \land (x, t) \models \psi \\
(x, t) \models \neg \varphi & \iff \neg((x, t) \models \varphi) \\
(x, t) \models \varphi \ U_{[a,b]} \psi & \iff \exists t' \in [t + a, t + b] \text{ s.t. } (x, t') \models \psi \land \\
& \quad \forall t'' \in [t, t'], (x, t'') \models \varphi
\end{align*}
\]

Additionally: \( \text{ev}_{[a,b]} \varphi = \top \ U_{[a,b]} \varphi \) and \( \text{alw}_{[a,b]} \varphi = \neg(\text{ev}_{a,b} \neg \varphi) \).
STL Examples
STL Examples

“The signal is never above 3.5”

\[ \varphi := \text{alw} \ (x[t] < 3.5) \]
“After 2s, the signal is never above 3”

\[ \varphi := \text{ev}_{[0,2]} \\text{alw} (x[t] < 3) \]
STL Examples

“Between 2s and 6s the signal is never above 2”

\[ \varphi := \text{alw}_{[2,6]} \ (x[t] < 2) \]
“Always when $x > 0.5$, 0.6s later it settles under 0.5 for 1.5s”

\[ \varphi := \text{alw}(x[t] > 0.5 \rightarrow \text{ev}_{[0, 0.6]}(\text{alw}_{[0, 1.5]} x[t] < 0.5)) \]
Quantitative Satisfaction

Given $\varphi$, a signal $x$ and a time $t$, define a function $\rho$:

$$
\rho(\varphi, x, t) > 0 \implies x, t \models \varphi
$$
$$
\rho(\varphi, x, t) < 0 \implies x, t \not\models \varphi
$$

$\rho(\varphi, \cdot, \cdot)$ transforms $x$ into a *satisfaction* signal, sometimes noted $\varphi(x)[t]$. 
STL Transducers

\[ x[t] \]

STL Monitor

Formula \( \varphi \)
STL Transducers

\[ x[t] \]

\[ \rho(\varphi, x, t) \]

STL Monitor

Formula \( \varphi \)
STL Transducers

\[ x[t] \]

STL Monitor

Formula \( \varphi \)

\[ \rho(\varphi, x, t) \]
STL Atomic Transducers

$$x[t] \xrightarrow{\text{Predicate } x > 5} x[t] - 5$$

$$\mu(x) = x - 5$$
STL Atomic Transducers

$x[t] \rightarrow \text{Predicate } x > 5$

$\mu(x) = x - 5 \rightarrow x[t] - 5$

$\varphi(x)[t] \rightarrow \text{Negation } \neg \varphi$

$\psi(x) = -\varphi(x) \rightarrow -\varphi(x)[t]$
STL Atomic Transducers

\[ x[t] \rightarrow \text{Predicate } x > 5 \]
\[ \mu(x) = x - 5 \rightarrow x[t] - 5 \]

\[ \varphi(x)[t] \rightarrow \text{Negation } \neg \varphi \]
\[ \psi(x) = \neg \varphi(x) \rightarrow \neg \varphi(x)[t] \]

\[ \varphi_1(x)[t] \rightarrow \text{Conjunction } \varphi_1 \land \varphi_2 \]
\[ \psi(x) = \min(\varphi_1(x), \varphi_2(x)) \rightarrow \min(\varphi_1(x)[t], \varphi_2(x)[t]) \]

Alexandre Donzé
Specification Mining for Industrial-scale Designs
DREAMS, May 14, 2013
STL Atomic Transducers

Eventually $\text{ev}[0.1, 0.2] \varphi$

$$\psi(x) = \max_{t + 0.1, t + 0.2} \varphi(x)$$

$$\varphi(x)[t] \rightarrow \psi(x) = \max_{t + 0.1, t + 0.2} \varphi(x) \rightarrow \max_{t' \in [t + 0.1, t + 0.2]} \varphi(x)[t']$$
STL Atomic Transducers

**Eventually** $\text{ev}_{[1,2]} \varphi$

\[
\psi(x) = \max_{[t+1,t+2]} \varphi(x)
\]

\[
\max_{t' \in [t+1,t+2]} \varphi(x)[t']
\]

**Always** $\text{alw}_{[1,2]} \varphi$

\[
\psi(x) = \min_{[t+1,t+2]} \varphi(x)
\]

\[
\min_{t' \in [t+1,t+2]} \varphi(x)[t']
\]
STL Atomic Transducers

Eventually $\text{ev}_{[1,2]} \varphi$

$$
\psi(x) = \max_{[t+1,t+2]} \varphi(x)
$$

$$
\max_{t' \in [t+1,t+2]} \varphi(x)[t']
$$

Always $\text{alw}_{[1,2]} \varphi$

$$
\psi(x) = \min_{[t+1,t+2]} \varphi(x)
$$

$$
\min_{t' \in [t+1,t+2]} \varphi(x)[t']
$$

Note

- The “Until” can be computed by a combination of untimed timed $\text{ev}$ and $\text{alw}$. 
Computing the Robust Satisfaction Function
(Donze, Ferrere, Maler, Efficient Robust Monitoring of STL Formula, CAV'13)

- Atomic transducers can be computed in linear time in the size of the input signals

- The function $\varphi(x)[t]$ is computed inductively on the structure of $\varphi$
  - linear time complexity in size of $x$ is preserved
  - exponential worst case complexity in the size of $\varphi$

- Note: current implementation is off-line
Dense-Time and Exponential Complexity

A theoretical example with exponential complexity
Dense-Time and Exponential Complexity

A theoretical example with exponential complexity
Experimental Results

| $|\varphi| = 1$, Time$_\rho \approx 2.34 \times 10^{-6} n_y$ |
| $|\varphi| = 25$, Time$_\rho \approx 1.63 \times 10^{-5} n_y$ |
| $|\varphi| = 50$, Time$_\rho \approx 2.45 \times 10^{-5} n_y$ |
Parametric STL

\[ \phi \colon [0, 2] \rightarrow [\text{always}] (x[t] < 3) \]
“After 2s, the signal is never above 3”

\[
\varphi := \text{ev}_{[0,2]} \, \text{alw} \, (x[t] < 3)
\]
“After $\tau$ s, the signal is never above $\pi$”

$\varphi := \text{ev}_{[0,\tau]} \text{ alw } (x[t] < \pi)$
1 Specification Using Signal Temporal Logics

2 Mining Algorithm

3 Experimental Results
Specification Mining Framework

\[ e \rightarrow \text{Controller} \rightarrow u \rightarrow \text{Plant Model} \]

\[ + \]

\[ y \]

\[ \text{CounterexampleFound} \]

\[ \text{No Counterexample} \]

\[ \text{Ev} \left[ 0, \tau_1 \right] (x_1 < \pi_1 \land \text{alw} \left[ 0, \tau_2 \right] (x_2 > \pi_2)) \]

\[ \tau_1 \leftarrow .7, \pi_1 \leftarrow 2, \pi_2 \leftarrow 3 \]

\[ \text{Ev} \left[ 0, 1.1 \right] (x_1 < 3.7 \land \text{alw} \left[ 0, 5 \right] (x_2 > 0.1)) \]
### Specification Mining Framework

![Diagram](image.png)

**Template Specification**

\[ \text{ev}_{[0, \tau_1]}(x_1 < \pi_1 \land \text{alw}_{[0, \tau_2]}(x_2 > \pi_2)) \]
Specification Mining Framework

\[ \text{ev}_{[0,\tau_1]}(x_1 < \pi_1 \land \text{alw}_{[0,\tau_2]}(x_2 > \pi_2)) \]

Template Specification
Specification Mining Framework

- **Controller**
  - $e$
  - $u$

- **Plant Model**
  - $y$

**Simulation Traces**

**FindParam**

**Candidate Requirement**

- $\tau_1 \leftarrow 0.7$, $\pi_1 \leftarrow 2$, $\pi_2 \leftarrow 3$

**Template Specification**

$\text{ev}_{[0,\tau_1]}(x_1 < \pi_1 \land \text{alw}_{[0,\tau_2]}(x_2 > \pi_2))$
Specification Mining Framework

Controller \[\rightarrow e \rightarrow \text{Controller} \]
Plant Model \[\rightarrow u \rightarrow \text{Plant Model} \]

Simulation Traces \[\rightarrow \text{Simulation Traces} \]

FINDPARAM \[\rightarrow \text{Candidate Requirement} \]

Candidate Requirement \[\rightarrow \text{FALSIFYALGO} \]

\[\tau_1 \leftarrow .7, \pi_1 \leftarrow 2, \pi_2 \leftarrow 3\]

Template Specification

\[\text{ev}_{[0,\tau_1]}(x_1 < \pi_1 \land \text{alw}_{[0,\tau_2]}(x_2 > \pi_2))\]
Specification Mining Framework

\[ \tau_1 \leftarrow 0.7, \pi_1 \leftarrow 2, \pi_2 \leftarrow 3 \]

\[ \text{ev}_{[0, \tau_1]}(x_1 < \pi_1 \land \text{alw}_{[0, \tau_2]}(x_2 > \pi_2)) \]

Template Specification
Specification Mining Framework

\[ \tau_1 \leftarrow 1.1, \pi_1 \leftarrow 3.7, \pi_2 \leftarrow 5 \]

\[ \text{ev}_{[0, \tau_1]}(x_1 < \pi_1 \wedge \text{alw}_{[0, \tau_2]}(x_2 > \pi_2)) \]
Specification Mining Framework

$$\text{Controller} \rightarrow e \rightarrow \text{Plant Model} \leftarrow u \leftarrow y$$

Simulation Traces \rightarrow \text{Counterexample Traces} \rightarrow \text{Candidate Requirement} \rightarrow \text{FALSIFYALGO}

- $$\tau_1 \leftarrow 1.1$$, $$\pi_1 \leftarrow 3.7$$, $$\pi_2 \leftarrow 5$$
- $$\text{ev}_{[0, \tau_1]}(x_1 < \pi_1 \land \text{alw}_{[0, \tau_2]}(x_2 > \pi_2))$$
- $$\text{ev}_{[0, 1.1]}(x_1 < 3.7 \land \text{alw}_{[0, 5]}(x_2 > 0.1))$$

Template Specification

Inferred Specification

Counterexample Found

No Counterexample
Parameter synthesis

Problem

Given a system $S$ with a PSTL formula with $n$ symbolic parameters $\varphi(p_1, \ldots, p_n)$, find a **tight** valuation function $v$ such that

$$x, t \models \varphi(v(p_1), \ldots, v(p_n)),$$

Challenges

- Multiple solutions: equivalent to multi-objective problem
- We require tight specifications (avoid over-conservatism)
Example
Example

$$\varphi := \text{alw}(x[t] > \pi \rightarrow \text{ev}_{[0, \tau_1]} (\text{alw}_{[0, \tau_2]} x[t] < \pi))$$
\varphi := \text{alw}(x[t] > \pi_1 \rightarrow \text{ev}_{[0,\tau_1]}(\text{alw}_{[0,\tau_2]}(x[t] < \pi_2)))
Parameter synthesis: Monotonicity

**Definition**

A PSTL formula $\varphi(p_1, \cdots, p_n)$ is monotonically increasing with respect to $p_i$ if for every signal $x$,

$$\forall v, v', x \models \varphi(..., v(p_i), ...)$$

$$v'(p_i) \geq v(p_i) \Rightarrow x \models \varphi(..., v'(p_i), ...) \quad (1)$$

It is monotonically decreasing if this holds when replacing $v'(p_i) \geq v(p_i)$ with $v'(p_i) \leq v(p_i)$. 
Parameter synthesis: Monotonicity

Definition

A PSTL formula $\varphi(p_1, \ldots, p_n)$ is monotonically increasing with respect to $p_i$ if for every signal $x$,

$$\forall v, v', x \models \varphi(\ldots, v(p_i), \ldots), \quad v'(p_i) \geq v(p_i) \Rightarrow x \models \varphi(\ldots, v'(p_i), \ldots) \quad (1)$$

It is monotonically decreasing if this holds when replacing $v'(p_i) \geq v(p_i)$ with $v'(p_i) \leq v(p_i)$.

- If a formula is monotonic, the parameter synthesis problem can be reduced to a generalized binary search
- Deciding monotonicity can be encoded in an SMT query (however, the problem is undecidable)
Falsification problem

Problem

Given the system:

\[ u(t) \rightarrow \text{System } S \rightarrow S(u(t)) \]

Find an input signal \( u \in \mathcal{U} \) such that \( S(u(t)), 0 \not\models \varphi \)

Approach: Solve

\[ \rho^* = \min_{u \in \mathcal{U}} \rho(\varphi, S(u(t)), 0) \quad (1) \]

If \( \rho^* < 0 \), we found a counterexample input.

In practice:

▶ We parameterize \( \mathcal{U} \)
▶ Try nonlinear, stochastic optimization methods
▶ Different algorithms implemented in S-TaLiRo (G. Fainekos et al) and Breach

Alexandre Donzé
Specification Mining for Industrial-scale Designs
DREAMS, May 14, 2013 25 / 32
Falsification problem

Given the system:

\[ u(t) \rightarrow \text{System } S \rightarrow S(u(t)) \]

Find an input signal \( u \in U \) such that \( S(u(t)), 0 \not\models \varphi \)

Approach: Solve \( \rho^* = \min_{u \in U} \rho(\varphi, S(u), 0) \) (1) If \( \rho^* < 0 \), we found a counterexample input.
Falsification problem

**Problem**

*Given the system:*

\[ u(t) \rightarrow \text{System } S \rightarrow S(u(t)) \]

*Find an input signal* \( u \in \mathcal{U} \) *such that* \( S(u(t)), 0 \not\models \varphi \)

**Approach:**

Solve \( \rho^* = \min_{u \in \mathcal{U}} \rho(\varphi, S(u), 0) \) \hspace{1cm} (1) \hspace{1cm} If \( \rho^* < 0 \), we found a counterexample input.

**In practice:**

- We parameterize \( \mathcal{U} \)
- Try nonlinear, stochastic optimization methods
- Different algorithms implemented in S-TaLiRo (G. Fainekos et al) and Breach
1 Specification Using Signal Temporal Logics

2 Mining Algorithm

3 Experimental Results
Automatic Transmission System
Formulas

- The speed is always below \( \pi_1 \) and RPM below \( \pi_2 \)

\[
\varphi_{sp\_rpm}(\pi_1, \pi_2) := \text{alw}\left( (\text{speed} < \pi_1) \land (\text{RPM} < \pi_2) \right).
\]

- The vehicle cannot reach 100 mph in \( \tau \) seconds with RPM always below \( \pi \)

\[
\varphi_{rpm100}(\tau, \pi) := \neg(\text{ev}_{[0, \tau]}(\text{speed} > 100) \land \text{alw}(\text{RPM} < \pi)).
\]

- Whenever it shift to gear 2, it dwells in gear 2 for at least \( \tau \) seconds

\[
\varphi_{stay}(\tau) := \text{alw}\left( \left( \text{gear} \neq 2 \land \text{ev}_{[0, \varepsilon]}(\text{gear} = 2) \right) \Rightarrow \text{alw}_{[\varepsilon, \tau]}(\text{gear} = 2) \right).
\]
## Results

<table>
<thead>
<tr>
<th>Template</th>
<th>S-Taliro-based falsification</th>
<th>Breach-based falsification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parameter values</td>
<td>Fals.</td>
</tr>
<tr>
<td>$\varphi_{sp_rpm}(\pi_1, \pi_2)$</td>
<td>(155 mph, 4858 rpm)</td>
<td>55 s</td>
</tr>
<tr>
<td>$\varphi_{rpm100}(\pi, \tau)$</td>
<td>(3278.3 rpm, 49.91 s)</td>
<td>6422 s</td>
</tr>
<tr>
<td>$\varphi_{rpm100}(\tau, \pi)$</td>
<td>(4997 rpm, 12.20 s)</td>
<td>8554 s</td>
</tr>
<tr>
<td>$\varphi_{stay}(\pi)$</td>
<td>1.79 s</td>
<td>18886 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18886 s</td>
</tr>
</tbody>
</table>

Breach-based falsification
Results on Industrial-scale Model

- 4000+ Simulink blocks
- Look-up tables
- Nonlinear dynamics

- Found max overshoot with 7000+ simulations in 13 hours
- Attempt to mine maximum observed settling time:
  - stops after 4 iterations
  - gives answer $t_{\text{settle}} = \text{simulation time horizon}$
Bug Finding

- The above trace found an actual (unexpected) bug in the model
- The cause was identified as a wrong value in a look-up table
Summary

- A general framework for specification mining of complex cyber-physical systems

Outlook

- Falsification/optimization of satisfaction functions
- Online monitoring and specification mining
- More elaborate templates mining (beyond parameters)
- How to help designers writing and using temporal logics templates and formulas?

Thanks!