Specification Mining of Industrial-scale Control Systems

Alexandre Donzé

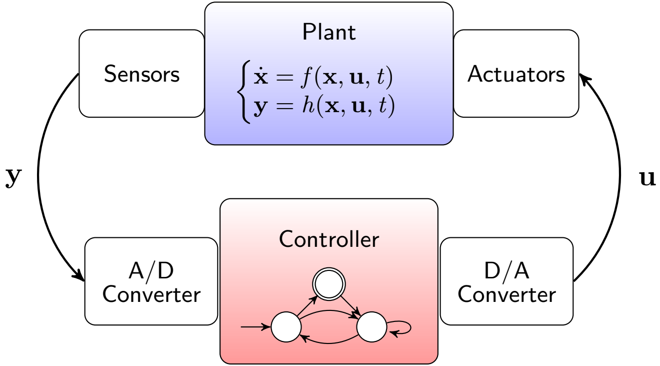
Joint work with

Xiaoqing Jin, Jyotirmoy V. Deshmuck, Sanjit A. Seshia

University of California, Berkeley

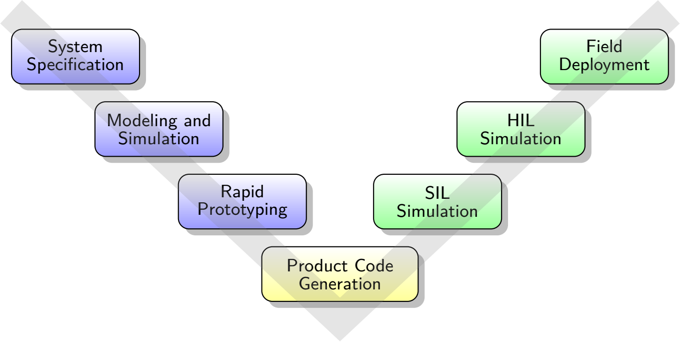
May 14, 2013

Control Systems Design



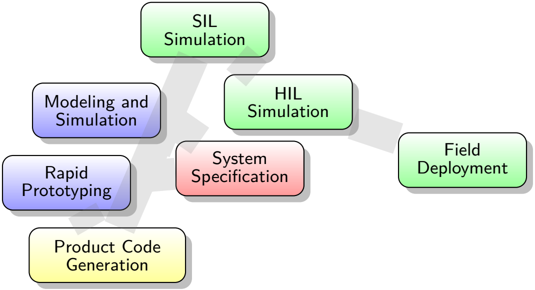
Technically, hybrid systems integrating continuous dynamics, switching logics, computations, etc.

Model-Based Design



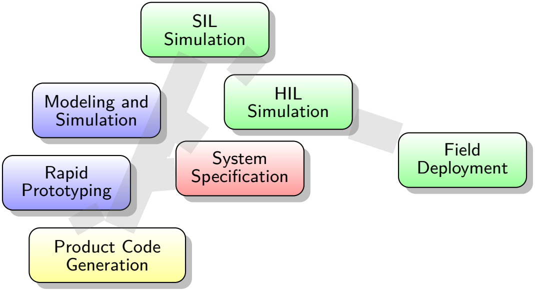
The model-based design (MBD) V design process.

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

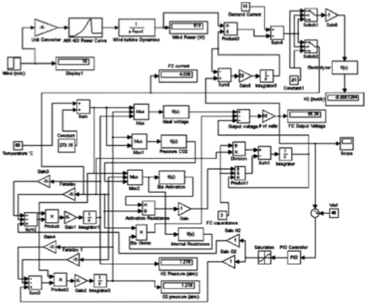
- this is not (yet) the case: specification are often high level, vague textual/oral/implicit requirements
- this was not the case: problems of reusability of older component (legacy code).

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

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 ($\neg$, $\wedge$, $\vee$) and temporal operators: "next", "always" (alw), "eventually" (ev) and "until" ($\mathbf{U}$)

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 => F b)
Boolean predicates, discrete-time

From LTL to STL

Extension of LTL with real-time and real valued constraints

LTL G(a => F b)
Boolean predicates, discrete-time

MITL G(a => $F_{[0,.5s]}$ b)
Boolean predicates, real time

From LTL to STL

Extension of LTL with real-time and real valued constraints

LTL G(a $=>$ F b)
Boolean predicates, discrete-time

MITL G(a $=>$ $\mathsf{F}_{[0,.5s]}$ b)
Boolean predicates, real time

STL G($f(x[t]) > 0 => \mathsf{F}_{[0,.5s]} g(y[t]) > 0$)
Predicates over real values , real time

Formal Definitions

Definition (STL Syntax)

$$\varphi := \mu \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi\ \mathbf{U}_{[a,b]}\ \psi$$

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

Formal Definitions

Definition (STL Syntax)

$$\varphi := \mu \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \ \mathbf{U}_{[a,b]} \ \psi$$

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

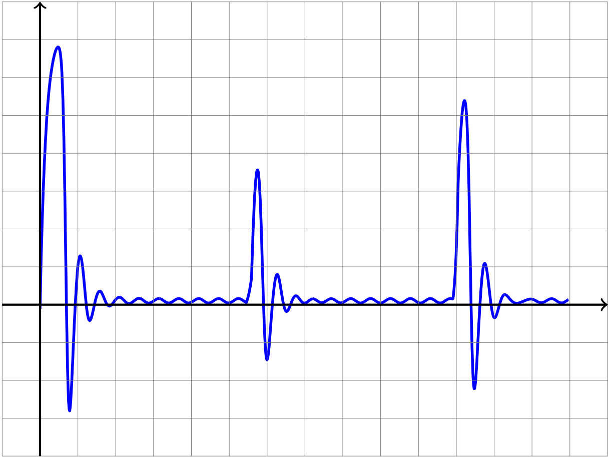
Definition (STL Semantics)

The validity of a formula φ with respect to a signal x at time t is

$$\begin{array}{lcl}
(x,t) \models \mu & \Leftrightarrow & \mu(x[t]) > 0 \\
(x,t) \models \varphi \wedge \psi & \Leftrightarrow & (x,t) \models \varphi \wedge (x,t) \models \psi \\
(x,t) \models \neg\varphi & \Leftrightarrow & \neg((x,t) \models \varphi) \\
(x,t) \models \varphi \,\mathcal{U}_{[a,b]}\, \psi & \Leftrightarrow & \exists t' \in [t+a, t+b] \text{ s.t. } (x,t') \models \psi \ \wedge \\
 & & \qquad \forall t'' \in [t,t'],\ (x,t'') \models \varphi \}
\end{array}$$

Additionally: $\text{ev}_{[a,b]}\ \varphi = \top\ \mathcal{U}_{[a,b]}\ \varphi$ and $\text{alw}_{[a,b]}\varphi = \neg(\text{ev}ab\neg\varphi)$.

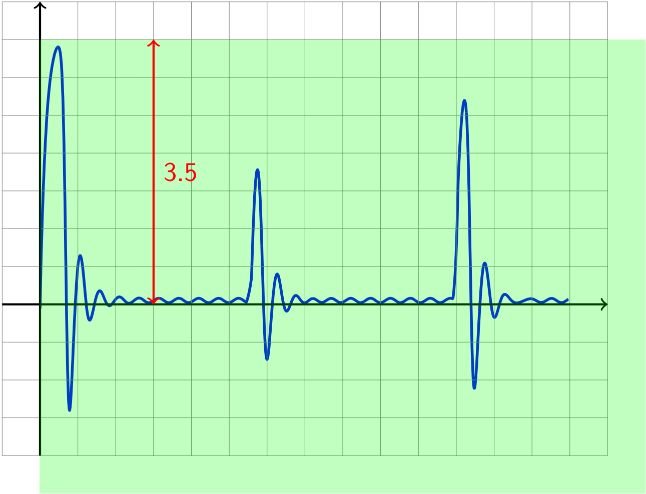
STL Examples



STL Examples

"The signal is never above 3.5"

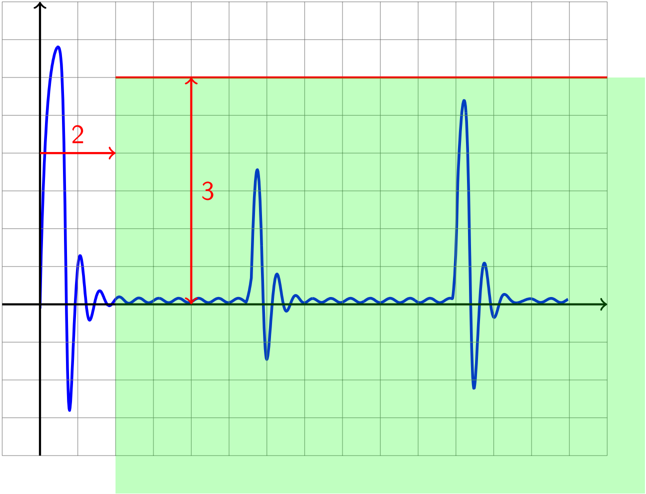
$\varphi :=$ alw $(x[t] < 3.5)$



STL Examples

"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"

$$\varphi := \text{alw}_{[2,6]} \ (x[t] < 2)$$

STL Examples

"Always when x >0.5, 0.6s later it settles under 0.5 for 1.5s"

$$\varphi := \mathsf{alw}(x[t] > .5 \rightarrow \ \mathsf{ev}_{[0,.6]} \ \ (\ \mathsf{alw}_{[0,1.5]}\ x[t] < 0.5))$$

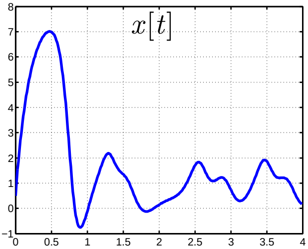
Quantitative Satisfaction

Given φ, a signal x and a time t, define a function ρ:

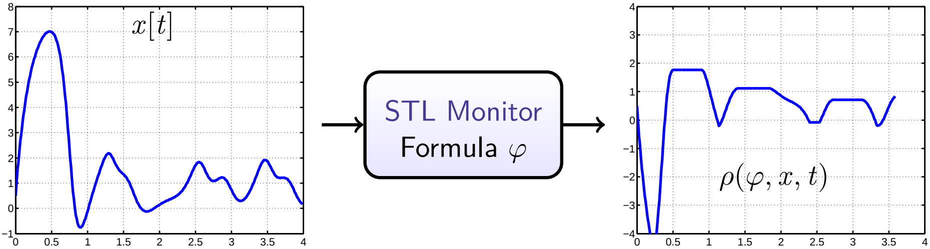
$$\rho(\varphi, x, t) > 0 \Rightarrow x, t \models \varphi$$
$$\rho(\varphi, x, t) < 0 \Rightarrow x, t \not\models \varphi$$

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

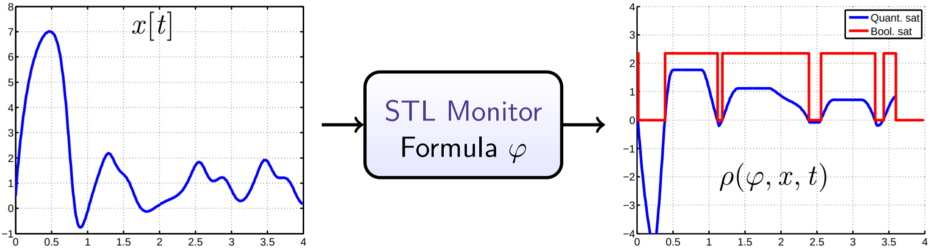
STL Transducers



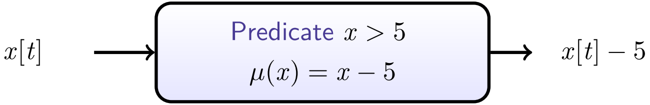
STL Transducers



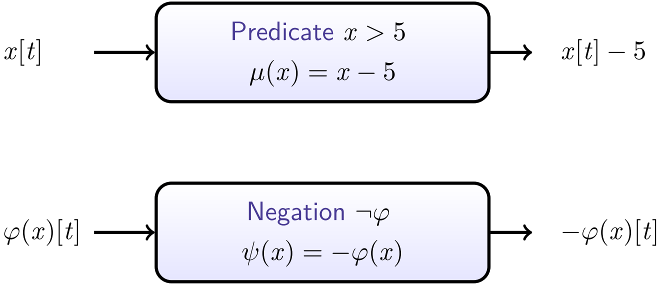
STL Transducers



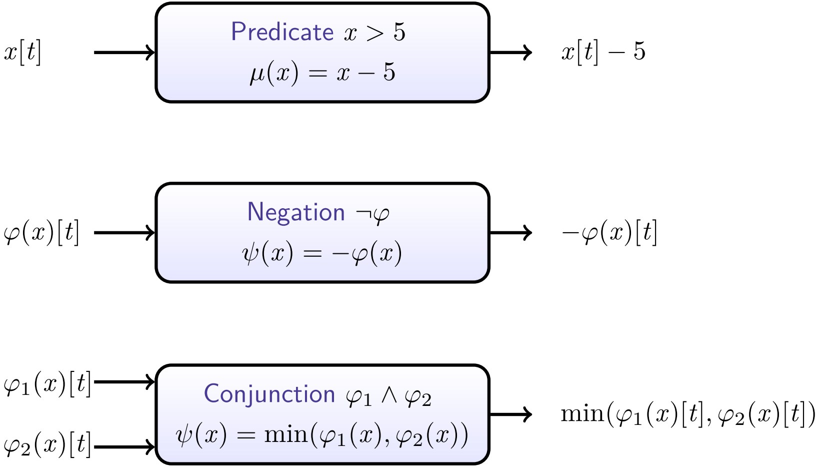
STL Atomic Transducers



STL Atomic Transducers



STL Atomic Transducers



STL Atomic Transducers

$\varphi(x)[t]$ →

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

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

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

STL Atomic Transducers

$\varphi(x)[t] \longrightarrow$ **Eventually** $\mathsf{ev}_{[.1,.2]}\varphi$: $\psi(x) = \max_{[t+.1,t+.2]} \varphi(x)$ $\longrightarrow \max_{t' \in [t+.1,t+.2]} \varphi(x)[t']$

$\varphi(x)[t] \longrightarrow$ **Always** $\mathsf{alw}_{[.1,.2]}\varphi$: $\psi(x) = \min_{[t+.1,t+.2]} \varphi(x)$ $\longrightarrow \min_{t' \in [t+.1,t+.2]} \varphi(x)[t']$

STL Atomic Transducers

$\varphi(x)[t]$ → **Eventually** $\mathsf{ev}_{[.1,.2]}\varphi$

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

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

$\varphi(x)[t]$ → **Always** $\mathsf{alw}_{[.1,.2]}\varphi$

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

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

Note

- The "Until" can be computed by a combination of untimed timed ev and alw.

Computing the Robust Satisfaction Function

(Donze, Ferrere, Maler, *Efficient Robust Monitoring of STL Formula*, CAV'13)

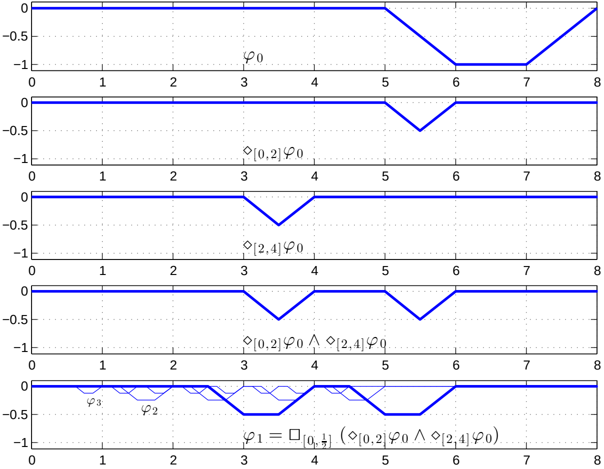
- Atomic transducers can be computed in linear time in the size of the input signals

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

- Note: current implementation is off-line

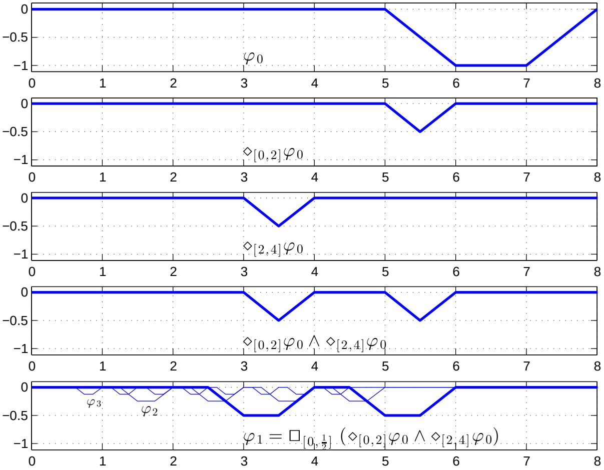
Dense-Time and Exponential Complexity

A theoretical example with exponential complexity

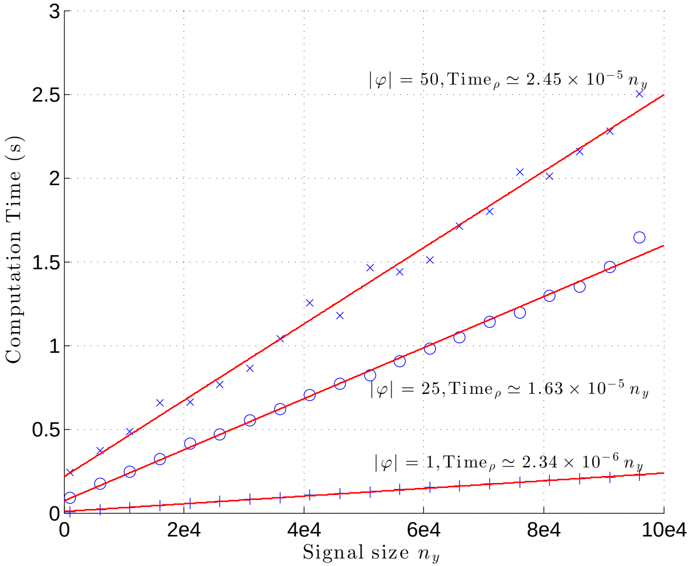


Dense-Time and Exponential Complexity

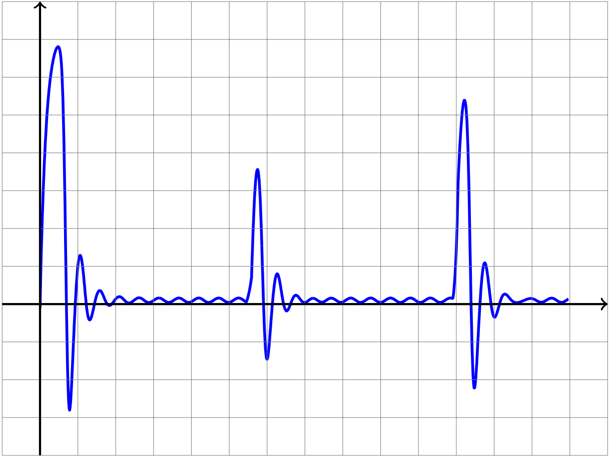
A theoretical example with exponential complexity



Experimental Results



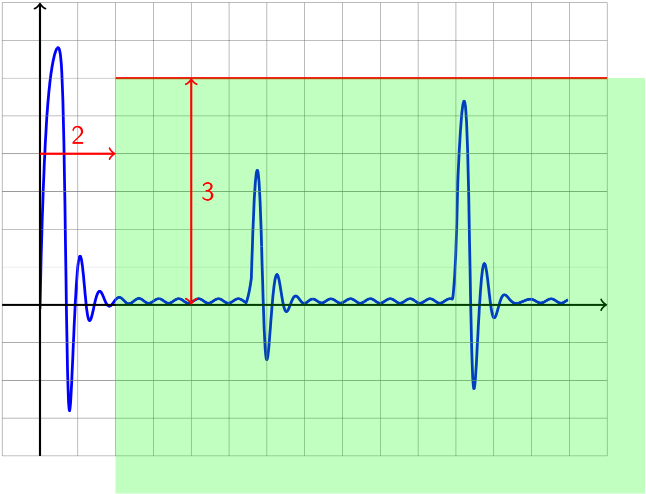
Parametric STL



Parametric STL

"After 2s, the signal is never above 3"

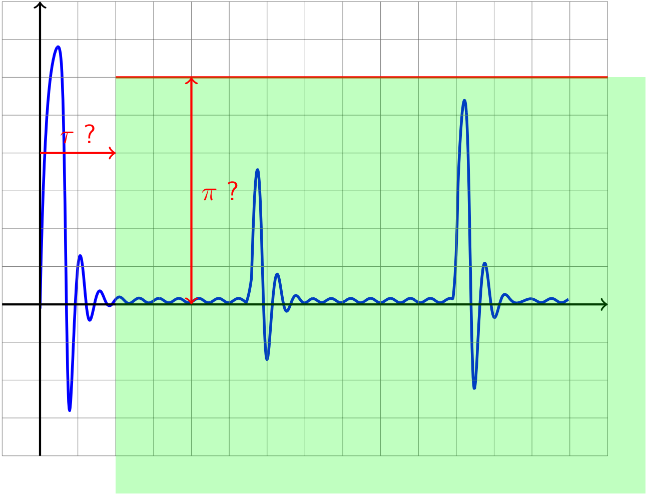
$$\varphi := \text{ ev}_{[0,2]} \text{ alw } (x[t] < 3)$$



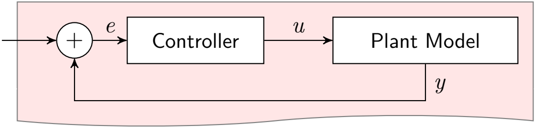
Parametric STL

"After τ s, the signal is never above π"

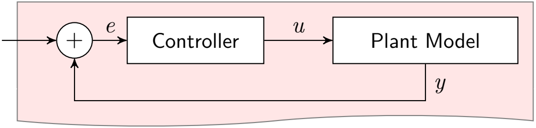
$$\varphi := \text{ev}_{[0,\tau]} \text{ alw } (x[t] < \pi)$$



Specification Mining Framework



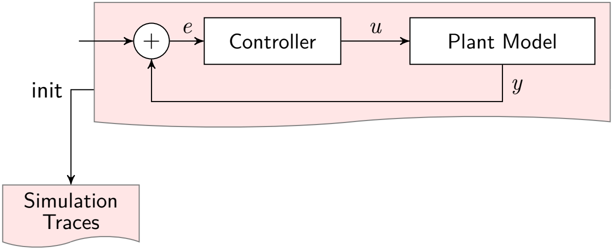
Specification Mining Framework



$$\mathsf{ev}_{[0,\tau_1]}(\mathtt{x_1} < \pi_1 \wedge \mathsf{alw}_{[0,\tau_2]}(\mathtt{x_2} > \pi_2))$$

Template Specification

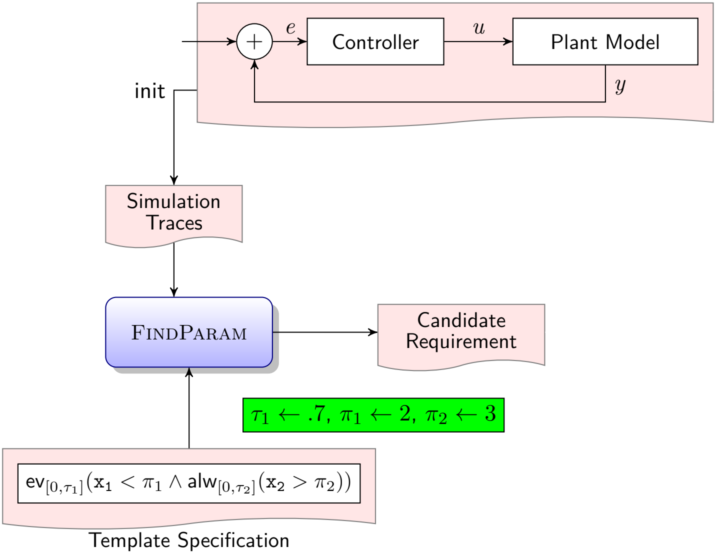
Specification Mining Framework



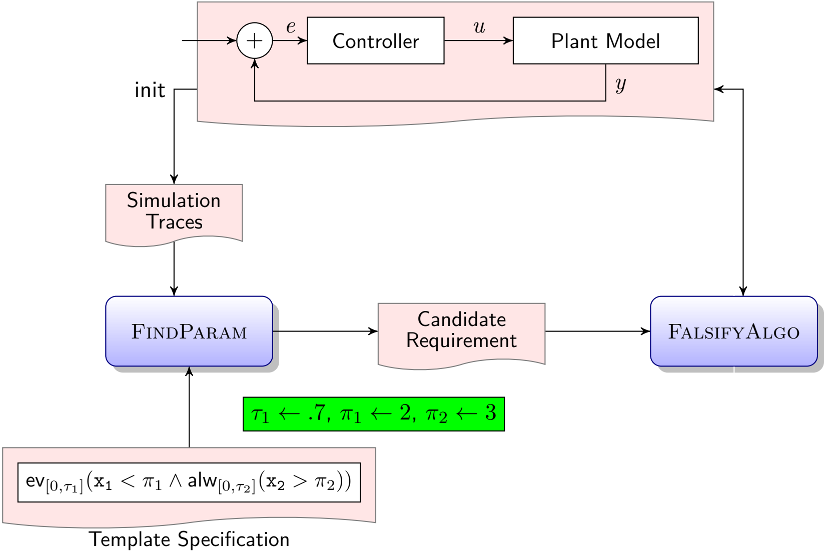
$$\mathsf{ev}_{[0,\tau_1]}(\mathtt{x}_1 < \pi_1 \wedge \mathsf{alw}_{[0,\tau_2]}(\mathtt{x}_2 > \pi_2))$$

Template Specification

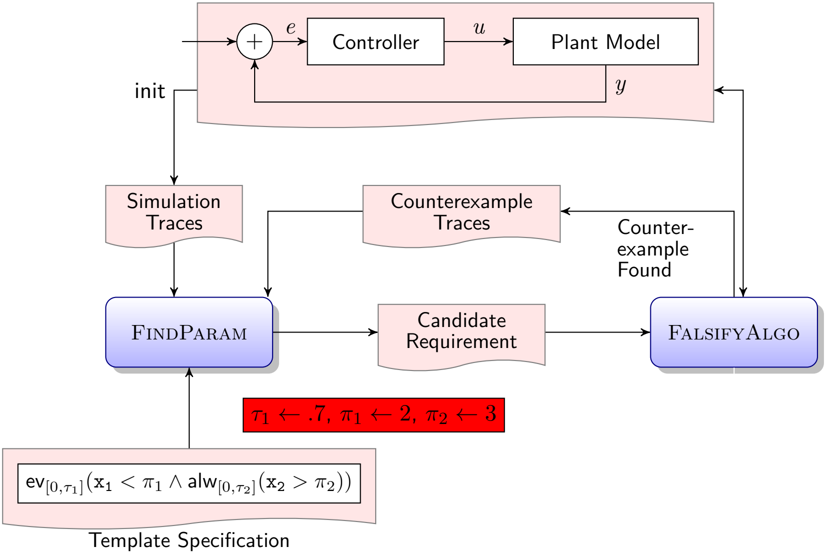
Specification Mining Framework



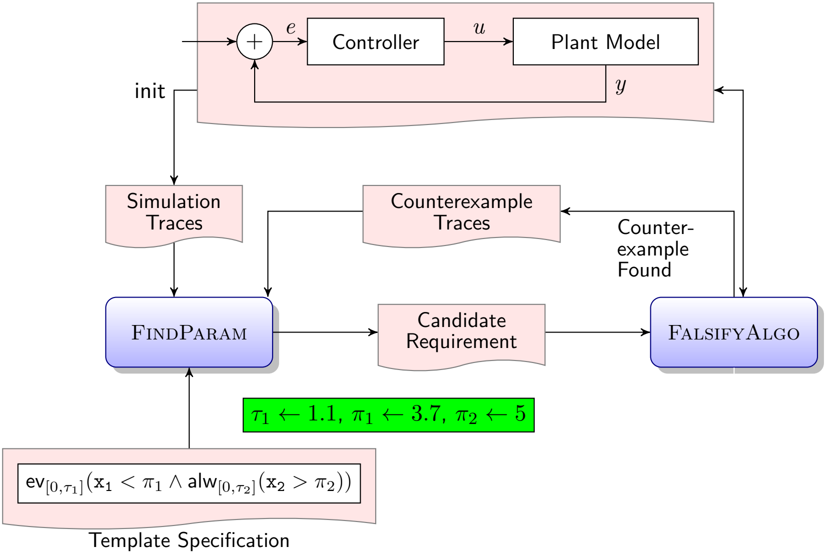
Specification Mining Framework



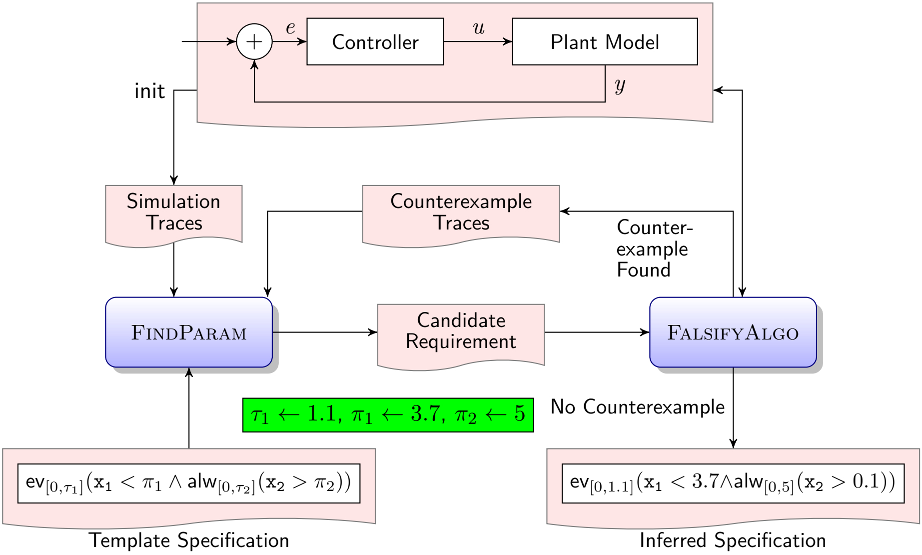
Specification Mining Framework



Specification Mining Framework



Specification Mining Framework



Parameter synthesis

Problem

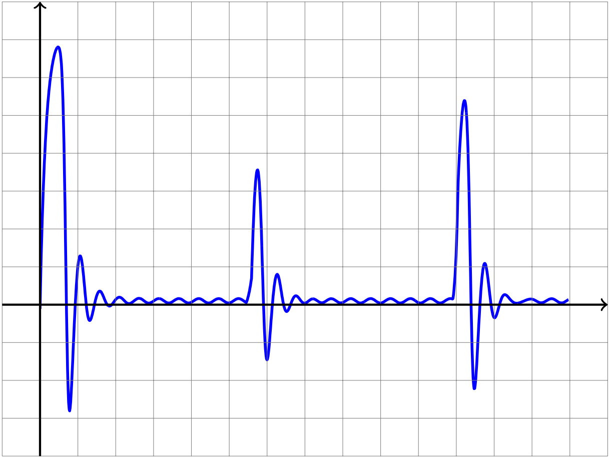
Given a system $\mathcal{S}$ with a PSTL formula with n symbolic parameters $\varphi(p_1, \ldots, p_n)$, find a tight valuation function v such that

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

Challenges

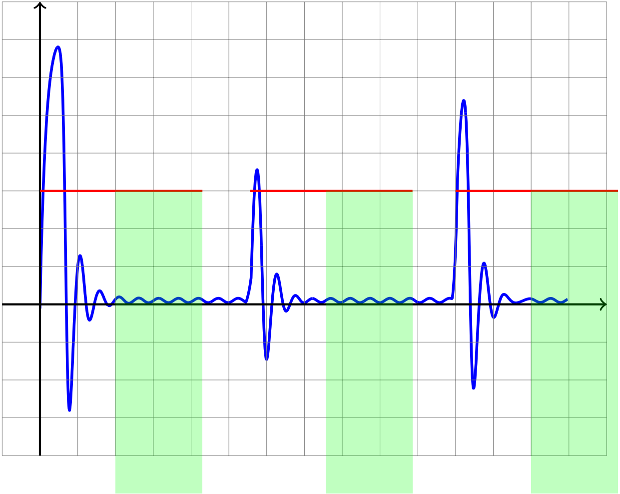
- Multiple solutions: equivalent to multi-objective problem
- We require tight specifications (avoid over-conservatism)

Example



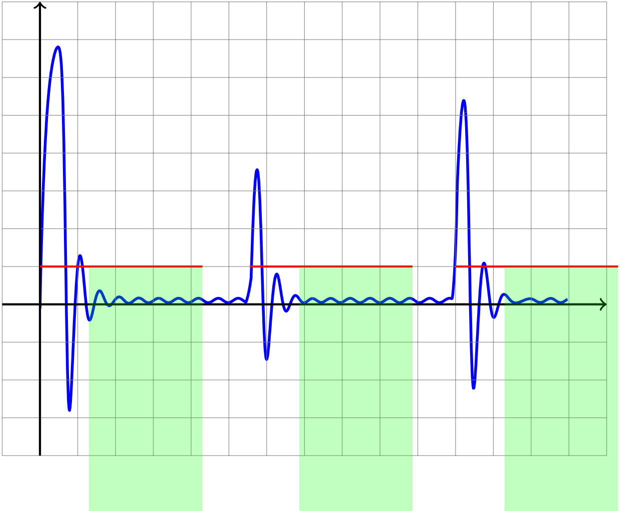
Example

$$\varphi := \mathsf{alw}(x[t] > \pi \rightarrow \ \mathsf{ev}_{[0,\tau_1]} \ \ (\ \mathsf{alw}_{[0,\tau_2]} \ x[t] < \pi))$$



Example

$$\varphi := \mathsf{alw}(x[t] > \pi_1 \rightarrow \; \mathsf{ev}_{[0,\tau_1]} \; (\; \mathsf{alw}_{[0,\tau_2]} \; x[t] < \pi_2))$$



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 $\mathbf{x}$,

$$\forall v, v', \mathbf{x} \models \varphi(\ldots, v(p_i), \ldots),$$

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

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

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 $\mathbf{x}$,

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

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:

$$u(t) \longrightarrow \boxed{\textit{System } \mathcal{S}} \longrightarrow \mathcal{S}(u(t))$$

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

Falsification problem

Problem

Given the system:

$$u(t) \longrightarrow \boxed{\textit{System } \mathcal{S}} \longrightarrow \mathcal{S}(u(t))$$

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

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

Falsification problem

Problem

Given the system:

$$u(t) \longrightarrow \boxed{\textit{System } \mathcal{S}} \longrightarrow \mathcal{S}(u(t))$$

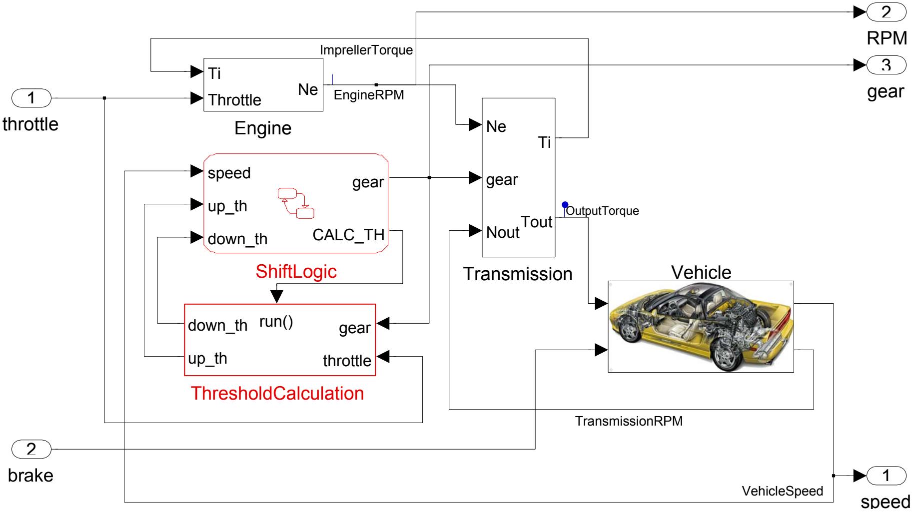
Find an input signal $u \in \mathcal{U}$ such that $S(u(t)), 0 \not\models \varphi$

Approach: Solve $\rho^* = \min_{u \in \mathcal{U}} \rho(\varphi, \mathcal{S}(u), 0)$ (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

Automatic Transmission System



Formulas

- *the* speed *is always below* π_1 *and* RPM *below* π_2

$$\varphi_{\texttt{sp\_rpm}}(\pi_1, \pi_2) := \mathsf{alw}\,(\ (\texttt{speed} < \pi_1) \wedge (\texttt{RPM} < \pi_2)\ )\,.$$

- *the vehicle cannot reach 100 mph in* τ *seconds with* RPM *always below* π

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

- *whenever it shift to gear* 2*, it dwells in gear* 2 *for at least* τ *seconds*

$$\varphi_{\texttt{stay}}(\tau) := \mathsf{alw}\left(\left(\begin{array}{c}\texttt{gear} \neq 2\ \wedge \\ \mathsf{ev}_{[0,\varepsilon]}\ \texttt{gear} = 2\end{array}\right) \Rightarrow \mathsf{alw}_{[\varepsilon,\tau]}\texttt{gear} = 2\right).$$

Results

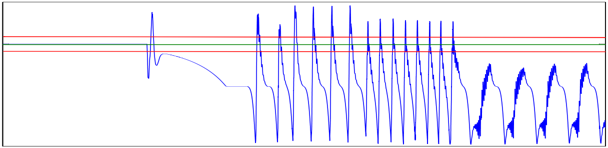
Template	S-Taliro-based falsification				
	Parameter values	Fals.	Synth.	#Sim.	Sat./$\mathbf{x}$
$\varphi_{\mathtt{sp\_rpm}}(\pi_1, \pi_2)$	(155 mph, 4858 rpm)	55 s	12 s	255	0.004 s
$\varphi_{\mathtt{rpm100}}(\pi, \tau)$	(3278.3 rpm, 49.91 s)	6422 s	26.5 s	9519	0.327 s
$\varphi_{\mathtt{rpm100}}(\tau, \pi)$	(4997 rpm, 12.20 s)	8554 s	53.8 s	18284	0.149 s
$\varphi_{\mathtt{stay}}(\pi)$	1.79 s	18886 s	0.868 s	130	147.2 s

Template	Breach-based falsification				
	Parameter values	Fals.	Synth.	#Sim.	Sat./$\mathbf{x}$
$\varphi_{\mathtt{sp\_rpm}}(\pi_1, \pi_2)$	(155 mph, 4858 rpm)	197.2 s	23.1 s	496	0.043 s
$\varphi_{\mathtt{rpm100}}(\pi, \tau)$	(3278.3 rpm, 49.91 s)	267.7 s	10.51 s	709	0.026 s
$\varphi_{\mathtt{rpm100}}(\tau, \pi)$	(4997 rpm, 12.20 s)	147.8 s	5.188 s	411	0.021 s
$\varphi_{\mathtt{stay}}(\pi)$	1.79 s	430.9 s	2.157 s	1015	0.032 s

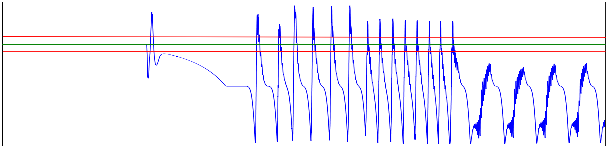
Results on Industrial-scale Model

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_{settle} = 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 speficition mining
- More elaborate templates mining (beyond parameters)
- How to help designers writing and using temporal logics templates and formulas ?

Thanks !