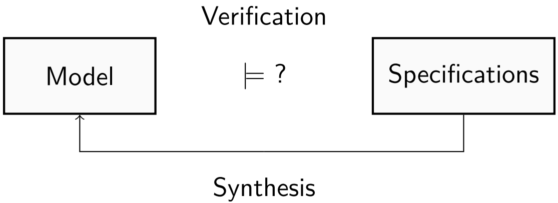
Parameter Synthesis for Signal Temporal Logic

Alexandre Donzé

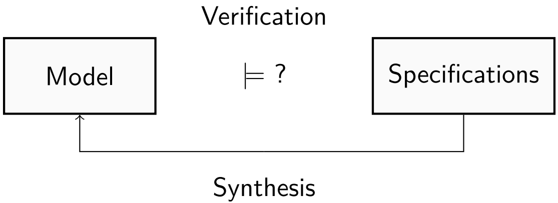
University of California, Berkeley

April 7, 2014

Formal methods and parameter synthesis



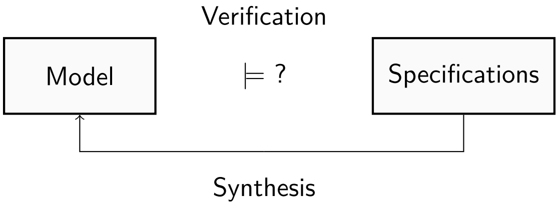
Formal methods and parameter synthesis



Problematics

- For complex systems (large-scale, hybrid dynamics), synthesis is intractable

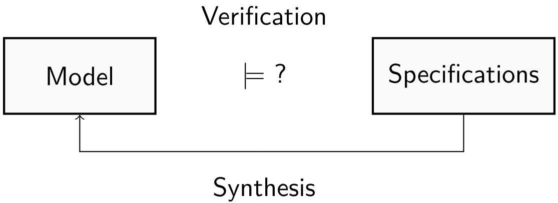
Formal methods and parameter synthesis



Problematics

- For complex systems (large-scale, hybrid dynamics), synthesis is intractable
- Parameter synthesis reduces to finding valid values for "a few" parameters

Formal methods and parameter synthesis

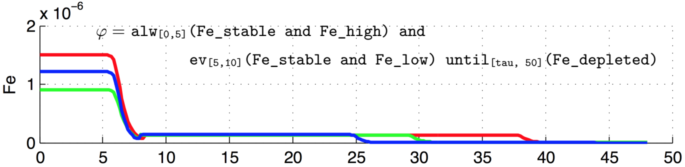


Problematics

- For complex systems (large-scale, hybrid dynamics), synthesis is intractable
- Parameter synthesis reduces to finding valid values for "a few" parameters
- We consider here *model parameters* and *specification parameters*

Example [1]: modeling iron homeostasis

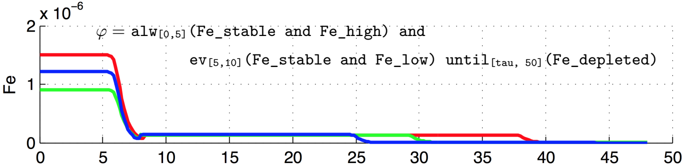
- Specifications
  Qualitative knowledge, quantitative measurements, partially formalizable



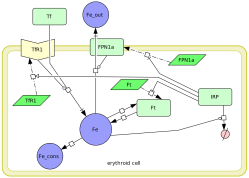
[1] (joint work with N. Mobilia, E. Fanchon, J-M Moulis et al)

Example [1]: modeling iron homeostasis

- Specifications
  Qualitative knowledge, quantitative measurements, partially formalizable



- Model

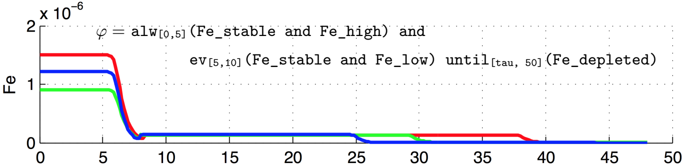


$$\frac{d}{dt}Fe = k_1\,TfR1\;\;Tf - k_2 Fe\;\,FPN1a + k_3 Fe$$

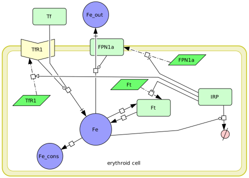
[1] (joint work with N. Mobilia, E. Fanchon, J-M Moulis et al)

Example [1]: modeling iron homeostasis

- **Specifications**
  Qualitative knowledge, quantitative measurements, partially formalizable



- **Model**



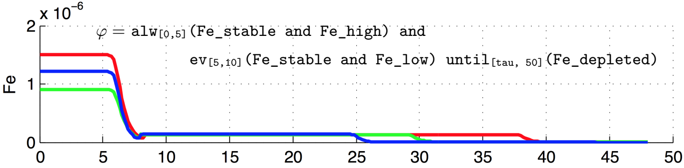
$$\frac{d}{dt}Fe = k_1\, TfR1\ \ Tf - k_2 Fe\ FPN1a + k_3 Fe$$

Problem: values for k_1, k_2, k_3, etc

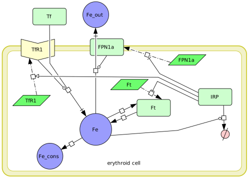
[1](joint work with N. Mobilia, E. Fanchon, J-M Moulis et al)

Example [1]: modeling iron homeostasis

- Specifications
  Qualitative knowledge, quantitative measurements, partially formalizable



- Model



$$\frac{d}{dt}Fe = k_1\, TfR1\ \ Tf - k_2 Fe\ FPN1a + k_3 Fe$$

Problem: values for k_1, k_2, k_3, etc

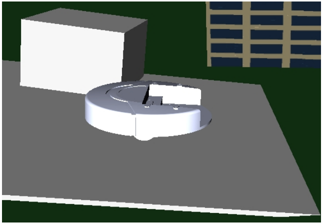
$\Rightarrow$ synthesis of *model* parameters

[1] (joint work with N. Mobilia, E. Fanchon, J-M Moulis et al)

Example [2]: faulty behaviors of a robot

Goal: autograding a robotic lab

Assignement: climb hills+avoid obstacles

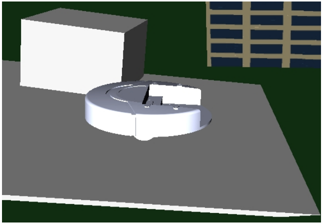


[2] (joint work with G. Juniwal, J. C. Jensen, S. A. Seshia)

Example [2]: faulty behaviors of a robot

Goal: autograding a robotic lab

Assignement: climb hills+avoid obstacles



Faulty behavior specifications

E.g.: *"The robot does not reach the top of the hill in τ seconds"*

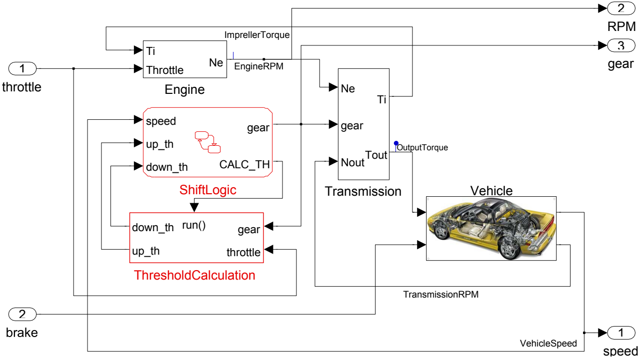
What is a value of τ that discriminate faulty from acceptable solutions ?

$\Rightarrow$ synthesis of *specification* parameters

[2](joint work with G. Juniwal, J. C. Jensen, S. A. Seshia)

Example [3]: specification mining

Design of an automatic transmission system:



- What is the maximum speed that the vehicule can reach ?
- What is the minimum dwell time in a given gear ?
- etc

⇒ synthesis of both *specification* and *model* parameters

[3](joint work with X. Jing, J. Deshmuhk, S.A. Seshia)

Outline

1. Preliminaries: Signal Temporal Logic
   - From LTL to STL
   - Robust semantics
2. Parameter synthesis
   - Property parameters
   - Model parameters
3. Putting it all together: specification mining

Outline

1. Preliminaries: Signal Temporal Logic
   - From LTL to STL
   - Robust semantics
2. Parameter synthesis
   - Property parameters
   - Model parameters
3. Putting it all together: specification mining

Temporal logics in a nutshell

Temporal logics 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" (G), "eventually" (F) and "until" ($\mathcal{U}$)

Linear Temporal Logic

An LTL formula φ is evaluated on a sequence, e.g., $w = aaabbaaa\ldots$

At each step of w, we can define a truth value of φ, noted $\chi^\varphi(w,i)$

LTL atoms are symbols: a, b:

$$\begin{array}{rccccccccc} i = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \ldots \\ w = & a & a & a & b & b & a & a & a & \ldots \\ \chi^a(w,i) = & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & \ldots \\ \chi^b(w,i) = & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & \ldots \end{array}$$

LTL, temporal operators

$\bigcirc$ ("next"), G ("globally"), F ("eventually") and $\mathbf{U}$ ("until").

They are evaluated at each step wrt the future of sequences

		$w =$	a	a	a	b	b	a	a	a	$\dots$
$\bigcirc b$	(next)	$\chi^{\bigcirc b}(w,i) =$	0	0	1	1	0	0	0	?	$\dots$
G a	(always)	$\chi^{\mathsf{G} a}(w,i) =$	0	0	0	0	0	1?	1?	1?	$\dots$
F b	(eventually)	$\chi^{\mathsf{F} b}(w,i) =$	1	1	1	1	1	0?	0?	0?	$\dots$
a $\mathbf{U}$ b	(until)	$\chi^{a \mathbf{U} b}(w,i) =$	1	1	1	0	0	0?	0?	0?	$\dots$

LTL, temporal operators

$\bigcirc$ ("next"), G ("globally"), F ("eventually") and $\mathbf{U}$ ("until").

They are evaluated at each step wrt the future of sequences

		$w =$	a	a	a	b	b	a	a	a	$\ldots$
$\bigcirc b$	(next)	$\chi^{\bigcirc b}(w,i) =$	0	0	1	1	0	0	0	?	$\ldots$
G a	(always)	$\chi^{\mathsf{G}a}(w,i) =$	0	0	0	0	0	1?	1?	1?	$\ldots$
F b	(eventually)	$\chi^{\mathsf{F}b}(w,i) =$	1	1	1	1	1	0?	0?	0?	$\ldots$
a $\mathbf{U}$ b	(until)	$\chi^{a\mathbf{U}b}(w,i) =$	1	1	1	0	0	0?	0?	0?	$\ldots$

LTL, temporal operators

$\bigcirc$ ("next"), G ("globally"), F ("eventually") and $\mathbf{U}$ ("until").

They are evaluated at each step wrt the future of sequences

		$w =$	a	a	a	b	b	a	a	a	$\dots$
$\bigcirc b$	(next)	$\chi^{\bigcirc b}(w,i) =$	0	0	1	1	0	0	0	0?	$\dots$
G a	(always)	$\chi^{\mathsf{G} a}(w,i) =$	0	0	0	0	0	1?	1?	1?	$\dots$
F b	(eventually)	$\chi^{\mathsf{F} b}(w,i) =$	1	1	1	1	1	0?	0?	0?	$\dots$
a $\mathbf{U}$ b	(until)	$\chi^{a \mathbf{U} b}(w,i) =$	1	1	1	0	0	0?	0?	0?	$\dots$

LTL, temporal operators

$\bigcirc$ ("next"), G ("globally"), F ("eventually") and $\mathbf{U}$ ("until").

They are evaluated at each step wrt the future of sequences

		$w =$	a	a	a	b	b	a	a	a	$\dots$
$\bigcirc b$	(next)	$\chi^{\bigcirc b}(w,i) =$	0	0	1	1	0	0	0	?	$\dots$
G a	(always)	$\chi^{\mathsf{G} a}(w,i) =$	0	0	0	0	0	1?	1?	1?	$\dots$
F b	(eventually)	$\chi^{\mathsf{F} b}(w,i) =$	1	1	1	1	1	0?	0?	0?	$\dots$
a $\mathbf{U}$ b	(until)	$\chi^{a \mathbf{U} b}(w,i) =$	1	1	1	0	0	0?	0?	0?	$\dots$

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

Ex: request-grant property

LTL G(r => F g)

Boolean predicates, discrete-time

From LTL to STL

Extension of LTL with real-time and real-valued constraints

Ex: request-grant property

LTL G(r => F g)
Boolean predicates, discrete-time

MTL G(r => $F_{[0,.5s]}$ g)
Boolean predicates, real-time

From LTL to STL

Extension of LTL with real-time and real-valued constraints

Ex: request-grant property

LTL G(r => F g)
Boolean predicates, discrete-time

MTL G(r => $\mathsf{F}_{[0,.5s]}$ g)
Boolean predicates, real-time

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

STL syntax

MTL/STL Formulas

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

- $\bot = \neg\top$
- Eventually is $\mathsf{F}_{[a,b]}\ \varphi = \top\ \mathcal{U}_{[a,b]}\ \varphi$
- Always is $\mathsf{G}_{[a,b]}\varphi = \neg(\ \mathsf{F}_{[a,b]}\ \neg\varphi)$

STL syntax

MTL/STL Formulas

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

- $\bot = \neg\top$
- Eventually is $\mathsf{F}_{[a,b]} \; \varphi = \top \; \mathcal{U}_{[a,b]} \; \varphi$
- Always is $\mathsf{G}_{[a,b]}\varphi = \neg(\; \mathsf{F}_{[a,b]} \; \neg\varphi)$

STL Predicates

STL adds an analog layer to MTL. Assume signals $x_1[t], \; x_2[t], \; \ldots, x_n[t]$, then atomic predicates are of the form:

$$\mu = f(x_1[t], \ldots, x_n[t]) > 0$$

STL semantics

The satisfaction of a formula φ by a signal $\mathbf{x} = (x_1, \ldots, x_n)$ at time t is

$$
\begin{array}{lcl}
(\mathbf{x}, t) \models \mu & \Leftrightarrow & f(x_1[t], \ldots, x_n[t]) > 0 \\
(\mathbf{x}, t) \models \varphi \wedge \psi & \Leftrightarrow & (x, t) \models \varphi \wedge (x, t) \models \psi \\
(\mathbf{x}, t) \models \neg\varphi & \Leftrightarrow & \neg((x, t) \models \varphi) \\
(\mathbf{x}, t) \models \varphi \, \mathcal{U}_{[a,b]} \, \psi & \Leftrightarrow & \exists t' \in [t + a, t + b] \text{ such that } (x, t') \models \psi \, \wedge \\
 & & \qquad \forall t'' \in [t, t'], \; (x, t'') \models \varphi \}
\end{array}
$$

STL semantics

The satisfaction of a formula φ by a signal $\mathbf{x} = (x_1, \ldots, x_n)$ at time t is

$$
\begin{array}{lcl}
(\mathbf{x}, t) \models \mu & \Leftrightarrow & f(x_1[t], \ldots, x_n[t]) > 0 \\
(\mathbf{x}, t) \models \varphi \wedge \psi & \Leftrightarrow & (x, t) \models \varphi \wedge (x, t) \models \psi \\
(\mathbf{x}, t) \models \neg\varphi & \Leftrightarrow & \neg((x, t) \models \varphi) \\
(\mathbf{x}, t) \models \varphi \, \mathcal{U}_{[a,b]} \, \psi & \Leftrightarrow & \exists t' \in [t+a, t+b] \text{ such that } (x, t') \models \psi \, \wedge \\
& & \qquad \forall t'' \in [t, t'], \; (x, t'') \models \varphi \}
\end{array}
$$

- Eventually is $\mathsf{F}_{[a,b]} \, \varphi = \top \, \mathcal{U}_{[a,b]} \, \varphi$

$$(\mathbf{x}, t) \models \mathsf{F}_{[a,b]} \, \psi \Leftrightarrow \exists t' \in [t+a, t+b] \text{ such that } (x, t') \models \psi$$

STL semantics

The satisfaction of a formula φ by a signal $\mathbf{x} = (x_1, \ldots, x_n)$ at time t is

$$
\begin{array}{lcl}
(\mathbf{x}, t) \models \mu & \Leftrightarrow & f(x_1[t], \ldots, x_n[t]) > 0 \\
(\mathbf{x}, t) \models \varphi \wedge \psi & \Leftrightarrow & (x, t) \models \varphi \wedge (x, t) \models \psi \\
(\mathbf{x}, t) \models \neg\varphi & \Leftrightarrow & \neg((x, t) \models \varphi) \\
(\mathbf{x}, t) \models \varphi\, \mathcal{U}_{[a,b]}\, \psi & \Leftrightarrow & \exists t' \in [t+a, t+b] \text{ such that } (x, t') \models \psi \wedge \\
 & & \qquad \forall t'' \in [t, t'],\ (x, t'') \models \varphi\}
\end{array}
$$

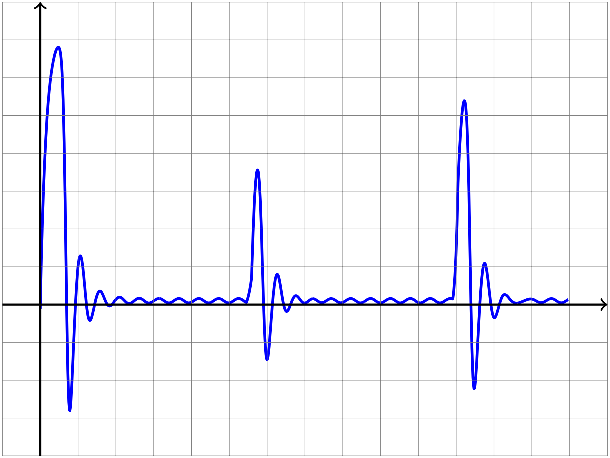
- Eventually is $\mathsf{F}_{[a,b]}\ \varphi = \top\ \mathcal{U}_{[a,b]}\ \varphi$

$$(\mathbf{x}, t) \models \mathsf{F}_{[a,b]}\ \psi \Leftrightarrow \exists t' \in [t+a, t+b] \text{ such that } (x, t') \models \psi$$

- Always is $\mathsf{G}_{[a,b]}\varphi = \neg(\ \mathsf{F}_{[a,b]}\ \neg\varphi)$

$$(\mathbf{x}, t) \models \mathsf{G}_{[a,b]}\psi \Leftrightarrow \forall t' \in [t+a, t+b] \text{ such that } (x, t') \models \psi$$

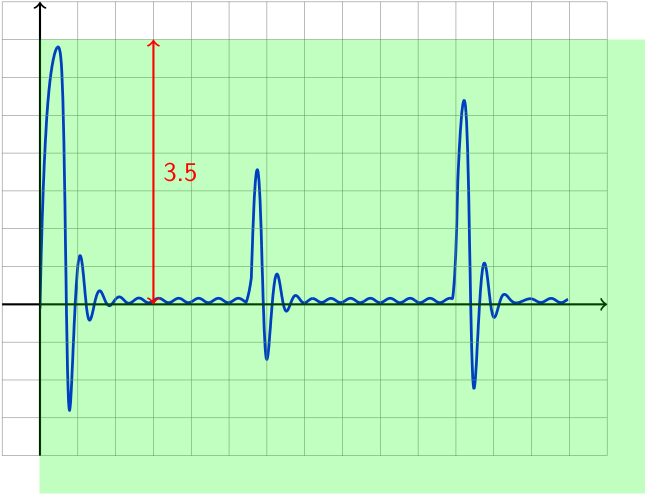
STL examples



STL examples

The signal is never above 3.5

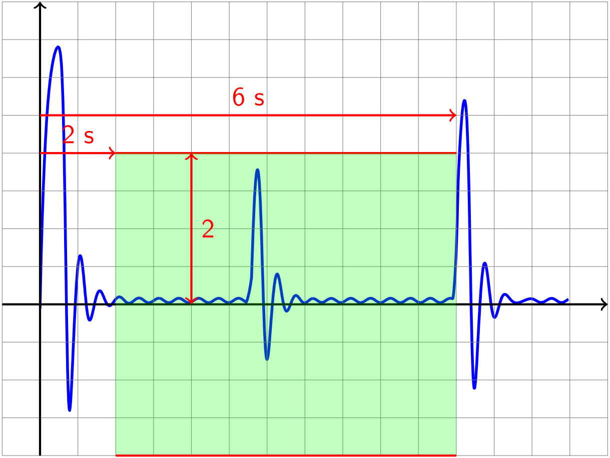
$$\varphi := \mathsf{G}\ (x[t] < 3.5)$$



STL examples

Between 2s and 6s the signal is between -2 and 2

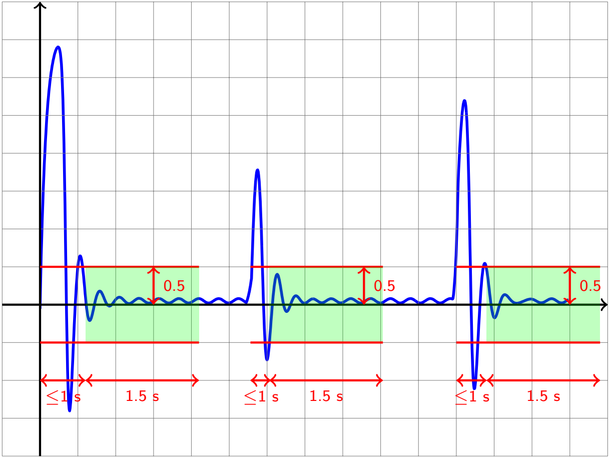
$$\varphi := \ \mathsf{G}_{[2,6]} \ \ (|x[t]| < 2)$$



STL examples

Always $|x| > 0.5 \Rightarrow$ *after 1 s,* $|x|$ *settles under 0.5 for 1.5 s*

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



Outline

1. Preliminaries: Signal Temporal Logic
   - From LTL to STL
   - Robust semantics
2. Parameter synthesis
   - Property parameters
   - Model parameters
3. Putting it all together: specification mining

STL semantics

The satisfaction of a formula φ by a signal $\mathbf{x} = (x_1, \ldots, x_n)$ at time t is

$$\begin{array}{lcl} (\mathbf{x}, t) \models \mu & \Leftrightarrow & f(x_1[t], \ldots, x_n[t]) > 0 \\ (\mathbf{x}, t) \models \varphi \wedge \psi & \Leftrightarrow & (x, t) \models \varphi \wedge (x, t) \models \psi \\ (\mathbf{x}, t) \models \neg\varphi & \Leftrightarrow & \neg((x, t) \models \varphi) \\ (\mathbf{x}, t) \models \varphi \,\mathcal{U}_{[a,b]}\, \psi & \Leftrightarrow & \exists t' \in [t+a, t+b] \text{ such that } (x, t') \models \psi \wedge \\ & & \qquad \forall t'' \in [t, t'],\ (x, t'') \models \varphi\} \end{array}$$

- Eventually is $\mathsf{F}_{[a,b]}\ \varphi = \top\ \mathcal{U}_{[a,b]}\ \varphi$

$$(\mathbf{x}, t) \models \mathsf{F}_{[a,b]}\ \psi \Leftrightarrow \exists t' \in [t+a, t+b] \text{ such that } (x, t') \models \psi$$

- Always is $\mathsf{G}_{[a,b]}\varphi = \neg(\mathsf{F}_{[a,b]}\ \neg\varphi)$

$$(\mathbf{x}, t) \models \mathsf{G}_{[a,b]}\psi \Leftrightarrow \forall t' \in [t+a, t+b] \text{ such that } (x, t') \models \psi$$

STL satisfaction function

The semantics can be defined as function $\chi^\varphi(x, t)$ such that:

$$x, t \models \varphi \Leftrightarrow \chi^\varphi(x, t) = \top$$

STL satisfaction function

The semantics can be defined as function $\chi^{\varphi}(x,t)$ such that:

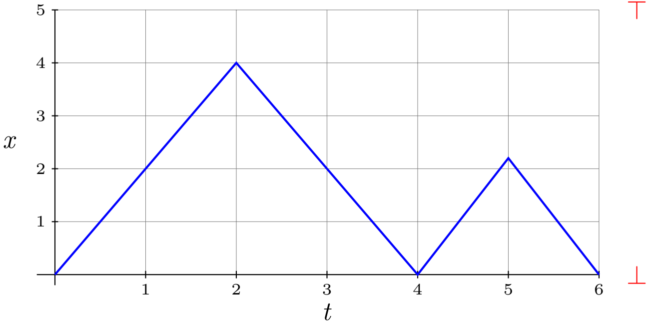
$$x, t \models \varphi \Leftrightarrow \chi^{\varphi}(x,t) = \top$$

Considering Booleans $(\mathbb{B}, <, -)$ as an order with involution:

$$\begin{aligned}
\chi^{\mu}(x,t) &= f(x_1[t], \ldots, x_n[t]) > 0 \\
\chi^{\neg\varphi}(x,t) &= -\chi^{\varphi}(x,t) \\
\chi^{\varphi_1 \wedge \varphi_2}(x,t) &= \min(\chi^{\varphi_1}(x,t), \chi^{\varphi_2}(w,t)) \\
\chi^{\varphi_1 \ \mathcal{U}_{[a,b]} \ \varphi_2}(x,t) &= \max_{\tau \in t+[a,b]}(\min(\chi^{\varphi_2}(x,\tau), \ \min_{s \in [t,\tau]} \chi^{\varphi_1}(x,s))
\end{aligned}$$

STL satisfaction function example

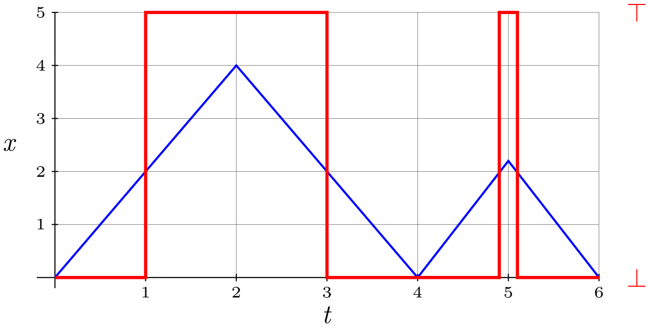
Consider a simple piecewise affine signal:



Satisfaction signal of :

STL satisfaction function example

Consider a simple piecewise affine signal:

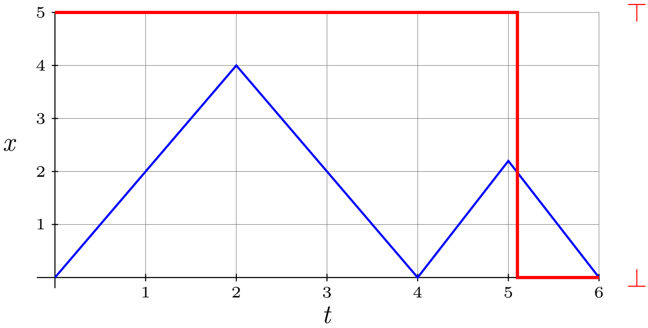


Satisfaction signal of :

- $\varphi = x \geq 2$

STL satisfaction function example

Consider a simple piecewise affine signal:

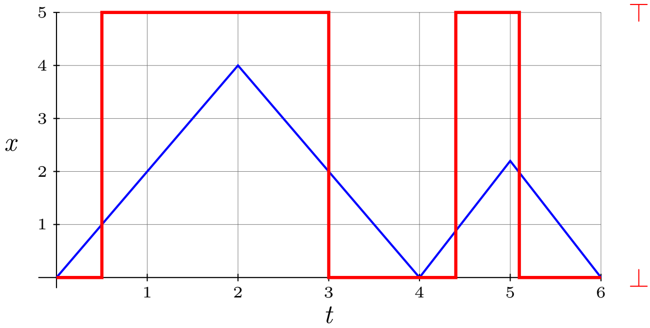


Satisfaction signal of :

- $\varphi = \mathbf{F}(x \geq 2)$

STL satisfaction function example

Consider a simple piecewise affine signal:



Satisfaction signal of :

- $\varphi = \mathbf{F}_{[0,0.5]}(x \geq 2)$

Robust satisfaction signal

The Reals $(\mathbb{R}, <, -)$ also form an order with involution:

$$
\begin{aligned}
\rho^{\mu}(x,t) &= f(x_1[t],\ldots,x_n[t]) \\
\rho^{\neg\varphi}(x,t) &= -\rho^{\varphi}(x,t) \\
\rho^{\varphi_1\wedge\varphi_2}(x,t) &= \min(\rho^{\varphi_1}(x,t),\rho^{\varphi_2}(w,t)) \\
\rho^{\varphi_1\ \mathcal{U}_{[a,b]}\ \varphi_2}(x,t) &= \sup_{\tau\in t+[a,b]}(\min(\rho^{\varphi_2}(x,\tau),\ \inf_{s\in[t,\tau]}\rho^{\varphi_1}(x,s))
\end{aligned}
$$

Properties of robust satisfaction signal

- Sign indicates satisfaction status

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

Properties of robust satisfaction signal

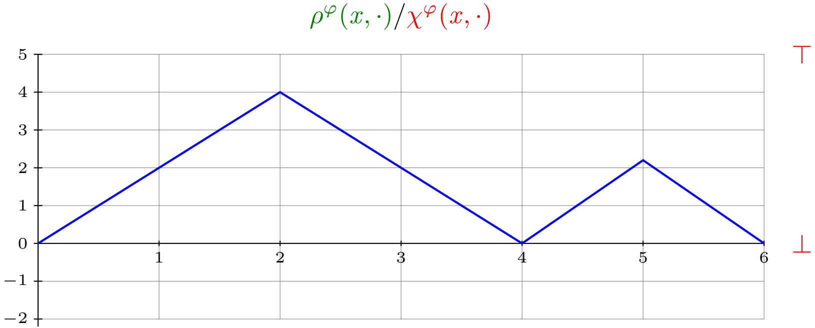
- Sign indicates satisfaction status

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

- Absolute value indicates tolerance

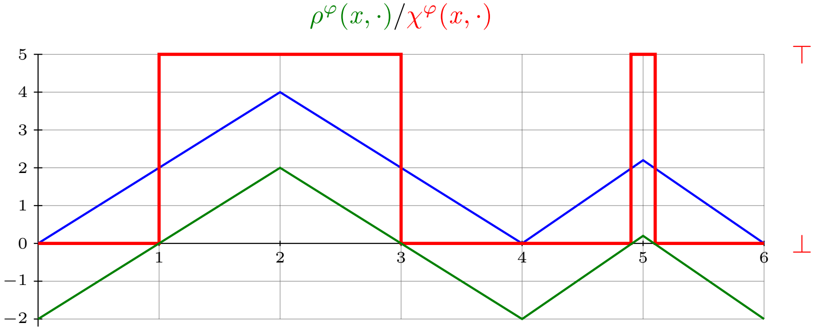
$$x,t \models \varphi \text{ and } \|x - x'\|_{\infty} \leq \rho^{\varphi}(x,t) \quad \Rightarrow \quad x',t \models \varphi$$
$$x,t \not\models \varphi \text{ and } \|x - x'\|_{\infty} \leq -\rho^{\varphi}(x,t) \quad \Rightarrow \quad x',t \not\models \varphi$$

STL satisfaction function example



Satisfaction signals of :

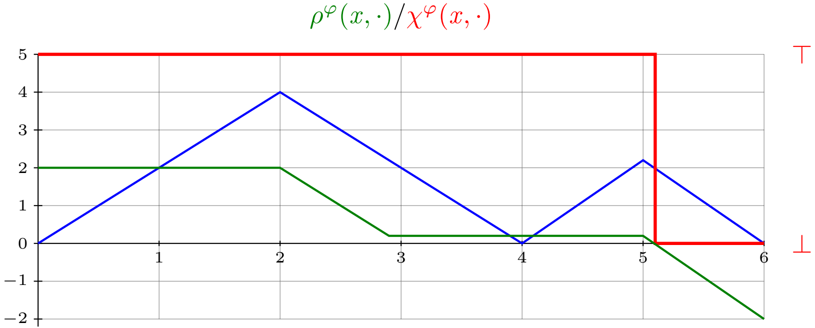
STL satisfaction function example



Satisfaction signals of :

- $\varphi = x \geq 2$

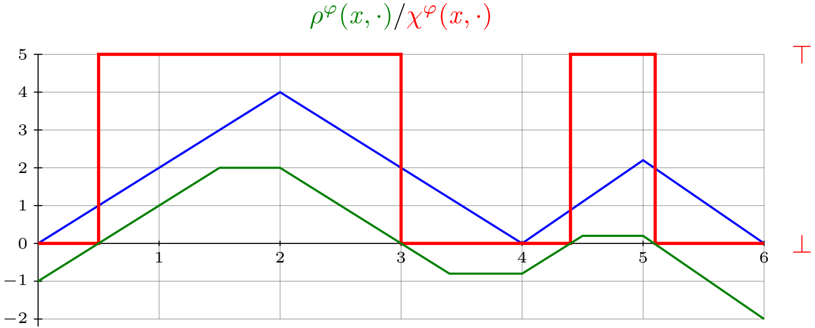
STL satisfaction function example



Satisfaction signals of :

- $\varphi = \mathbf{F}(x \geq 2)$

STL satisfaction function example



Satisfaction signals of :

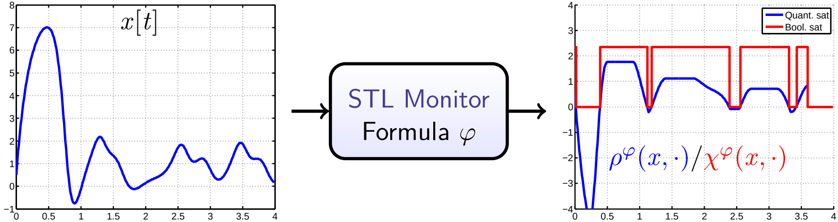
- $\varphi = \mathbf{F}_{[0,0.5]}(x \geq 2)$

Robust monitoring

A robust STL monitor is a *transducer* that transform x into $\rho^{\varphi}(x, .)$

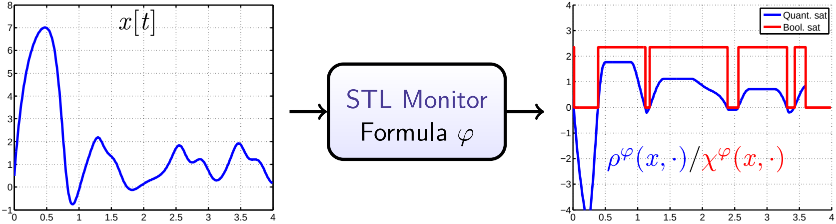
Robust monitoring

A robust STL monitor is a *transducer* that transform x into $\rho^{\varphi}(x,.)$



Robust monitoring

A robust STL monitor is a *transducer* that transform x into $\rho^{\varphi}(x, .)$



In practice

- Trace: time words over alphabet $\mathbb{R}$, linear interpolation

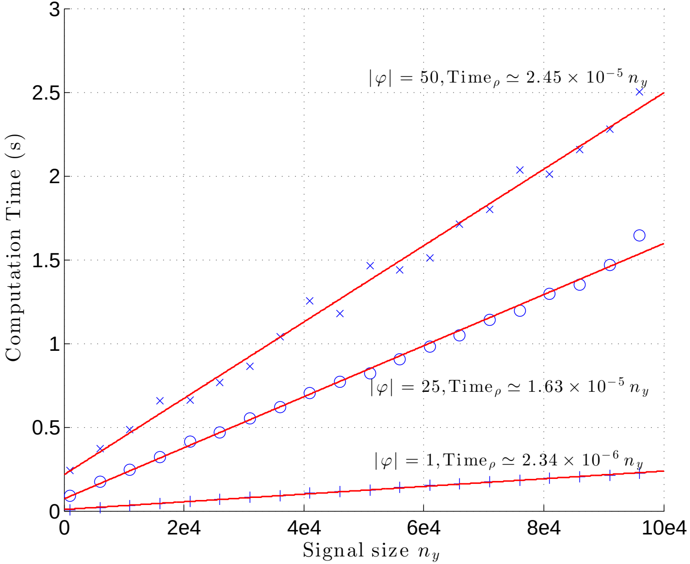
  Input: $x(\cdot) \triangleq (t_i, x(t_i))_{i\in\mathbb{N}}$ Output: $\rho^{\varphi}(x,\cdot) \triangleq (r_j, z(r_j))_{j\in\mathbb{N}}$
- Continuity, and piecewise affine property preserved

Computing the robust satisfaction function

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

- The function $\rho^{\varphi}(x, t)$ 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
- Atomic transducers compute in linear time in the size of the input
  - Key idea is to exploit efficient streaming algorithm (Lemire's) computing the max and min over a moving window

Performance results



Parametric STL

Informally, a PSTL formula is an STL formula where (some) numeric constants are left unspecified, represented by symbolic parameters.

Definition (PSTL syntax)

$$\varphi := \mu(x[t]) > \pi \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \, \mathbf{U}_{[\tau_1,\tau_2]} \, \psi$$

where

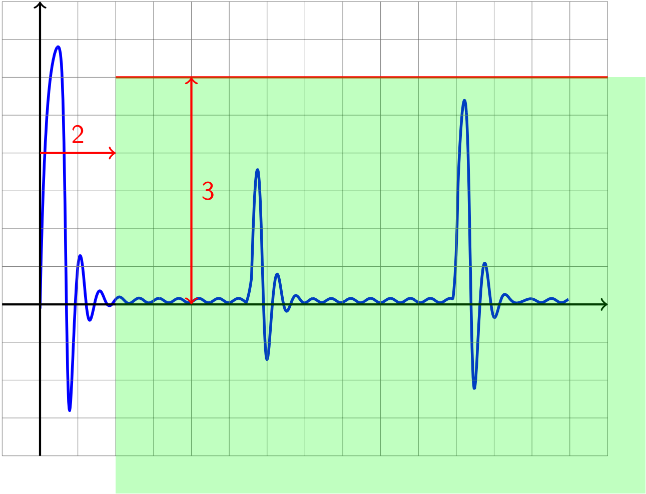
- π is a scale parameter
- τ_1, τ_2 are time parameters

Parametric STL

Parametric STL

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

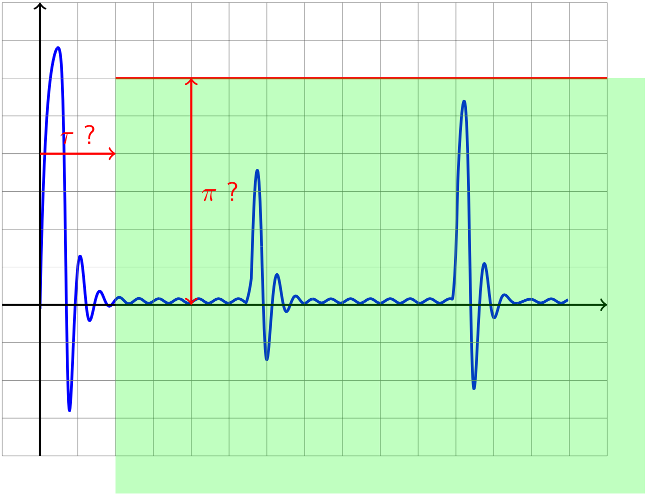
$$\varphi := \mathsf{F}_{[2,\infty]} \ \ (x[t] < 3)$$



Parametric STL

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

$$\varphi := \mathsf{G}_{[\tau,\infty]} \; (x[t] < \pi)$$



Parameter synthesis for PSTL

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

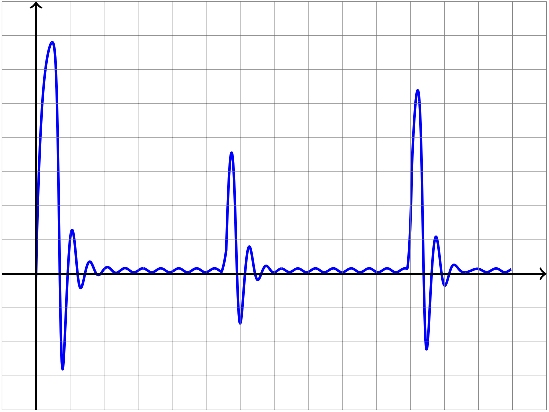
$$x, t \models \varphi(v(p_1), \ldots, v(p_n)),$$

Informally, a valuation v is tight if there exists a valuation v' in a δ-close neighborhood of v, with δ "small", such that

$$x, t \not\models \varphi(v'(p_1), \ldots, v'(p_n))$$

Example

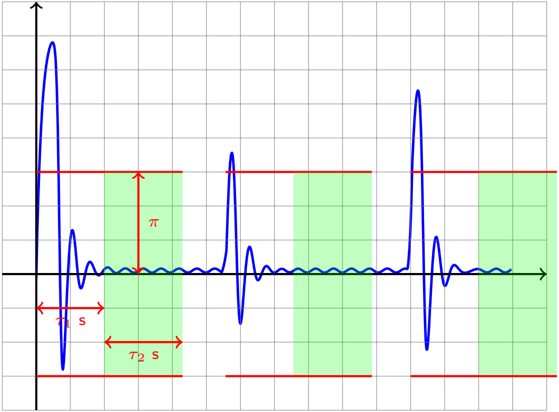
$\varphi := \mathsf{G}\left(x[t] > \pi \rightarrow \ \mathsf{F}_{[0,\tau_1]} \ \ (\ \mathsf{G}_{[0,\tau_2]} \ x[t] < \pi)\right)$



Example

$\varphi := \mathsf{G}\left(x[t] > \pi \rightarrow \mathsf{F}_{[0,\tau_1]} \left(\mathsf{G}_{[0,\tau_2]}\ x[t] < \pi\right)\right)$

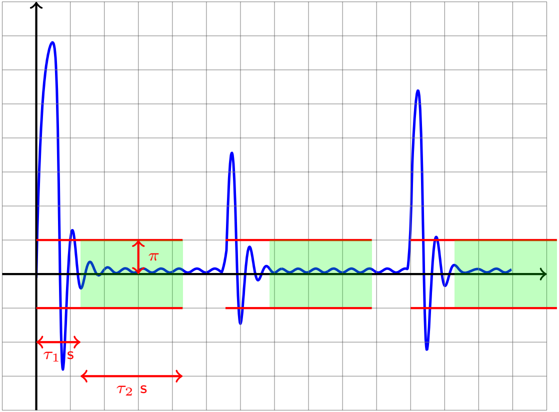
- Valuation 1: $\pi \leftarrow 1.5$, $\tau_1 \leftarrow 1\ s$, $\tau_2 \leftarrow 1.15\ s$



Example

$\varphi := \mathsf{G}\left(x[t] > \pi \rightarrow \ \mathsf{F}_{[0,\tau_1]} \ \ (\ \mathsf{G}_{[0,\tau_2]} \ x[t] < \pi)\right)$

- Valuation 1: $\pi \leftarrow 1.5$, $\tau_1 \leftarrow 1\ s$, $\tau_2 \leftarrow 1.15\ s$
- Valuation 2 (tight): $\pi \leftarrow .5$, $\tau_1 \leftarrow 0.65\ s$, $\tau_2 \leftarrow 2\ s$



Parameter synthesis

Challenges

- Multiple solutions: which one to chose ?
- Tightness implies to "optimize" the valuation $v(p_i)$ for each p_i

The problem can be greatly simplified if the formula is *monotonic* in each p_i.

Parameter synthesis

Challenges

- Multiple solutions: which one to chose ?
- Tightness implies to "optimize" the valuation $v(p_i)$ for each p_i

The problem can be greatly simplified if the formula is *monotonic* in each p_i.

Definition

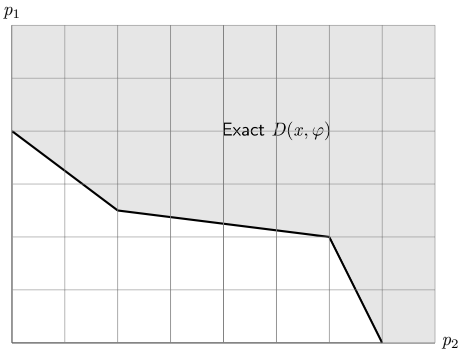
A PSTL formula $\varphi(p_1, \cdots, p_n)$ is monotonically increasing wrt p_i if

$$\forall \mathbf{x}, v, v', \left(\begin{array}{l} \mathbf{x} \models \varphi(v(p_1), \ldots, v(p_i), \ldots) \\ v(p_j) = v'(p_j), j \neq i \\ v'(p_i) \geq v(p_i) \end{array} \right) \Rightarrow \mathbf{x} \models \varphi(v'(p_1), \ldots, v'(p_i), \ldots)$$

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

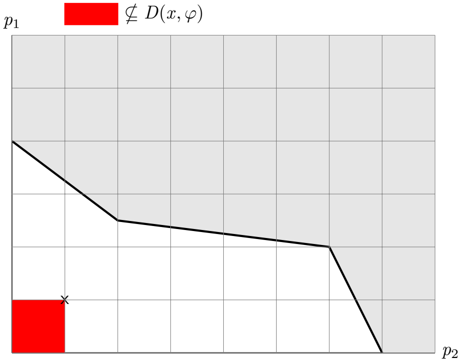
Monotonic validity domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- A tight valuation is a valuation in D close to its boundary ∂D
- In case of monoticity, ∂D has the structure of a Pareto front which can be estimated with generalized binary search heuristics



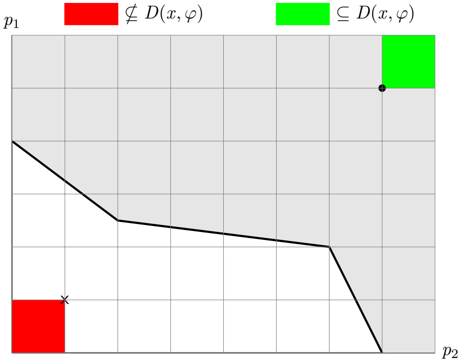
Monotonic validity domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- A tight valuation is a valuation in D close to its boundary ∂D
- In case of monoticity, ∂D has the structure of a Pareto front which can be estimated with generalized binary search heuristics



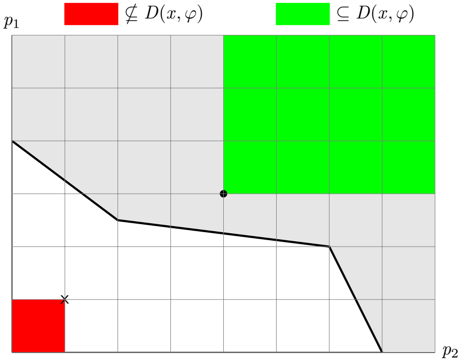
Monotonic validity domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- A tight valuation is a valuation in D close to its boundary ∂D
- In case of monoticity, ∂D has the structure of a Pareto front which can be estimated with generalized binary search heuristics



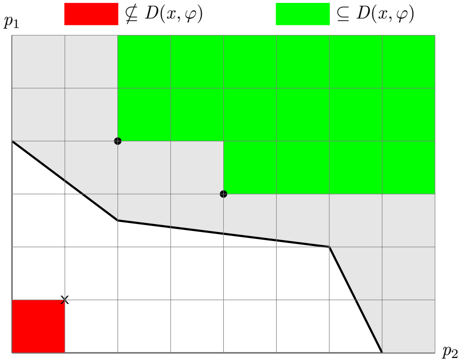
Monotonic validity domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- A tight valuation is a valuation in D close to its boundary ∂D
- In case of monoticity, ∂D has the structure of a Pareto front which can be estimated with generalized binary search heuristics



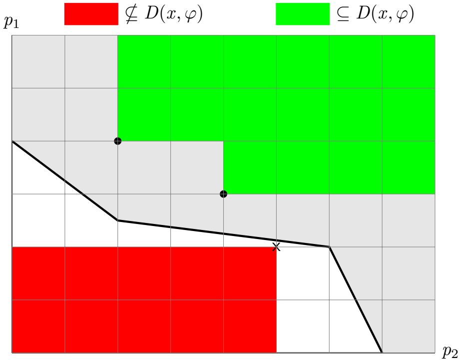
Monotonic validity domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- A tight valuation is a valuation in D close to its boundary ∂D
- In case of monoticity, ∂D has the structure of a Pareto front which can be estimated with generalized binary search heuristics



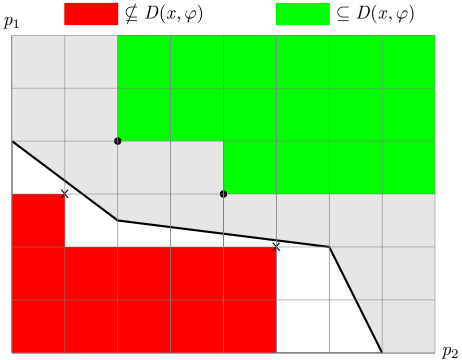
Monotonic validity domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- A tight valuation is a valuation in D close to its boundary ∂D
- In case of monoticity, ∂D has the structure of a Pareto front which can be estimated with generalized binary search heuristics



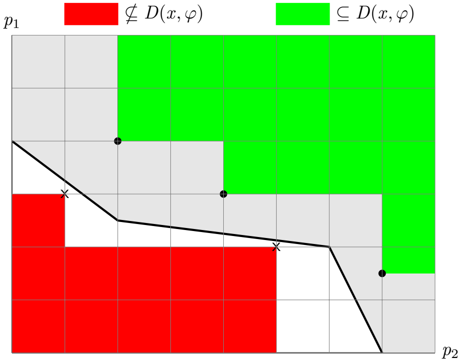
Monotonic validity domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- A tight valuation is a valuation in D close to its boundary ∂D
- In case of monoticity, ∂D has the structure of a Pareto front which can be estimated with generalized binary search heuristics



Monotonic validity domains

- The validity domain D of φ and x is the set of valuations v s.t. $x \models \varphi(v)$
- A tight valuation is a valuation in D close to its boundary ∂D
- In case of monoticity, ∂D has the structure of a Pareto front which can be estimated with generalized binary search heuristics



Deciding monotonicity

Simple cases

- $f(x) > \pi \searrow \qquad f(x) < \pi \nearrow$
- $\mathsf{G}_{[0,\tau]}\ \varphi \searrow \qquad \mathsf{F}_{[0,\tau]}\ \varphi \nearrow$
- etc

Deciding monotonicity

Simple cases

- $f(x) > \pi \searrow$ $\quad f(x) < \pi \nearrow$
- $\mathsf{G}_{[0,\tau]}\ \varphi \searrow$ $\quad \mathsf{F}_{[0,\tau]}\ \varphi \nearrow$
- etc

General case

- Deciding monotonicity can be encoded in an SMT query
- However, the problem is undecidable, due to undecidabilty of STL
- In practice, monotonicity can be decided easily (in our experience so far)

Parameter synthesis problem

Problem

Given the system:

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

Find an input signal $u \in \mathcal{U},\ p \in \mathcal{P}$ *such that* $S(u(t), p), 0 \models \varphi$

Parameter synthesis problem

Problem

Given the system:

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

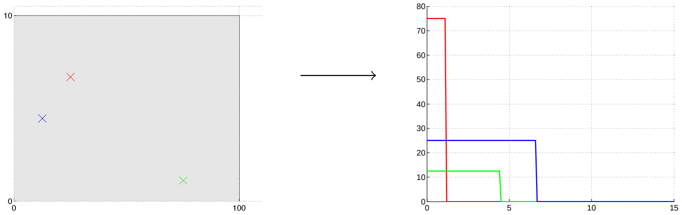
Find an input signal $u \in \mathcal{U},\ p \in \mathcal{P}$ *such that* $S(u(t), p), 0 \models \varphi$

In practice

- We parameterize $\mathcal{U}$ and reduce the problem to a parameter synthesis problem within some set $\mathcal{P}_u \times \mathcal{P}$
- The search of a solution is guided by the quantitative measure of satisfaction of φ

Parameterizing the input space

Input parameter set $\mathcal{P}_u$ $\qquad$ Input signals $u(t) \in \mathcal{U}$



Note

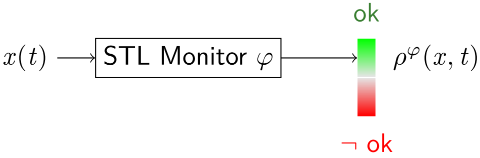
The set of input signals generated by $\mathcal{P}_u$ is in general a subset of $\mathcal{U}$

I.e., we do not guarantee completeness.

Parameter synthesis with quantitative satisfaction

Given a formula φ, a signal x and a time t, recall that we have:

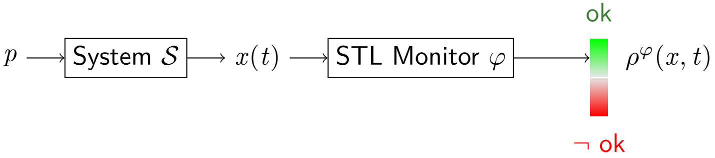
$$\rho^{\varphi}(x,t) > 0 \Rightarrow x, t \models \varphi$$
$$\rho^{\varphi}(x,t) < 0 \Rightarrow x, t \not\models \varphi$$



Parameter synthesis with quantitative satisfaction

Given a formula φ, a signal x and a time t, recall that we have:

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



As x is obtained by simulation using input parameters p, the falsification problem can be reduced to solving

$$\rho^* = \min_{p \in \mathcal{P}} \rho^{\varphi}(x, 0)$$

If $\rho^* < 0$, we found a counterexample.

Open question: optimizing satisfaction function

Solving

$$\rho^* = \min_{p_u \in \mathcal{P}_u} F(p_u) = \rho^\varphi(x, 0)$$

is difficult in general, as nothing can be assumed on F.

In practice, use of global nonlinear optimization algorithms

Success will depend on how smooth is F_u, its local optima, etc

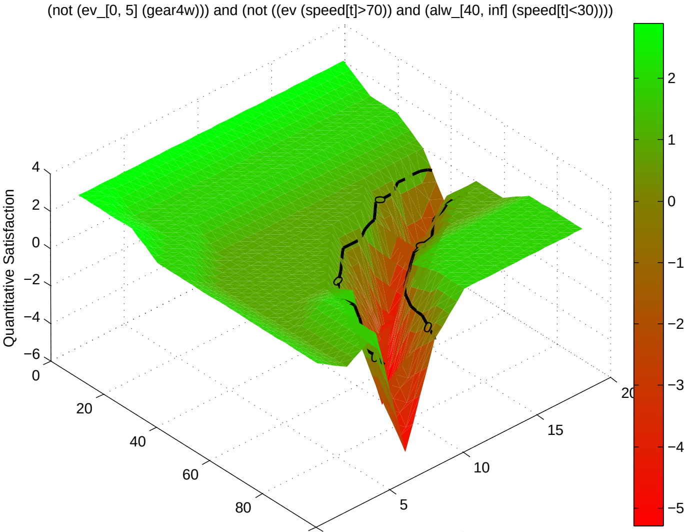
Critical is the ability to compute ρ efficiently.

Open question (cont'd): smoothing quantitative satisfaction functions

Depending on how ρ is defined, the function to optimize can have different profiles

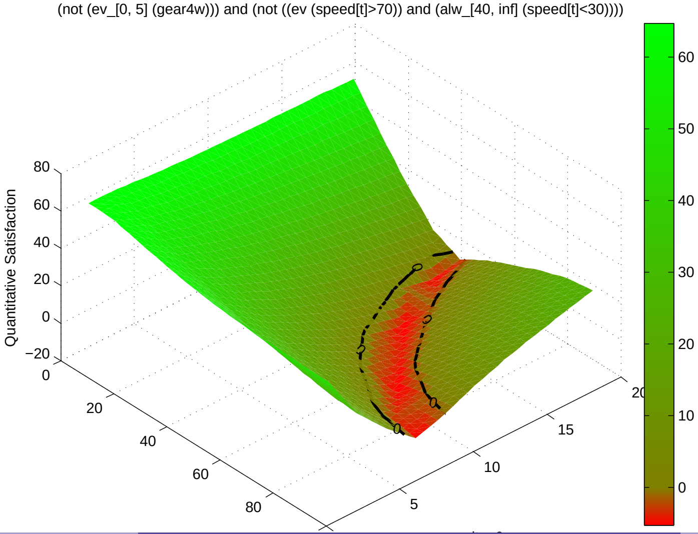
Open question (cont'd): smoothing quantitative satisfaction functions

Depending on how ρ is defined, the function to optimize can have different profiles



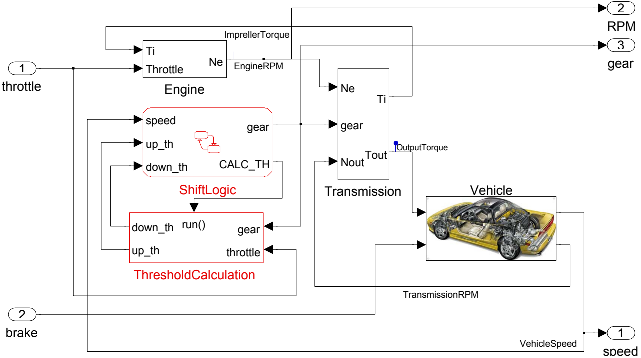
Open question (cont'd): smoothing quantitative satisfaction functions

Depending on how ρ is defined, the function to optimize can have different profiles



Specification mining

Consider the following automatic transmission system:



- What is the maximum speed that the vehicule can reach ?
- What is the minimum dwell time in a given gear ?
- etc

Specification synthesis

The approach takes two major ingredients

- PSTL to formulate template specifications
- A counter-example guided inductive synthesis loop alternating parameter synthesis and falsification

Template specification examples

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

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

Template specification examples

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

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

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

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

Template specification examples

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

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

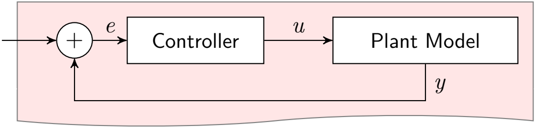
- *the vehicle cannot reach 100 mph in* τ *seconds with* RPM *always below* π

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

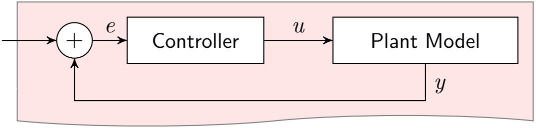
- *whenever it shift to gear* 2*, it dwells in gear* 2 *for at least* τ *seconds*

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

Specification mining algorithm



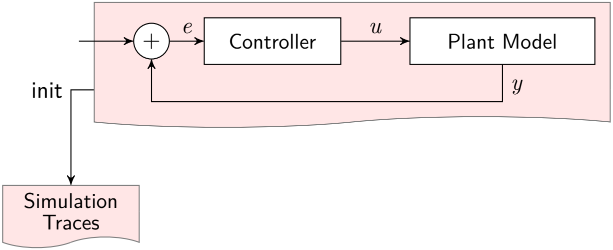
Specification mining algorithm



$$F_{[0,\tau_1]}(\mathtt{x}_1 < \pi_1 \wedge G_{[0,\tau_2]}(\mathtt{x}_2 > \pi_2))$$

Template Specification

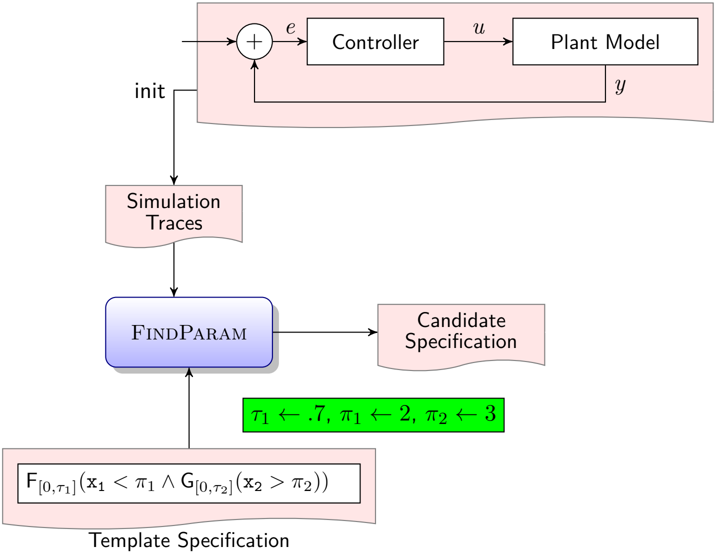
Specification mining algorithm



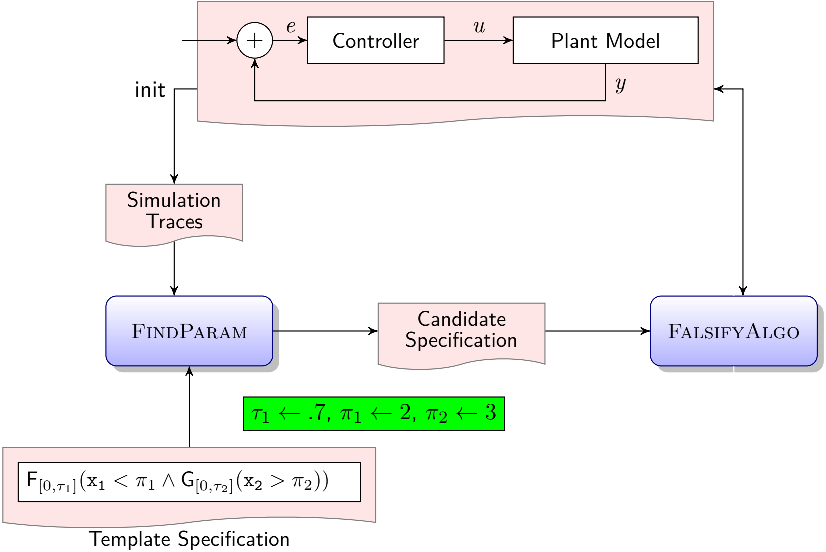
$$F_{[0,\tau_1]}(\mathtt{x_1} < \pi_1 \wedge G_{[0,\tau_2]}(\mathtt{x_2} > \pi_2))$$

Template Specification

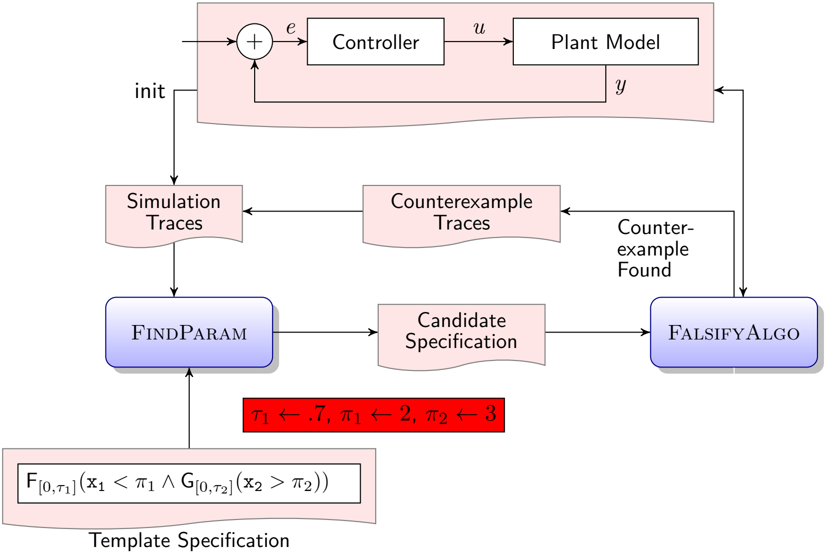
Specification mining algorithm



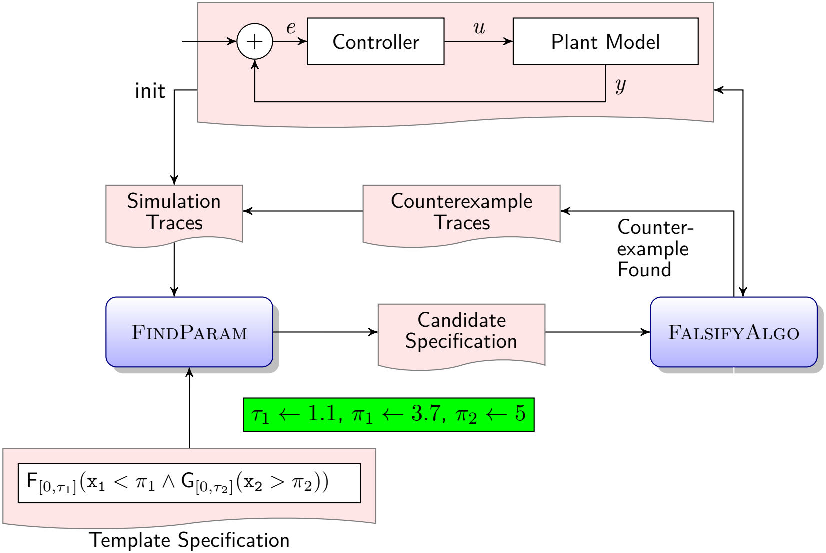
Specification mining algorithm



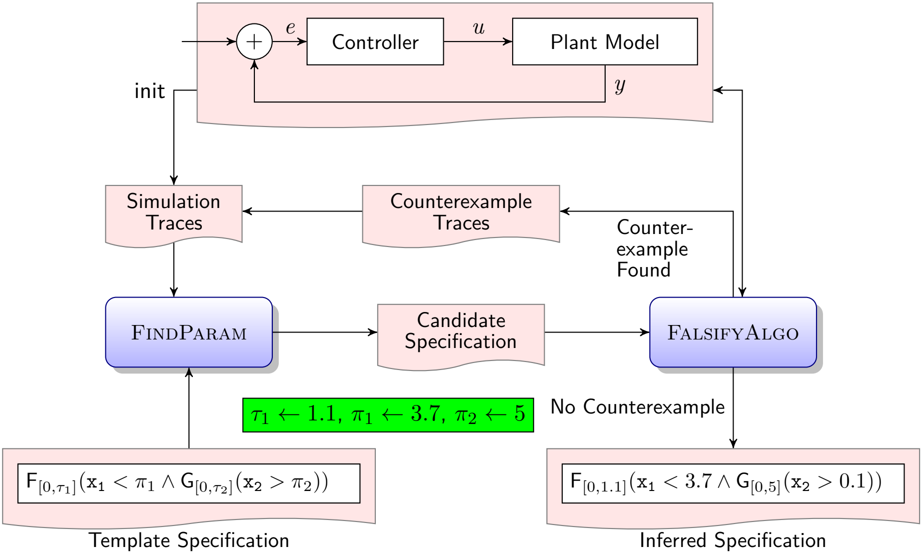
Specification mining algorithm



Specification mining algorithm



Specification mining algorithm



Results

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

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

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

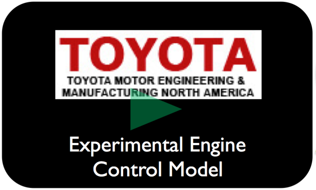
$$\varphi_{\texttt{rpm100}}(\tau,\pi) := \neg(\mathsf{F}_{[0,\tau]} (\texttt{speed} > 100) \wedge \mathsf{G}(\texttt{RPM} < \pi)).$$

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

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

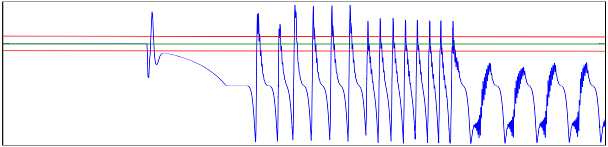
Template	Parameter values	Fals.	Synth.	#Sim.	Sat./x
$\varphi_{\texttt{sp\_rpm}}(\pi_1,\pi_2)$	(155 mph, 4858 rpm)	197.2 s	23.1 s	496	0.043 s
$\varphi_{\texttt{rpm100}}(\pi,\tau)$	(3278.3 rpm, 49.91 s)	267.7 s	10.51 s	709	0.026 s
$\varphi_{\texttt{rpm100}}(\tau,\pi)$	(4997 rpm, 12.20 s)	147.8 s	5.188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

Results on Industrial-scale Model

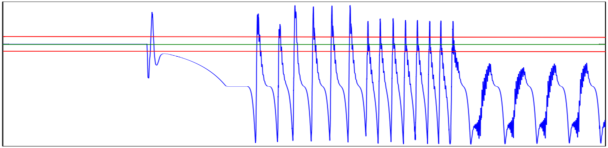


4000+ Simulink blocks
Look-up tables
nonlinear dynamics

- Attempt to mine maximum observed settling time:
  - stops after 4 iterations
  - gives answer t_{settle} = simulation time horizon...



Results on Industrial-scale Model



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

Conclusion and future work

Summary

- Efficient parameter synthesis PSTL for monotonic formulas
- Model parameter synthesis based on quantitative semantics
- Parametric specification mining combining both
- Tools support: Breach toolbox

To dos

- Efficient synthesis for non-monotonic formulas ?
- Better optimization algorithm for quantitative semantics
- Beyond parameter synthesis (signals, formulas, systems)