# On Signal Temporal Logic

Alexandre Donzé

University of California, Berkeley

November 7, 2013

# Outline

# Outline

# 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 $\varphi$ is evaluated on a sequence, e.g., $w = aaabbaaa\ldots$

At each step of $w$, we can define a truth value of $\varphi$, 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

| | *Trace* | $w =$ | $a$ | $a$ | $a$ | $b$ | $b$ | $a$ | $a$ | $a$ | $\ldots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\bigcirc b$ | (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

| | *Trace* | $w =$ | $a$ | $a$ | $a$ | $b$ | $b$ | $a$ | $a$ | $a$ | $\ldots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\bigcirc b$ | (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

| | *Trace* | $w =$ | $a$ | $a$ | $a$ | $b$ | $b$ | $a$ | $a$ | $a$ | $\dots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\bigcirc b$ | (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

| | *Trace* | $w =$ | $a$ | $a$ | $a$ | $b$ | $b$ | $a$ | $a$ | $a$ | $\ldots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\bigcirc b$ | (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

| | *Trace* | $w =$ | $a$ | $a$ | $a$ | $b$ | $b$ | $a$ | $a$ | $a$ | $\dots$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\bigcirc b$ | (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? | $\dots$ |

## Remarks

$\chi$ is acausal: it depends on future events

Finite sequences semantics allows to define a unique value $\forall(w,i)$

Notation: $w \models \varphi \Leftrightarrow \chi^{\varphi}(w,0) = 1$

# Model-Checking

Suppose $w$ are execution traces of some system $\mathcal{M}$

$$\boxed{\text{System } \mathcal{M}} \longrightarrow aaaabbbaa\ldots \longrightarrow \boxed{\text{Property } \varphi} \longrightarrow 111000\ldots$$

Model-checking: proving that $\mathcal{M} \models \varphi$

where $\mathcal{M} \models \varphi \;\Leftrightarrow\;$ For all $w$ in traces($\mathcal{M}$), $\chi^{\varphi}(w, 0) = 1$

# Model-Checking

Suppose $w$ are execution traces of some system $\mathcal{M}$

$$\boxed{\text{System } \mathcal{M}} \longrightarrow aaaabbbaa\ldots \longrightarrow \boxed{\text{Property } \varphi} \longrightarrow 111000\ldots$$

Model-checking: proving that $\mathcal{M} \models \varphi$

where $\mathcal{M} \models \varphi \;\Leftrightarrow\;$ For all $w$ in traces($\mathcal{M}$), $\chi^{\varphi}(w, 0) = 1$

Monitoring: computing $\chi^{\varphi}(w, 0)$ for finite sets of $w$

# Model-Checking

Suppose $w$ are execution traces of some system $\mathcal{M}$

$$\boxed{\text{System } \mathcal{M}} \longrightarrow aaaabbbaa\ldots \longrightarrow \boxed{\text{Property } \varphi} \longrightarrow 111000\ldots$$

Model-checking: proving that $\mathcal{M} \models \varphi$

$$\text{where } \mathcal{M} \models \varphi \Leftrightarrow \text{ For all } w \text{ in traces}(\mathcal{M}),\ \chi^{\varphi}(w,0) = 1$$

Monitoring: computing $\chi^{\varphi}(w,0)$ for finite sets of $w$

Remark: Statistical model checking
Doing statistics on $\chi^{\varphi}(w,0)$ for populations of $w$

# Temporal Logics in the Wild

Model checking temporal logics successful in formal verification and synthesis for hardware digital circuits

# Temporal Logics in the Wild

Model checking temporal logics successful in formal verification and synthesis for hardware digital circuits

Most on-going research in model checking aims at software

# Temporal Logics in the Wild

Model checking temporal logics successful in formal verification and synthesis for hardware digital circuits

Most on-going research in model checking aims at software

But growing interest/needs in even scarier fields such as analog/mixed-signal circuits, systems biology, cyber-physical systems

# Temporal Logics in the Wild

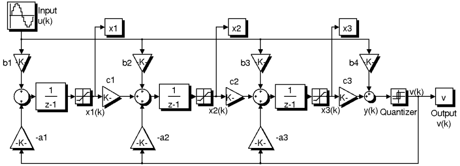
Model checking temporal logics successful in formal verification and synthesis for hardware digital circuits

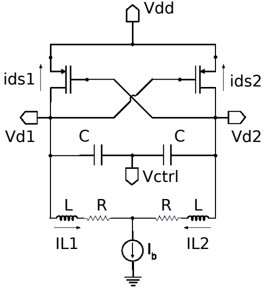
Most on-going research in model checking aims at software

But growing interest/needs in even scarier fields such as analog/mixed-signal circuits, systems biology, cyber-physical systems

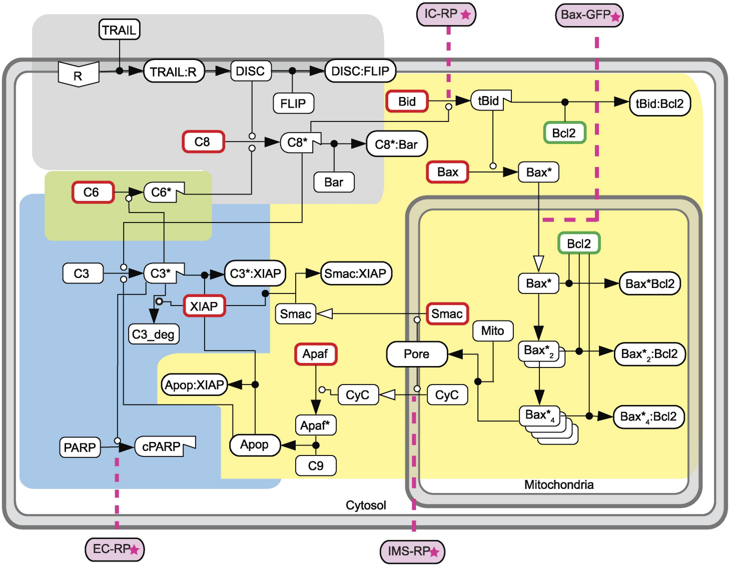
$\Rightarrow$ Tendency to move from discrete-time discrete systems to hybrid (discrete-continuous) systems

# Temporal Logics in the Wild

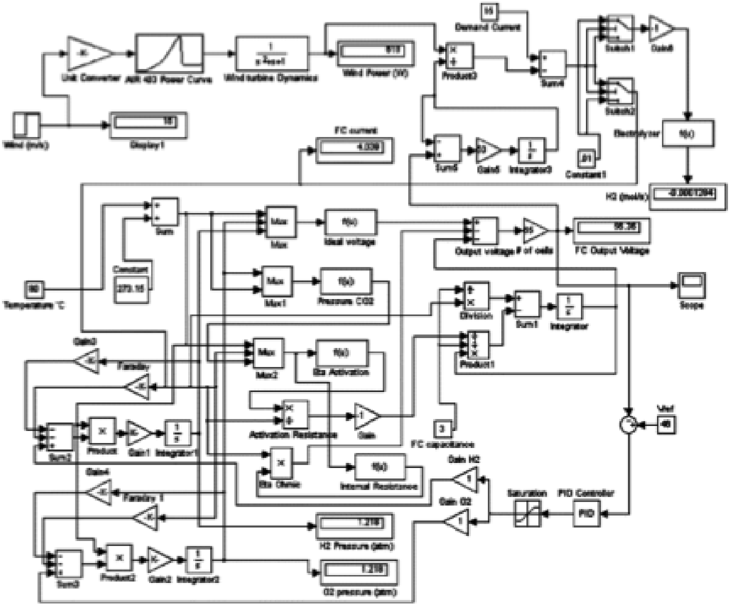


# Temporal Logics in the Wild



# Temporal Logics in the Wild



# Temporal Logics in the Wild

*On Temporal Logic and Signal Processing*, A. Donzé, O. Maler, E. Bartocci, D. Nickovic, R. Grosu, S. Smolka,, ATVA 2012

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

STL G( $x[t] > 0$ => $F_{[0,.5s]} y[t] > 0$ )
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

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

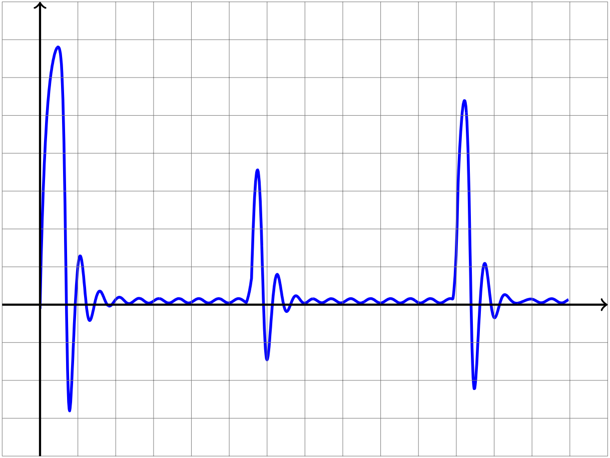
- $\bot = \neg\top$
- Eventually is $\mathsf{F}_{[a,b]} \ \varphi = \top \ \mathcal{U}_{[a,b]} \ \varphi$
- Always is $\mathsf{G}_{[a,b]}\varphi = \neg( \ \mathsf{F}_{[a,b]} \ \neg\varphi)$

## 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:

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

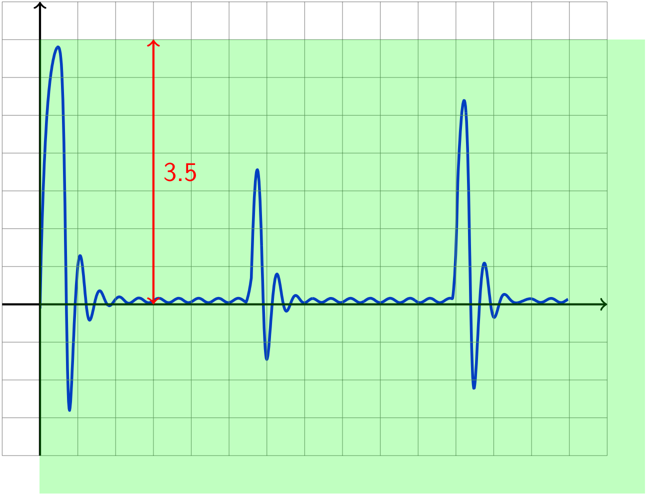
# STL Examples



# STL Examples

*The signal is never above 3.5*

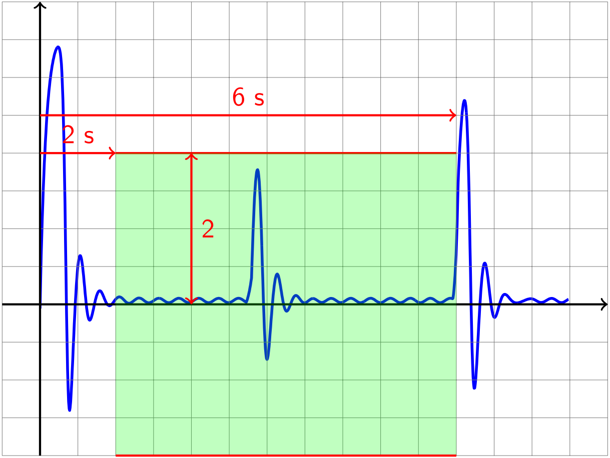
$$\varphi := \mathsf{G}\ (x[t] < 3.5)$$



# STL Examples

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

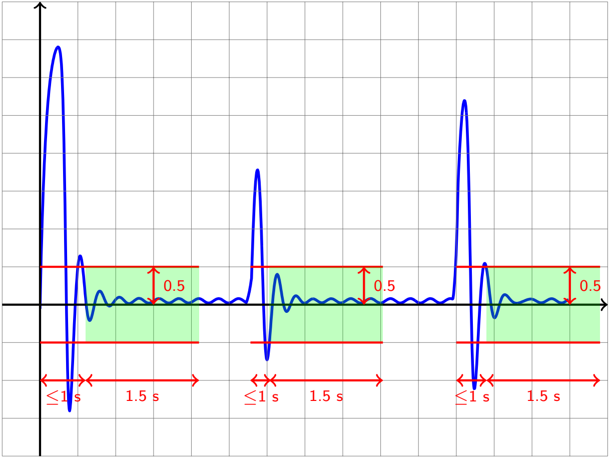
$$\varphi := \ \mathsf{G}_{[2,6]} \ \ (|x[t]| < 2)$$



# STL Examples

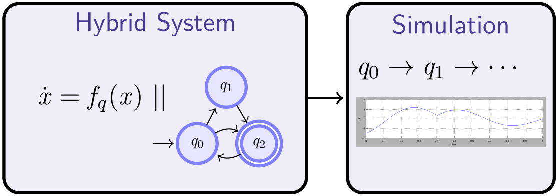
*Always* $|x| > 0.5 \Rightarrow$ *after 1 s,* $|x|$ *settles under 0.5 for 1.5 s*

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



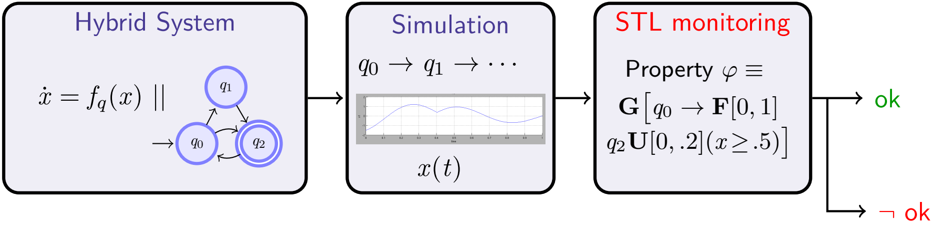
# Model-Checking STL

- Models are generally hybrid systems producing hybrid traces



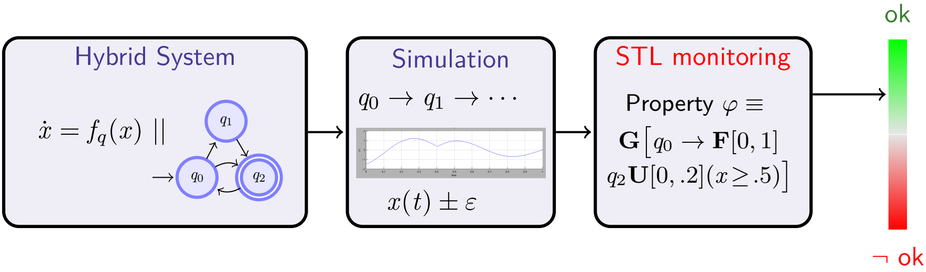
# Model-Checking STL

- Models are generally hybrid systems producing hybrid traces
- Model-Checking untractable except in restrictive cases, resort to monitoring



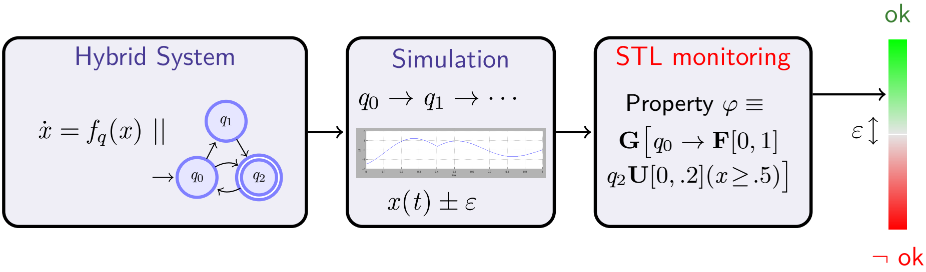
# Model-Checking STL

- Models are generally hybrid systems producing hybrid traces
- Model-Checking untractable except in restrictive cases, resort to monitoring
- Quantitative satisfaction of STL can accomodate noise/approximations and more



# Model-Checking STL

- Models are generally hybrid systems producing hybrid traces
- Model-Checking untractable except in restrictive cases, resort to monitoring
- Quantitative satisfaction of STL can accomodate noise/approximations and more




Tool Support: Breach Toolbox

# Outline

# STL Semantics

The validity of a formula $\varphi$ with respect to 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 Satisfaction Function

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

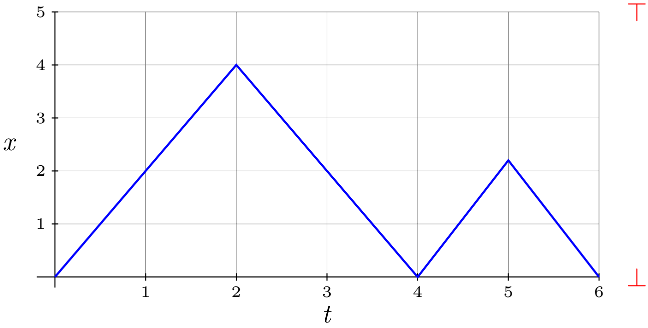
$$x, t \models \varphi \Leftrightarrow \chi^\varphi(x,t) = \top$$

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

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

# Example

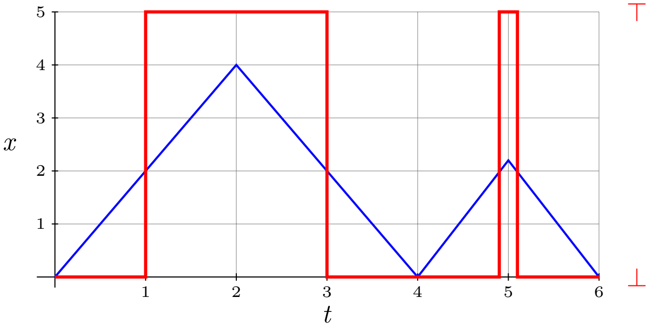
Consider a simple piecewise affine signal:




Satisfaction signal of :

# Example

Consider a simple piecewise affine signal:

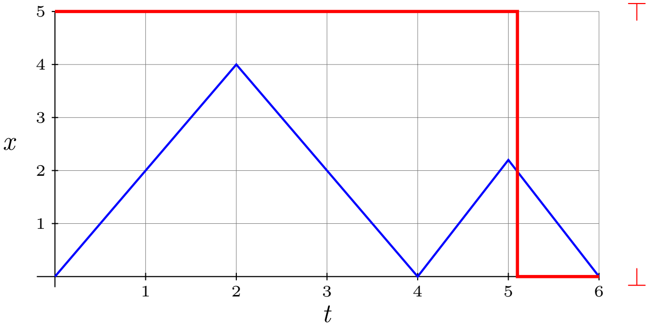



Satisfaction signal of :

- $\varphi = x \geq 2$

# Example

Consider a simple piecewise affine signal:

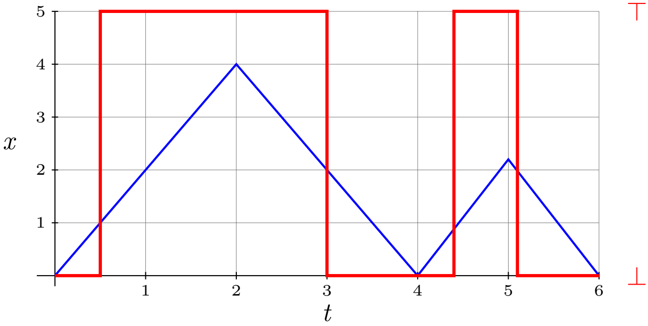



Satisfaction signal of :

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

# 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:

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

# Property of Robust Satisfaction Signal

- Sign indicates satisfaction status

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

# Property 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

$$\begin{array}{lcl} x, t \models \varphi \text{ and } \|x - x'\|_{\infty} \leq \rho^{\varphi}(x,t) & \Rightarrow & x', t \models \varphi \\ x, t \not\models \varphi \text{ and } \|x - x'\|_{\infty} \leq -\rho^{\varphi}(x,t) & \Rightarrow & x', t \not\models \varphi \end{array}$$

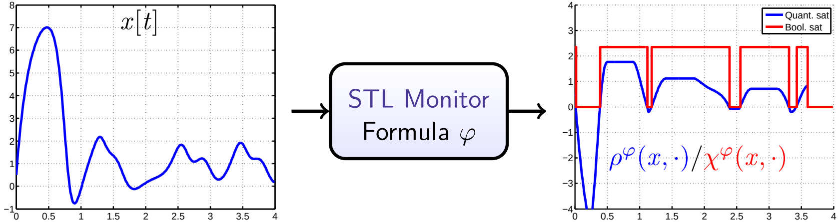
# Outline

# Robust Monitoring

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

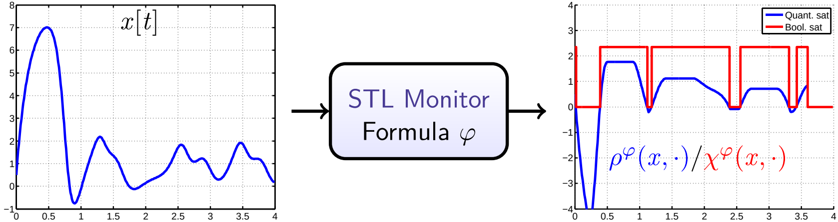
# Robust Monitoring

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)

- 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

- The function $\rho^{\varphi}(x, t)$ 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 $\varphi$

# Boolean operators

## Negation

- Input signal: $(t_i, x(t_i))_{i \leq n_x}$
- Output signal: $(t_i, -x(t_i))_{i \leq n_x}$

# Boolean operators

## Negation

- Input signal: $(t_i, x(t_i))_{i \leq n_x}$
- Output signal: $(t_i, -x(t_i))_{i \leq n_x}$

## Conjunction

- Input signals: $(t_i, x(t_i))_{i \leq n_x}$, $(t'_i, x'(t'_i))_{i \leq n_{x'}}$
- Output signal: $(r_i, z(r_i))_{i \leq n_z}$
  Time sequence $r$ contains $t$, $t'$, and punctual intersections $x \cap x'$
  Value $z(r_i) = \min\{x(r_i), x'(r_i)\}$

# Until

### Rewrite Property

- Boolean Semantics
  $\varphi \mathbf{U}_{[a,b]} \psi \;\sim\; \mathbf{G}_{[0,a]} \varphi \wedge \mathbf{F}_{[a,b]} \psi \wedge \mathbf{F}_{\{a\}} (\varphi \mathbf{U} \psi)$

Combines *untimed until* and *timed eventually*

# Until

## Rewrite Property

- Boolean Semantics
  $\varphi \mathbf{U}_{[a,b]} \psi \;\sim\; \mathbf{G}_{[0,a]} \varphi \wedge \mathbf{F}_{[a,b]} \psi \wedge \mathbf{F}_{\{a\}} (\varphi \mathbf{U} \psi)$

- Quantitative Semantics
  $\rho^{\varphi \mathbf{U}[a,b] \psi}(x, t) = \rho^{\mathbf{G}_{[0,a]} \varphi \wedge \mathbf{F}_{[a,b]} \psi \wedge \mathbf{F}_{\{a\}} (\varphi \mathbf{U} \psi)}(x, t)$

Combines *untimed until* and *timed eventually*

# Untimed Until

Computed by *backward induction*:

For all $s < t$, we note $x_{\upharpoonright[s,t)}$ the restriction of $x$ to $[s, t)$.

- Boolean Semantics $\quad x, s \vDash \varphi \mathbf{U} \psi \quad$ iff
  $x_{\upharpoonright[s,t)}, s \vDash \varphi \mathbf{U} \psi$ or $\left(x_{\upharpoonright[s,t)}, s \vDash \mathbf{G}\varphi\right.$ and $\left. x, t \vDash \varphi \mathbf{U} \psi\right)$

# Untimed Until

Computed by *backward induction*:

For all $s < t$, we note $x_{\upharpoonright[s,t)}$ the restriction of $x$ to $[s, t)$.

- Boolean Semantics $\quad x, s \models \varphi \mathbf{U} \psi \quad$ iff
  $x_{\upharpoonright[s,t)}, s \models \varphi \mathbf{U} \psi$ or $\left(x_{\upharpoonright[s,t)}, s \models \mathbf{G}\varphi \text{ and } x, t \models \varphi \mathbf{U} \psi\right)$

- Quantitative Semantics $\quad \rho^{\varphi \mathbf{U} \psi}(x, s) =$
  $\max\left\{\rho^{\varphi \mathbf{U} \psi}(x_{\upharpoonright[s,t)}, s), \min\{\rho(\mathbf{G}\varphi, x_{\upharpoonright[s,t)}, s), \rho(\varphi \mathbf{U} \psi, x, t)\}\right\}$

# Timed Eventually

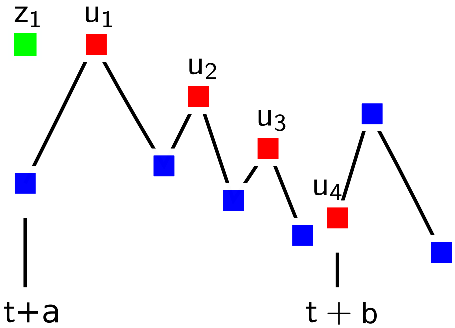
Definition: $\rho^{\mathbf{F}_{[a,b]}\varphi}(x,t) = \sup_{t' \in [t+a,t+b]} \rho^{\varphi}(x,t) = \sup_{[t+a,t+b]} x$

Computation:

- the maximum is reached at $t+a, t+b$, or at sample point in $\{t_i \mid t_i \in (t+a, t+b]\}$
- $\max\{x(t_i) \mid t_i \in (t+a, t+b]\}$ computed by Lemire's algorithm:

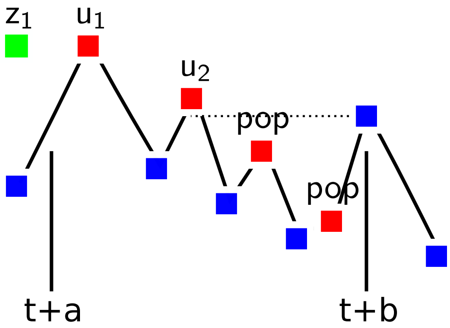
  we maintain an ordered set $M$ such that $\max\{x(t_i) | i \in M\} = \max\{x(t_i) \mid t_i \in (t+a, t+b]\}$

# Timed Eventually: two steps in Lemire's algorithm



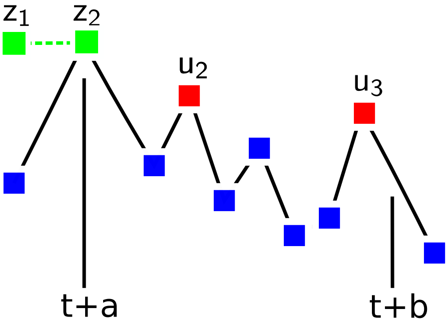

Maximum candidates $\{x(t_i)|i \in M\} = \{u_1, u_2, u_3, u_4\}$

# Timed Eventually: two steps in Lemire's algorithm



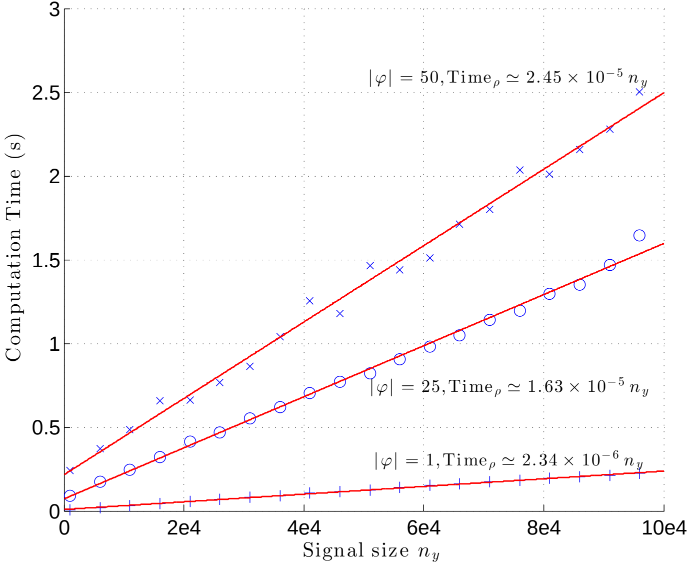

Maximum candidates $\{x(t_i) | i \in M\} = \{u_1, u_2, u_3\}$

# Timed Eventually: two steps in Lemire's algorithm




Maximum candidates $\{x(t_i) | i \in M\} = \{u_2, u_3\}$

# Performance Results



# Parametric STL

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

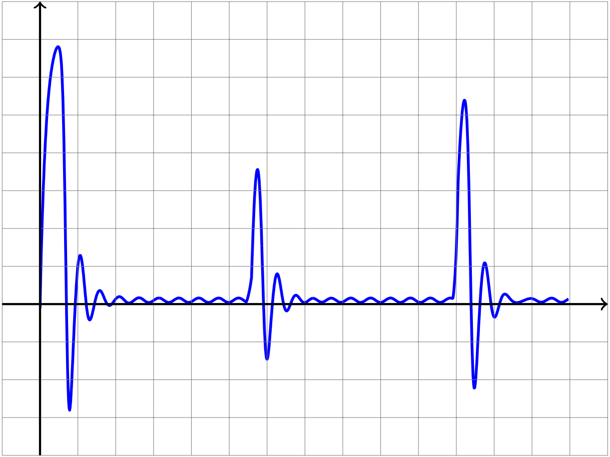
## Definition (PSTL syntax)

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

where

- $\pi$ is a scale parameter
- $\tau_1$, $\tau_2$ are time parameters

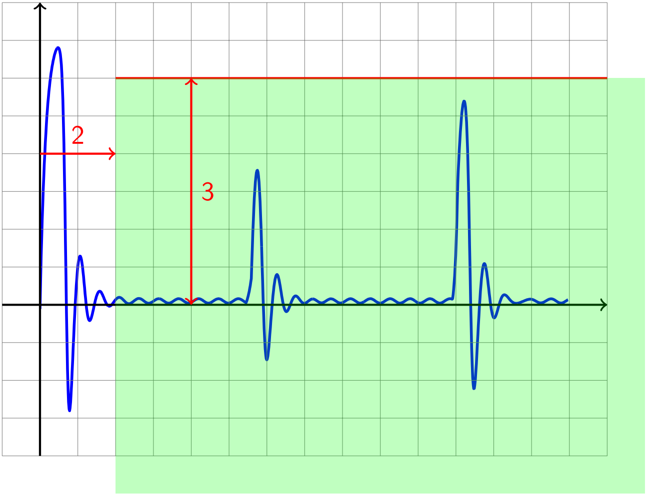
# Parametric STL - Illustration



# Parametric STL - Illustration

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

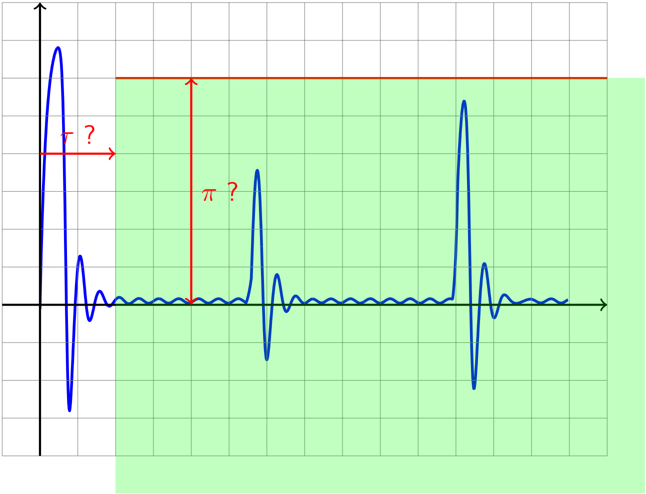
$$\varphi := \ \mathsf{F}_{[2,\infty]} \ \ (x[t] < 3)$$



# Parametric STL - Illustration

*"After* $\tau$ *s, the signal is never above* $\pi$*"*

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



# 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*

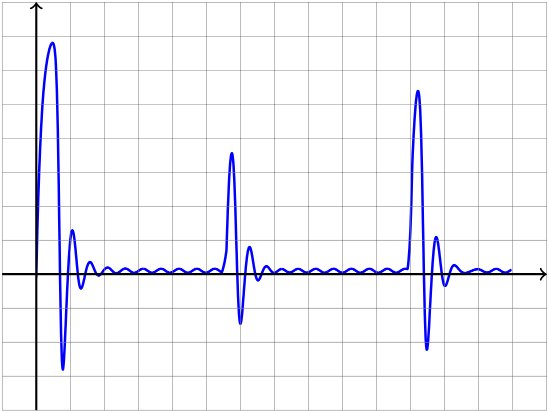
$$x, t \models \varphi(v(p_1), \ldots, v(p_n)),$$

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

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

# Example

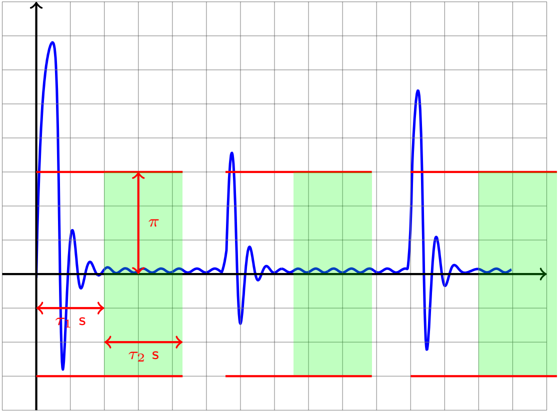
$$\varphi := \mathsf{G}\left(x[t] > \pi \rightarrow \ \mathsf{F}_{[0,\tau_1]} \ \ ( \ \mathsf{G}_{[0,\tau_2]} \ x[t] < \pi)\right)$$



# Example

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

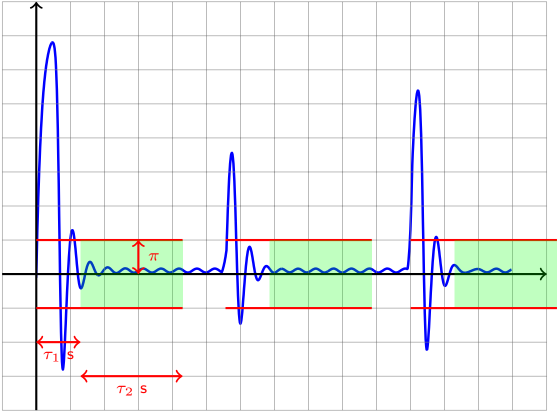
- Valuation 1: $\pi \leftarrow 1.5$, $\tau_1 \leftarrow 1 \ s$, $\tau_2 \leftarrow 1.15 \ s$



# 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

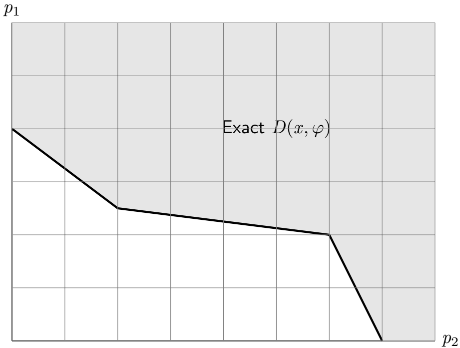
A PSTL formula $\varphi(p_1, \cdots, p_n)$ is monotonically increasing wrt $p_i$ if

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

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

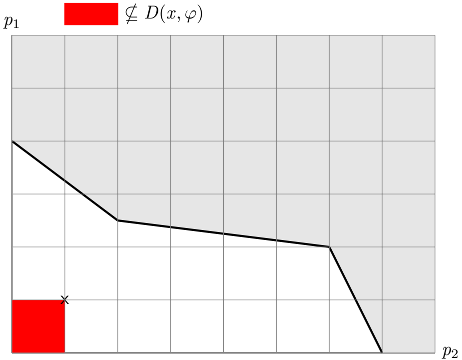
# Monotonic Validity Domains

- The validity domain $D$ of $\varphi$ 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 $\partial D$
- In case of monoticity, $\partial D$ has the structure of a Pareto front which can be estimated with generalized binary search heuristics



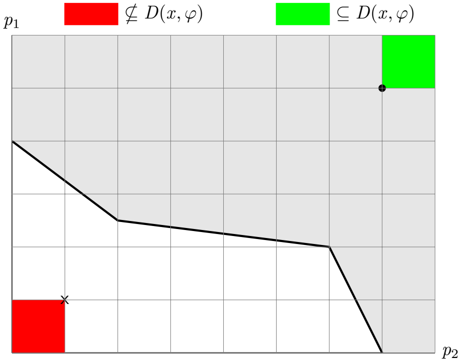
# Monotonic Validity Domains

- The validity domain $D$ of $\varphi$ 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 $\partial D$
- In case of monoticity, $\partial D$ has the structure of a Pareto front which can be estimated with generalized binary search heuristics



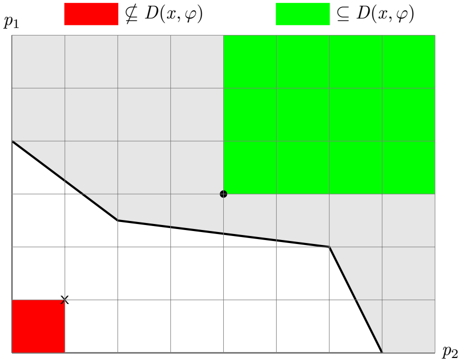
# Monotonic Validity Domains

- The validity domain $D$ of $\varphi$ 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 $\partial D$
- In case of monoticity, $\partial D$ has the structure of a Pareto front which can be estimated with generalized binary search heuristics



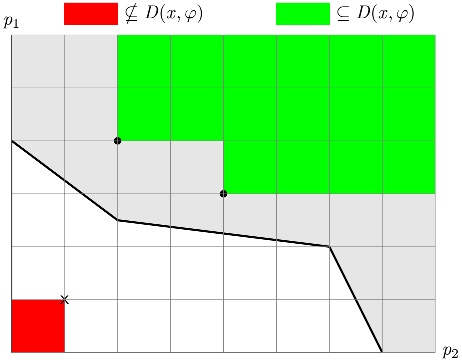
# Monotonic Validity Domains

- The validity domain $D$ of $\varphi$ 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 $\partial D$
- In case of monoticity, $\partial D$ has the structure of a Pareto front which can be estimated with generalized binary search heuristics



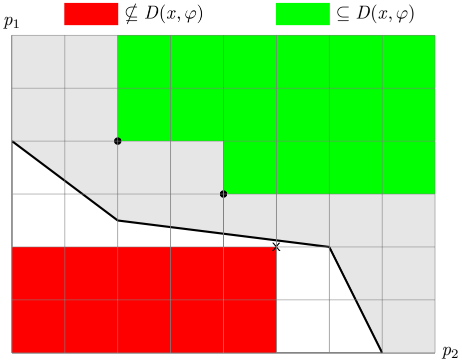
# Monotonic Validity Domains

- The validity domain $D$ of $\varphi$ 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 $\partial D$
- In case of monoticity, $\partial D$ has the structure of a Pareto front which can be estimated with generalized binary search heuristics



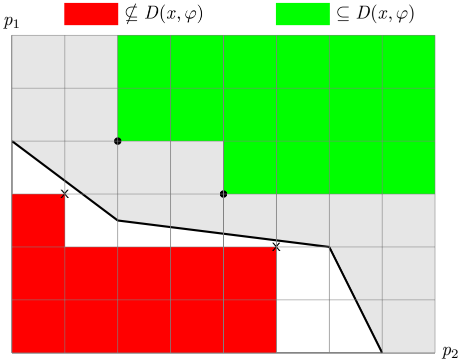
# Monotonic Validity Domains

- The validity domain $D$ of $\varphi$ 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 $\partial D$
- In case of monoticity, $\partial D$ has the structure of a Pareto front which can be estimated with generalized binary search heuristics



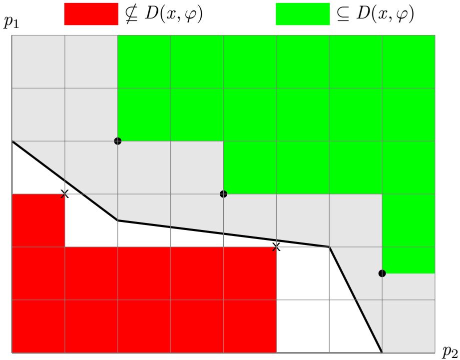
# Monotonic Validity Domains

- The validity domain $D$ of $\varphi$ 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 $\partial D$
- In case of monoticity, $\partial D$ has the structure of a Pareto front which can be estimated with generalized binary search heuristics



# Monotonic Validity Domains

- The validity domain $D$ of $\varphi$ 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 $\partial D$
- In case of monoticity, $\partial D$ has the structure of a Pareto front which can be estimated with generalized binary search heuristics



# Deciding Monotonicity

## Simple cases

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

# 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

## 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)

# Solving the 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* $S(u(t)), 0 \not\models \varphi$

# Solving the Falsification problem

## Problem

*Given the system:*

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

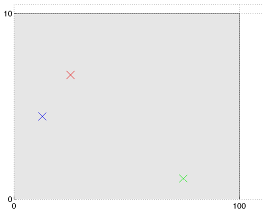
*Find an input signal* $u \in \mathcal{U}$ *such that* $S(u(t)), 0 \not\models \varphi$

## In practice

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

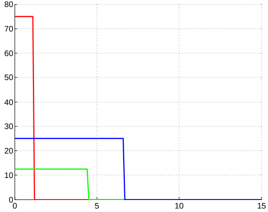
# Parameterizing the Input Space

Input parameter set $\mathcal{P}_u$




$\longrightarrow$

Input signals $u(t) \in \mathcal{U}$




### Note

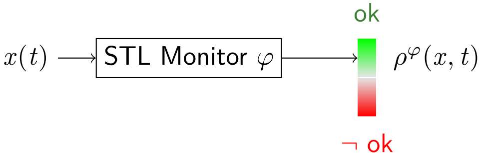
The set of input signals generated by $\mathcal{P}_u$ is in general a subset of $\mathcal{U}$

I.e., we do not guarantee completeness.

# Falsification with Quantitative Satisfaction

Given a formula $\varphi$, a signal $x$ and a time $t$, recall that we have:

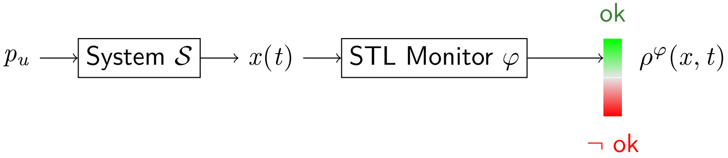
$$\rho^{\varphi}(x,t) > 0 \Rightarrow x, t \models \varphi$$
$$\rho^{\varphi}(x,t) < 0 \Rightarrow x, t \not\models \varphi$$



# Falsification with Quantitative Satisfaction

Given a formula $\varphi$, 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 $p_u$, the falsification problem can be reduced to solving

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

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

# 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

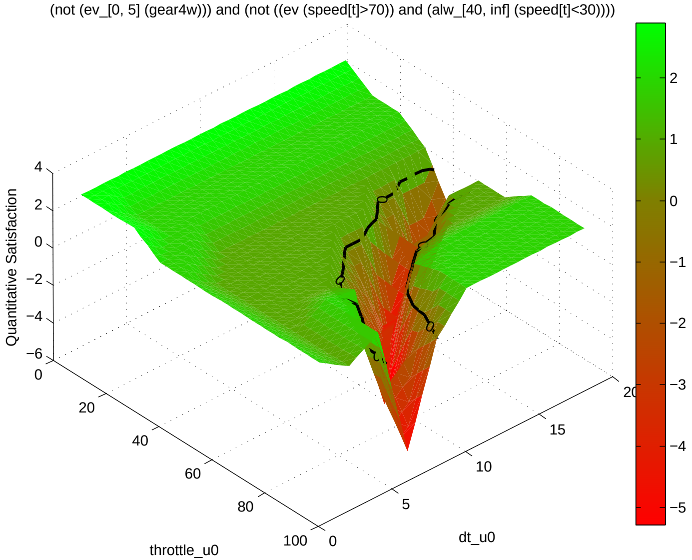
Critical is the ability to compute $\rho$ efficiently.

# Smoothing Quantitative Satisfaction Functions

Depending on how $\rho$ is defined, the function to optimize can have different profiles

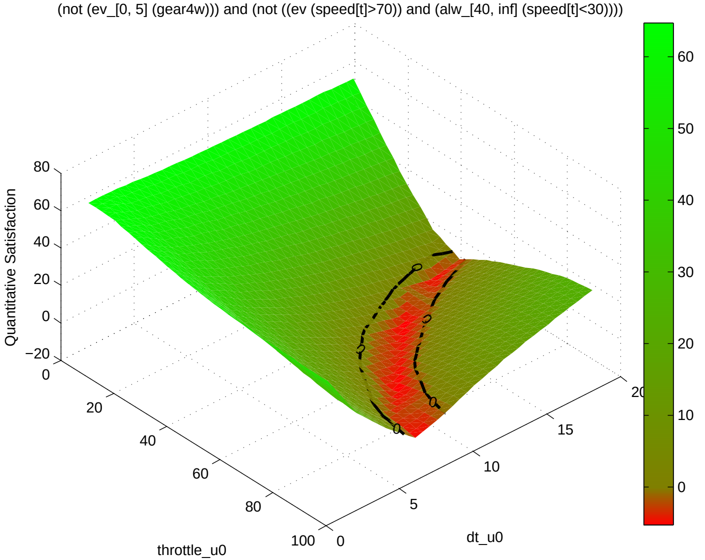
# Smoothing Quantitative Satisfaction Functions

Depending on how $\rho$ is defined, the function to optimize can have different profiles



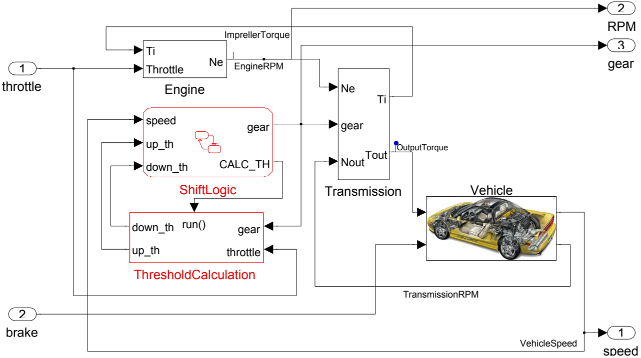
# Smoothing Quantitative Satisfaction Functions

Depending on how $\rho$ is defined, the function to optimize can have different profiles



# Specification Mining Problem

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

Our 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* $\pi_1$ *and* RPM *below* $\pi_2$

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

# Template Specification Examples

- *the* speed *is always below* $\pi_1$ *and* RPM *below* $\pi_2$

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

- *the vehicle cannot reach 100 mph in* $\tau$ *seconds with* RPM *always below* $\pi$

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

# Template Specification Examples

- *the* speed *is always below* $\pi_1$ *and* RPM *below* $\pi_2$

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

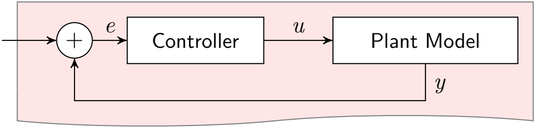
- *the vehicle cannot reach 100 mph in* $\tau$ *seconds with* RPM *always below* $\pi$

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

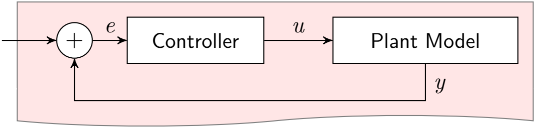
- *whenever it shift to gear* $2$, *it dwells in gear* $2$ *for at least* $\tau$ *seconds*

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

# Specification Synthesis Algorithm



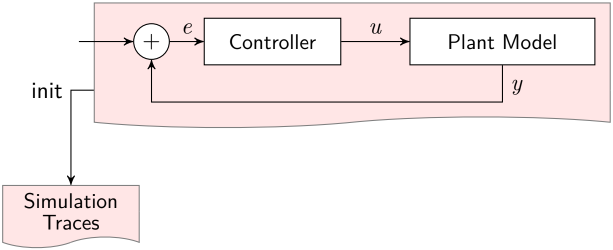
# Specification Synthesis Algorithm




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

Template Specification

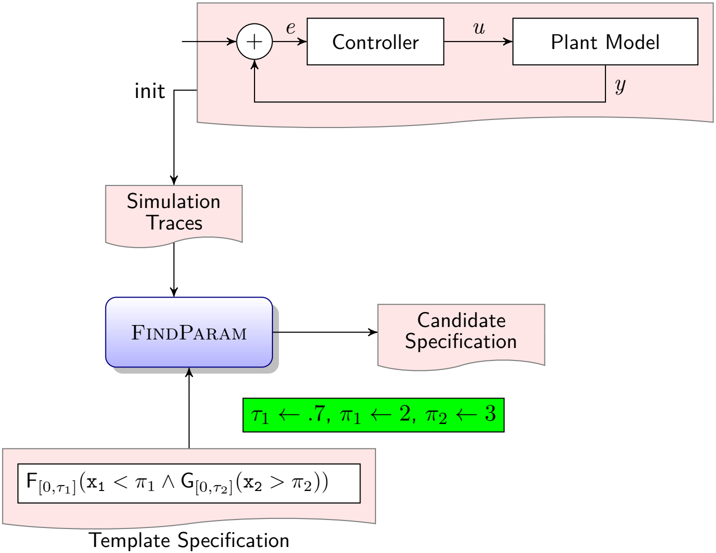
# Specification Synthesis Algorithm



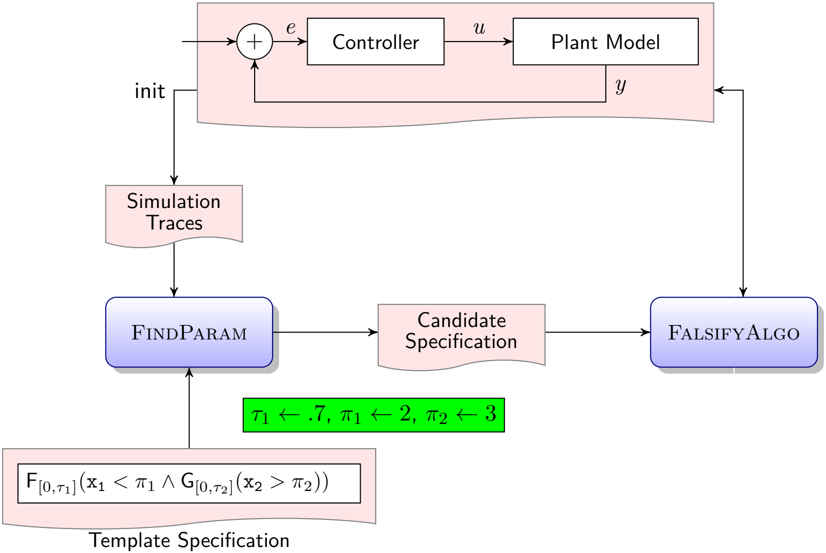

$$\mathsf{F}_{[0,\tau_1]}(\mathtt{x}_1 < \pi_1 \wedge \mathsf{G}_{[0,\tau_2]}(\mathtt{x}_2 > \pi_2))$$

Template Specification

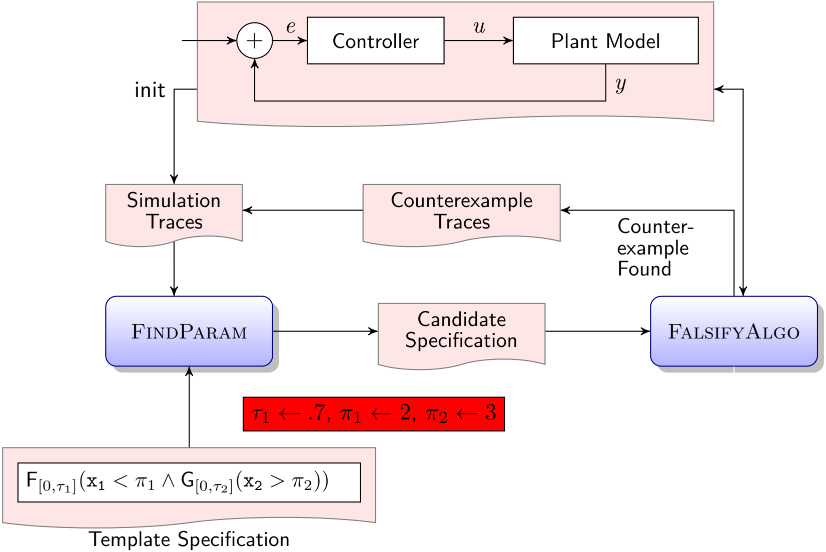
# Specification Synthesis Algorithm



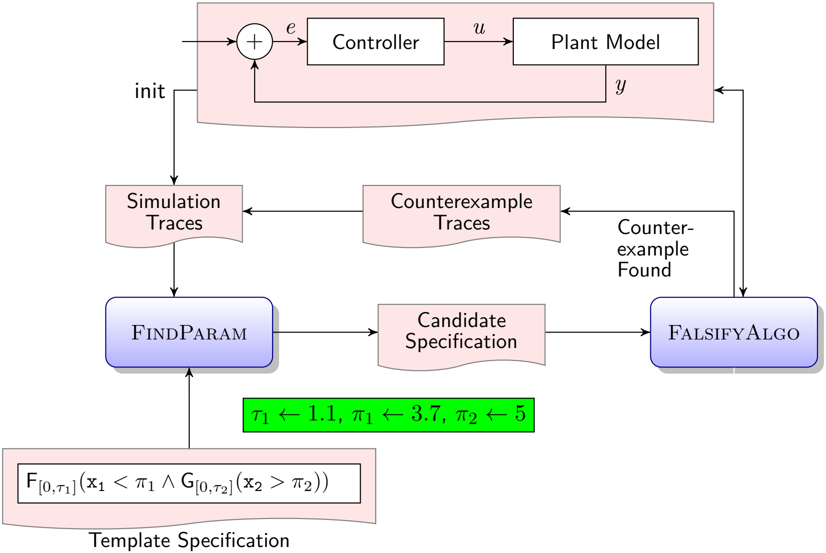
# Specification Synthesis Algorithm



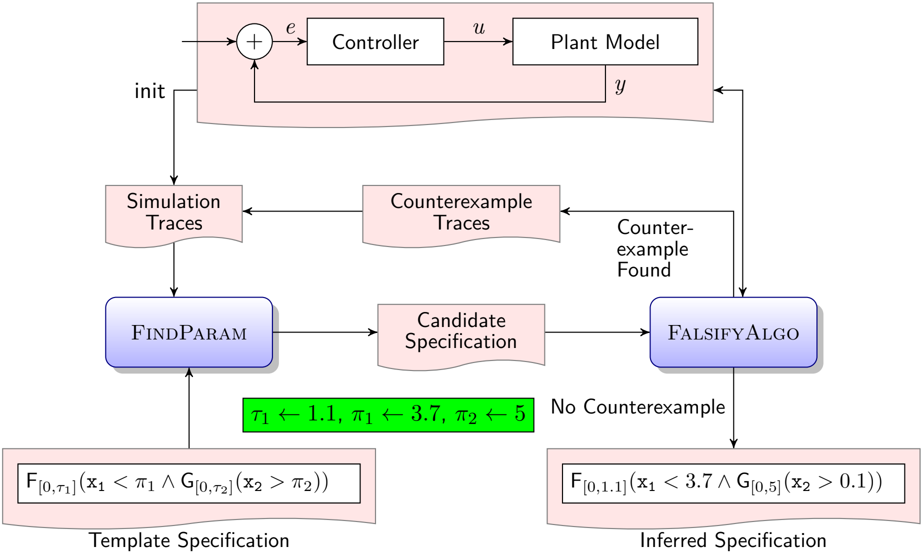
# Specification Synthesis Algorithm



# Specification Synthesis Algorithm



# Specification Synthesis Algorithm



# Results

- *the* speed *is always below* $\pi_1$ *and* RPM *below* $\pi_2$

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

- *the vehicle cannot reach 100 mph in* $\tau$ *seconds with* RPM *always below* $\pi$

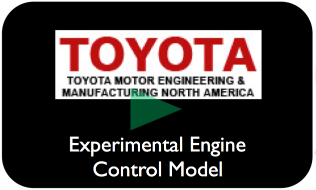
$$\varphi_{\texttt{rpm100}}(\tau, \pi) := \neg(\ \mathsf{F}_{[0,\tau]}\ (\texttt{speed} > 100) \wedge \mathsf{G}(\texttt{RPM} < \pi)).$$

- *whenever it shift to gear* $2$*, it dwells in gear* $2$ *for at least* $\tau$ *seconds*

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

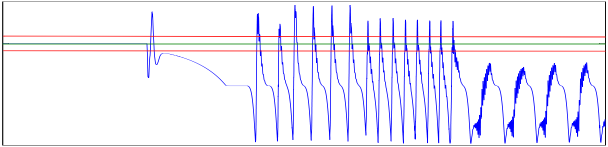
| Template | Parameter values | Fals. | Synth. | #Sim. | Sat./x |
|---|---|---|---|---|---|
| $\varphi_{\texttt{sp\_rpm}}(\pi_1, \pi_2)$ | (155 mph, 4858 rpm) | 197.2 s | 23.1 s | 496 | 0.043 s |
| $\varphi_{\texttt{rpm100}}(\pi, \tau)$ | (3278.3 rpm, 49.91 s) | 267.7 s | 10.51 s | 709 | 0.026 s |
| $\varphi_{\texttt{rpm100}}(\tau, \pi)$ | (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

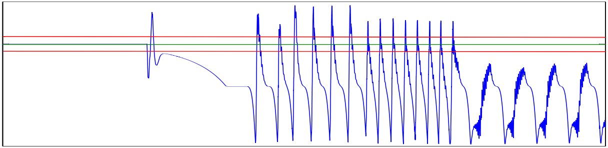



4000+ Simulink blocks
Look-up tables
nonlinear dynamics

- Attempt to mine maximum observed settling time:
  - stops after 4 iterations
  - gives answer $t_{\text{settle}}$ = 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

A lot of work still to be done:

- Online monitoring and mining
- STL and timed/hybrid automa
- Better falsification/optimization of satisfaction functions
- STL templates mining (beyond parameters in PSTL)
- Helping designers writing and using STL