Robustness of Temporal Logic Specifications for Signals

Georgios Fainekos dissertation series - Part I

Akshay Rajhans

ECE Department, CMU

SVC Seminar: Aug 21, 2008

Outline

1. Background and definitions
2. Boolean satisfaction of a specification by a signal - Boolean abstraction
3. Robust satisfaction of a specification by a signal - “Robustness degree”
4. Robust TL semantics - “Robustness estimate”
5. Discrete time signals, timed state sequences and their robustness
6. Continuous time reasoning using discrete time analysis
7. Monitoring algorithm and software tool

Outline

1. Background and definitions
2. Boolean satisfaction of a specification by a signal - Boolean abstraction
3. Robust satisfaction of a specification by a signal - "Robustness degree"
4. Robust TL semantics - "Robustness estimate"
5. Discrete time signals, timed state sequences and their robustness
6. Continuous time reasoning using discrete time analysis
7. Monitoring algorithm and software tool

Background

On the use of temporal logic (TL)

- TL useful in software and hardware verification
- But verification undecidable/expensive in continuous and hybrid systems
- Testing of the systems or numerical simulation of the system models are preferred choices in these cases; steady state properties can be fairly easily tested or numerically simulated

Important idea

- Use TL as a specification language for testing
  - Oded Maler and Dejan Nickovic. *Monitoring temporal properties of continuous signals*. FORMATS, 2004
- Advantage: Transient properties can be specified (and hence tested) if we use temporal logic.

Typical structure for testing using TL

Signal

- We either give an analytical formula or some samples of the signal

Specification

- We use metric interval temporal logic to specify some formula
- Observation map: is a Boolean abstraction map from signal space to true/false
- Intuition: We specify that the signal must be within this range during this time span

Monitoring algorithm

- We have some sort of algorithm to check whether or not the signal was indeed within that range during that time span

Result

- If yes, we get a 'true' result; 'false' otherwise

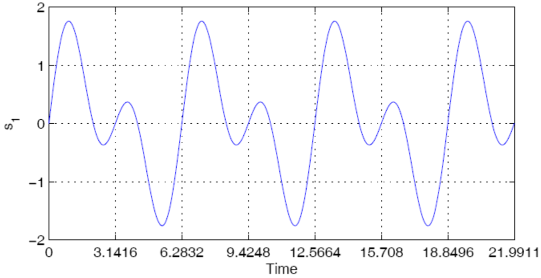
Continuous time signal

Formal definition

- A signal s is a map $s : T \rightarrow X$, where
  - T is a time domain, some subset of $\mathbb{R}_{\geq 0}$
  - X is a metric space, to be defined in the next slide

Example:

$$s_1 = sin(t) + sin(2t), \; T = [0, 7\pi]$$



Metric space, metric, ε-ball

Metric space

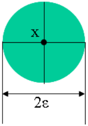
- A metric space (X, d) is an ordered pair of a set X and a metric d.

Metric

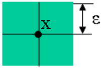
- A metric d is a non-negative function $d : X \times X \to \mathbb{R}_{\geq 0}$, such that $\forall x_1, x_2, x_3 \in X$, we have:
  - $d(x_1, x_2) = 0 \Leftrightarrow x_1 = x_2$
  - $d(x_1, x_2) = d(x_2, x_1)$
  - $d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$

ε-ball

- An ε-ball $B_d(x, \varepsilon)$ is defined as
  - $B_d(x, \varepsilon) = \{y \in X | d(x, y) < \varepsilon\}$



Ball in L_2 or Euclidian norm



Ball in L_∞ or *sup* norm

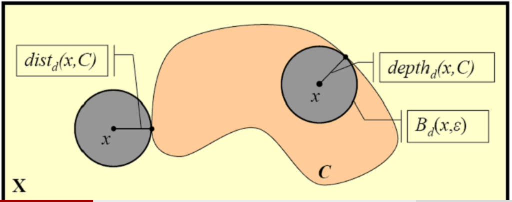
Signed distance

Formal definition

- Let $x \in X$ be a point, $C \subseteq X$ be a set and d be a metric.
- Then, the signed distance from x to C is:
  - $$Dist_d(x, C) = \begin{cases} -dist_d(x, C), & \text{if } x \notin C \\ depth_d(x, C), & \text{if } x \in C \end{cases} \quad (1)$$

    where,
  - $dist_d(x, C) = inf\,\{d(x, y) | y \in Cl(C)\}$
  - $depth_d(x, C) = dist_d(x, X \setminus C)$

Pictorially:



Metric (Interval) Temporal Logic - Syntax

Inductive grammar

$\varphi = \mathsf{T} \mid \perp \mid \mathsf{p} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_{\mathcal{I}} \varphi_2 \mid \varphi_1 \mathcal{R}_{\mathcal{I}} \varphi_2$

Note that:

- In MTL, $\mathcal{I}$ can be any bounded or unbounded but non-empty interval of $\mathbb{R}_{\geq 0}$ e.g. $[a, b], [a, b), (a, b], (a, b)$, where $0 \leq a \leq b$
- In addition, MITL requires $\mathcal{I}$ to be non-singleton, i.e. $a \neq b$
- If $a = 0$ and $b = \infty$, the M(I)TL formula is equivalent to LTL formula
- 'Eventually' and 'Always' operators can be derived as follows:
  - $\Diamond_{\mathcal{I}}\varphi = \mathsf{T}\mathcal{U}_{\mathcal{I}}\varphi$ and $\Box_{\mathcal{I}}\varphi = \perp \mathcal{R}_{\mathcal{I}}\varphi$

Outline

Boolean satisfaction of a specification by a signal

Boolean abstraction

- We specify an observation map
- Observation map labels regions of state space with atomic propositions, e.g. $\mathcal{O}(p_1) = [4, 7]$
  - The signal satisfies p_1 if its value is between 4 and 7, otherwise does not satisfy p_1
- Preimage of observation map: $\mathcal{O}^{-1}(x) = \{p \in AP | x \in \mathcal{O}(p)\}$

Rewriting MITL semantics for testing

- We rewrite $(\mathcal{O}^{-1} \circ s, t) \models \varphi$ as $\ll \varphi, \mathcal{O} \gg = \mathrm{T}$
- If the mapping $\mathcal{O}$ remains constant, we can drop it for brevity and write $\ll \varphi \gg = \mathrm{T}$

Boolean satisfaction of a specification by a signal

Rewriting the MTL grammar for testing:

$$\langle\langle \top \rangle\rangle_C(s,t) := \top$$

$$\langle\langle p \rangle\rangle_C(s,t) := K_{\in}(s(t), \mathcal{O}(p)) = \begin{cases} \top \;\; \text{if } s(t) \in \mathcal{O}(p) \\ \bot \;\; \text{otherwise} \end{cases}$$

$$\langle\langle \neg\phi_1 \rangle\rangle_C(s,t) := \neg\langle\langle \phi_1 \rangle\rangle_C(s,t)$$

$$\langle\langle \phi_1 \vee \phi_2 \rangle\rangle_C(s,t) := \langle\langle \phi_1 \rangle\rangle_C(s,t) \sqcup \langle\langle \phi_2 \rangle\rangle_C(s,t)$$

$$\langle\langle \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2 \rangle\rangle_C(s,t) := \bigsqcup_{t' \in (t +_R \mathcal{I})} \left(\langle\langle \phi_2 \rangle\rangle_C(s,t') \sqcap \prod_{t < t'' < t'} \langle\langle \phi_1 \rangle\rangle_C(s,t'') \right)$$

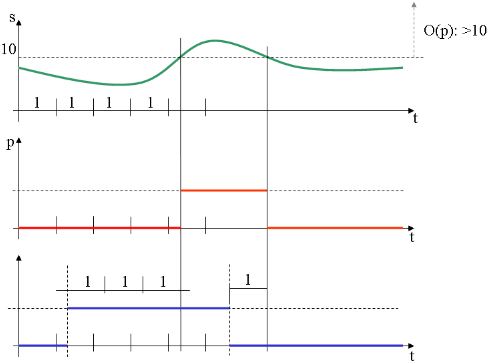
where $t, t', t'' \in R$ *and* $K_{\in}$ *is the characteristic function of the* $\in$ *relation.*

$\sqcap$ means *min* and $\sqcup$ means *max*; Subscript C: continuous time signals

An example

MTL specification $\varphi = \Diamond_{[1,3]} p$

where $\mathcal{O}(p)$ is the set of reals strictly greater than 10



Last graph shows $\ll \varphi \gg (s,t)$ where t is time. When we are talking about $\ll \varphi \gg (s,0)$ we drop 0 for brevity and write $\ll \varphi \gg (s)$

Problems with a Boolean result

Vulnerability to perturbations

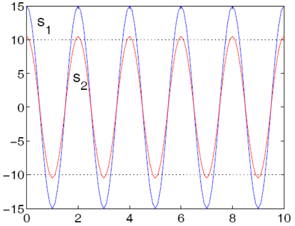


Figure 1.1: Two signals s_1 and s_2 which satisfy the specification: $\Box(\pi_1 \to \Diamond_{\leq 2}\pi_2)$. Here, $\mathcal{O}(\pi_1) = \mathbb{R}_{\leq -10}$ and $\mathcal{O}(\pi_2) = \mathbb{R}_{\geq 10}$.

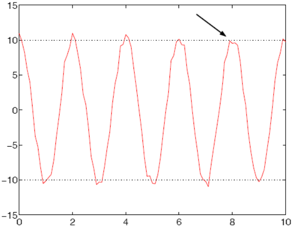


Figure 1.2: The signal s_2 modified by random noise. The arrow points to the point in time where the property fails.

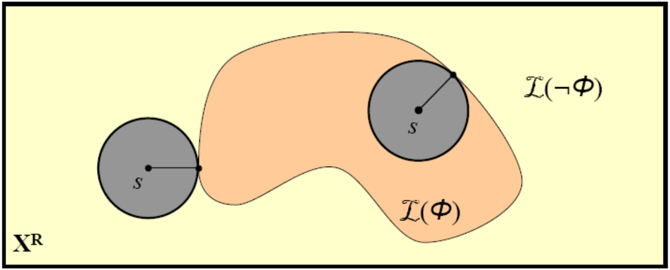
- We cannot distinguish between good and better satisfactions (nor between bad and worse)

Outline

‘Robust’ satisfaction of a specification by a signal

Definition of ‘robustness degree’

- Given a signal s, we define the robustness degree ε as
  - $\varepsilon = Dist_{\rho}(s, \mathcal{L}(\phi))$ [This is a signed distance]
  - where $\rho(s, s') = sup_t \{d(s(t), s'(t)) | t \in T\}$
  - and where $\mathcal{L}(\phi)$ is the set of all signals that satisfy ϕ

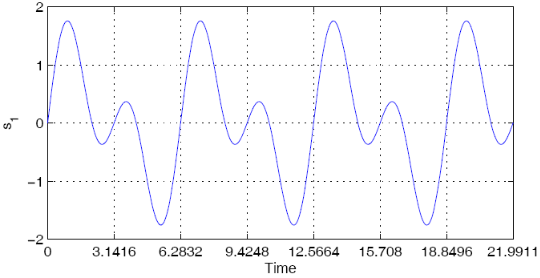


Note that the robustness degree is the radius of the largest (open) ball centered at s that you can fit within $\mathcal{L}(\phi)$

An example where the robustness degree can be computed

A simple example

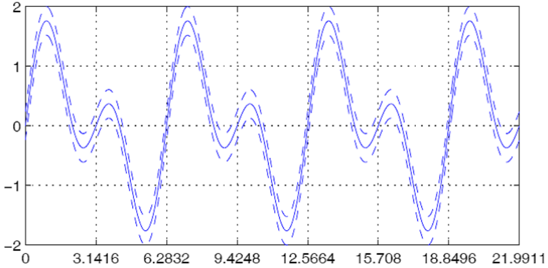
- $s(t) = sin(t) + sin(2t)$
- $\varphi_0 = \Box p_1$ and $\mathcal{O}(p_1) = [-2, 2]$



An example where the robustness degree can be computed

A simple example

- Here, ε can be computed as 0.2398



In general, robustness degree cannot be computed directly, since we don’t know ‘the set of all signals that satisfy the given formula’.

Outline

Multi-valued (aka 'robust') TL semantics

Previous ideas

- De Alfaro et al:
  - Propositions can take values not from $\{0, 1\}$ but in $[0, 1]$
  - Idea used in 'Discounted' model checking - a discount factor between 0 to 1
  - Also used for model checking of say Markov decision processes - transition probabilities between 0 to 1

Idea by Fainekos

- Propositions can take real values
- Details follow...

Robust TL semantics

Inductive grammar

$$[\![\top]\!]_C(s,t) := +\infty$$

$$[\![c]\!]_C(s,t) := c$$

$$[\![p]\!]_C(s,t) := \mathbf{Dist}_d(s(t), \mathcal{O}(p))$$

$$[\![\neg\phi_1]\!]_C(s,t) := -[\![\phi_1]\!]_C(s,t)$$

$$[\![\phi_1 \vee \phi_2]\!]_C(s,t) := [\![\phi_1]\!]_C(s,t) \sqcup [\![\phi_2]\!]_C(s,t)$$

$$[\![\phi_1 \mathcal{U}_{\mathcal{I}} \phi_2]\!]_C(s,t) := \bigsqcup_{t' \in (t +_R \mathcal{I})} \left([\![\phi_2]\!]_C(s,t') \sqcap \bigsqcap_{t<t''<t'} [\![\phi_1]\!]_C(s,t'')\right)$$

$where\ t, t', t'' \in R.$

$\sqcap$ means *inf* and $\sqcup$ means *sup*; Subscript C: continuous time signals

Robustness estimate a lower bound on robustness degree

Important result

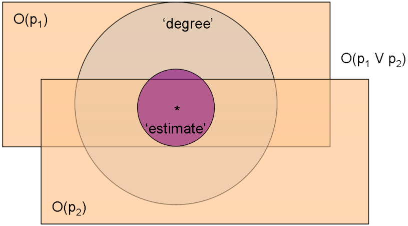
$$\varepsilon_{estimate} = \left| [\![\phi]\!](s) \right| \leq \left| \mathbf{Dist}_{\rho}(s, L(\phi)) \right| = \varepsilon_{actual}$$

which implies

$$\forall s' \in B_\rho(s, \varepsilon), \ll \phi \gg (s', t) = \ll \phi \gg (s, t)$$

‘estimate’ a lower bound on ‘degree’ - why?

By construction of the semantics: An example



- Robustness degree: The radius of the lartest ε-ball we can fit within the $\mathcal{O}(p_1 \vee p_2)$
- Robustness estimate: The semantics ask us to take the *sup* of the radii of the ε-balls we can fit within both observation maps individually.

Outline

Why talk about discrete time signals?

Practical reasons

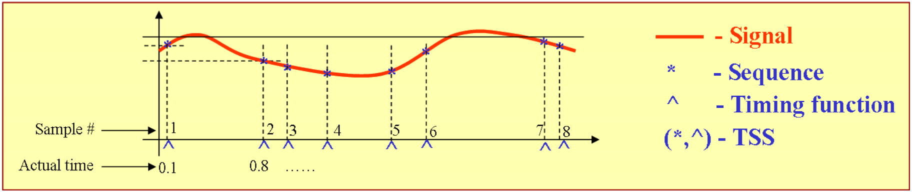
- We may not know the analytical equation of the signal
- We may not have access or to all the (infinite) values of a continuous signal even on a finite real time domain
- All we might have is a number of samples and the corresponding time stamps
- Typical example: result of a variable step numerical ODE integration in Matlab.

Timed State Sequences (TSS)

In words:

- Discrete time signal σ is a sequence of samples, no timing info
- A timing function τ associates time with each sample
- A pair μ=(σ,τ) is called a timed state sequence

Pictorially:



MITL semantics for testing

MITL semantics for TSS

$$\langle\langle \top \rangle\rangle_D(\mu, i) := \top$$

$$\langle\langle p \rangle\rangle_D(\mu, i) := K_\in(\sigma(i), \mathcal{O}(p))$$

$$\langle\langle \neg\phi_1 \rangle\rangle_D(\mu, i) := \neg\langle\langle \phi_1 \rangle\rangle_D(\mu, i)$$

$$\langle\langle \phi_1 \vee \phi_2 \rangle\rangle_D(\mu, i) := \langle\langle \phi_1 \rangle\rangle_D(\mu, i) \sqcup \langle\langle \phi_2 \rangle\rangle_D(\mu, i)$$

$$\langle\langle \phi_1 \mathcal{U}_\mathcal{I} \phi_2 \rangle\rangle_D(\mu, i) := \bigsqcup_{j \in \tau^{-1}(\tau(i)+\mathcal{I})} \left(\langle\langle \phi_2 \rangle\rangle_D(\mu, j) \sqcap \bigsqcap_{i \le k < j} \langle\langle \phi_1 \rangle\rangle_D(\mu, k) \right)$$

where $i, j, k \in N$, $\sigma = \mu^{(1)}$, $\tau = \mu^{(2)}$ *and* $K_\in$ *is the characteristic function of the* $\in$ *relation.*

Robust semantics

Robust semantics for TSS

$$[\![\top]\!]_D(\mu, i) := +\infty$$

$$[\![c]\!]_D(\mu, i) := c$$

$$[\![p]\!]_D(\mu, i) := \mathbf{Dist}_d(\sigma(i), \mathcal{O}(p))$$

$$[\![\neg\phi_1]\!]_D(\mu, i) := -[\![\phi_1]\!]_D(\mu, i)$$

$$[\![\phi_1 \vee \phi_2]\!]_D(\mu, i) := [\![\phi_1]\!]_D(\mu, i) \sqcup [\![\phi_2]\!]_D(\mu, i)$$

$$[\![\phi_1 \mathcal{U}_{\mathcal{I}} \phi_2]\!]_D(\mu, i) := \bigsqcup_{j \in \tau^{-1}(\tau(i)+\mathcal{I})} \left([\![\phi_2]\!]_D(\mu, j) \sqcap \bigsqcap_{i \le k < j} [\![\phi_1]\!]_D(\mu, k)\right)$$

where $i, j, k \in N$, $\sigma = \mu^{(1)}$ *and* $\tau = \mu^{(2)}$.

Outline

Why are DT and CT semantics NOT equivalent?

Consider the DT Until operator

$$\langle\langle \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2 \rangle\rangle_D(\mu, i) := \bigsqcup_{j \in \tau^{-1}(\tau(i)+\mathcal{I})} \left(\langle\langle \phi_2 \rangle\rangle_D(\mu, j) \sqcap \bigsqcap_{i \leq k < j} \langle\langle \phi_1 \rangle\rangle_D(\mu, k) \right)$$

where $i, j, k \in N$, $\sigma = \mu^{(1)}$ *and* $\tau = \mu^{(2)}$.

Observations:

- The actual interval and the samples within that interval do not coincide
- If we have no sample within some interval, TSS cannot capture the properties of the original signal

Can we force DT and CT equivalence?

Strengthening of formulas

- Idea introduced by Huang et al
  - Jinfeng Huang, Jeroen Voeten, and Marc Geilen, *Real-time property preservation in approximations of timed systems*, Conference on Formal Methods and Models for Co-Design, 2003
- Satisfaction of a strengthened formula by a TSS will guarante (under certain assumptions) the satisfaction of the original formula by the signal

Assumptions on signal behavior - bounded spread

- Intuitively: Signal doesn't spread infinitely in a finite duration

Assumptions for sampling - at least one sample per interval

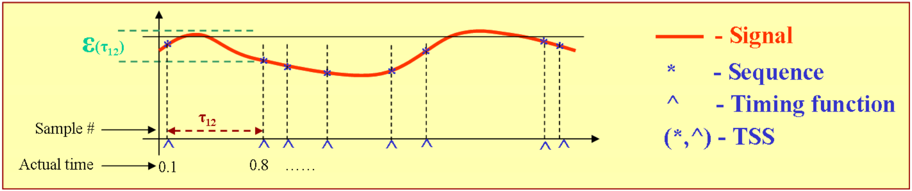
- We have enough data to build on

Let us look at these assumptions in detail...

Well-behavedness assumptions

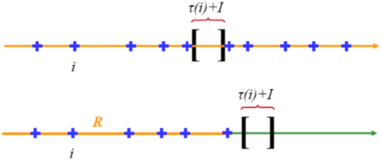
Bounded spread

$\forall t, t' \in R, d(s(t), s(t')) \leq \mathcal{E}(|t - t'|)$, R: signal domain on real number line



At least one sample per interval

Disallowed cases (pictorially):



No sample in the interval

Empty intersection with the signal domain R

Strengthening of formulas with resepect to time

Main idea: Satisfying strenghened formula in DT $\Rightarrow$ satisfying the original (weaker) formula in CT

Atomic predicates and their Boolean combinations

No direct strenghening needed (no 'interval' for these operators)

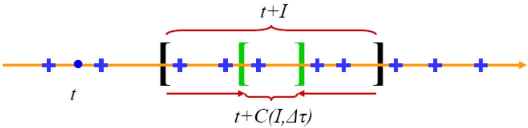
- $\mathbf{str}_{\Delta\tau}(p) = p$
- $\mathbf{str}_{\Delta\tau}(\neg p) = \neg p$
- $\mathbf{str}_{\Delta\tau}(\varphi_1 \vee \varphi_2) = \mathbf{str}_{\Delta\tau}(\varphi_1) \vee \mathbf{str}_{\Delta\tau}(\varphi_2)$
- $\mathbf{str}_{\Delta\tau}(\varphi_1 \wedge \varphi_2) = \mathbf{str}_{\Delta\tau}(\varphi_1) \wedge \mathbf{str}_{\Delta\tau}(\varphi_2)$

Let's see the cases where strengthening is needed, in detail on the next slide...

Strengthening of formulas with respect to time

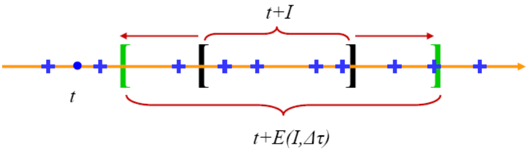
'Until' operator: Compress by $\Delta\tau$ [where $\Delta\tau = sup_i(\tau_{i+1} - \tau_i)$]

- $\mathbf{str}_{\Delta\tau}(\varphi_1 \mathcal{U}_{\mathcal{I}} \varphi_2) = \mathbf{str}_{\Delta\tau}(\varphi_1)\ \mathcal{U}_{C(\mathcal{I},\Delta\tau)}\ \mathbf{str}_{\Delta\tau}(\varphi_2)$



'Release' operator: Expand by $\Delta\tau$

- $\mathbf{str}_{\Delta\tau}(\varphi_1 \mathcal{R}_{\mathcal{I}} \varphi_2) = \mathbf{str}_{\Delta\tau}(\varphi_1)\ \mathcal{R}_{E(\mathcal{I},\Delta\tau)}\ \mathbf{str}_{\Delta\tau}(\varphi_2)$



DT - CT equivalence

Given a specification and a TSS, if

- we know the value of $\Delta\tau$ and strengthen the specification by $\Delta\tau$ and
- well-behavedness assumptions are satisfied post-strengthening and
- we (somehow) know $\mathcal{E}(\Delta\tau)$ and
- and we find the robustness estimate of the given TSS on this strengthened specification

and if

- the robustness estimate of the TSS for the strengthened specification turnes out to be greater than $\mathcal{E}(\Delta\tau)$

then

- the original continuous time signal satisfies the original specification

Outline

Monitoring algorithm

A recursive algorithm

Algorithm 1 Monitoring the Robustness of Timed State Sequences

Input: An MTL formula ϕ, a finite timed state sequence $\mu = (\sigma, \tau)$ and a predicate map $\mathcal{O}$

Output: The formula's robustness estimate

procedure MONITOR($\phi, \mu, \mathcal{O}$)
    $i \leftarrow 0$
    while $\phi \neq \varepsilon \in \overline{\mathbb{R}}$ **do** $\qquad \triangleright$ ϕ has not been reduced to a value
        if $i < \max \mathbf{dom}(\tau)$ **then** $\phi \leftarrow$ DERIVE($\phi, \sigma(i), \delta\tau(i), \perp, \mathcal{O}$)
        else $\phi \leftarrow$ DERIVE($\phi, \sigma(i), 0, \top, \mathcal{O}$)
        end if
        $i \leftarrow i + 1$
    end while
end procedure

Monitoring algorithm

A recursive algorithm (continued...)

Algorithm 2 Deriving the Future

Input: The MTL formula ϕ, the current value of the signal x, the time period δt before the next value in the signal, a variable *last* indicating whether the next state is the last and the predicate map $\mathcal{O}$

Output: The MTL formula ϕ that has to hold at the next moment in time

procedure $\textsc{Derive}(\phi, x, \delta t, last, \mathcal{O})$
  if $\phi = \top$ **then return** $+\infty$
  else if $\phi = \varepsilon \in \overline{\mathbb{R}}$ **then return** ε
  else if $\phi = p \in AP$ **then return** $\mathbf{Dist}_d(x, \mathcal{O}(p))$
  else if $\phi = \neg\phi_1$ **then return** $\neg\textsc{Derive}(\phi_1, x, \delta t, last, \mathcal{O})$
  else if $\phi = \phi_1 \vee \phi_2$ **then**
    return $\textsc{Derive}(\phi_1, x, \delta t, last, \mathcal{O}) \vee \textsc{Derive}(\phi_2, x, \delta t, last, \mathcal{O})$
  else if $\phi = \phi_1 \mathcal{U}_{\mathcal{I}} \phi_2$ **then**
    $\alpha \leftarrow K_{\in}^{\infty}(0, \mathcal{I}) \wedge \textsc{Derive}(\phi_2, x, \delta t, last, \mathcal{O})$
    if $last = \top$ **then return** α
    else return $\alpha \vee (\textsc{Derive}(\phi_1, x, \delta t, last, \mathcal{O}) \wedge \phi_1 \mathcal{U}_{(-\delta t)+_R \mathcal{I}} \phi_2)$
    end if
  end if
end procedure

Software tool

TaLiRo

- Computes the robustness estimate
- Takes MTL specifications as input
- Can handle 1D signals as of now
- Can handle polytopic observation maps
- Available at:

  `http://www.seas.upenn.edu/~fainekos/robustness.html`

Summary

Take-away messages from this talk

- Multi-valued TL semantics make the use of TL more robust in testing
- Hopefully could help to popularize the use of TL beyond purely discrete systems, into continuous and hybrid systems :-)

Stay tuned for part II talk

- Exciting new extensions possible
- We will discuss: Verification using robust testing
- We will also briefly review approximate bisimulations
- Stay tuned. . .

References

Georgios Fainekos's work

- All credit should go to Georgios Fainekos, this is his work.
- On the other hand, if there were any mistakes, they were most likely mine.

References

- Some figures and formulas were taken from the thesis and talk slides by Georgios.
- The references i.e. the presentations, publications and other interesting reference material is available at:

  `http://www.seas.upenn.edu/~fainekos/the_public.html`

Thank you

Thank you

- Thanks to Ed Clarke for hosting me
- Thanks to Bruce Krogh and Alex Donzé for reviewing the slides
- Thank you all for attending