Using Temporal Logics in CPS Autograders

Alexandre Donzé

University of California, Berkeley

October 15, 2015

Grading a CPS Model Design

Purpose of grading

1. Does the design solve the assignement ?
2. In case of imperfect design, provide a hint/explanation of what is wrong.

Grading a CPS Model Design

Purpose of grading

1. Does the design solve the assignement ?

   Model Checking

2. In case of imperfect design, provide a hint/explanation of what is wrong.

Grading a CPS Model Design

Purpose of grading

1. Does the design solve the assignement ?

   Model Checking

2. In case of imperfect design, provide a hint/explanation of what is wrong.

   Counter-examples

Model Checking

Model satisfies ? Specifications

Model Checking

Model satisfies ? Specifications

Classical Setting

- Model is a Finite State Machine (FSM)
- Specifications are LTL formulas

Model Checking

Model satisfies ? Specifications

Classical Setting

- Model is a Finite State Machine (FSM)
- Specifications are LTL formulas

CPS Setting

- Model is a hybrid system
- Specifications are Signal Temporal Logic (STL) formulas

From Model Checking to Runtime Monitoring

Problem: Model checking STL is intractable for complex CPS

From Model Checking to Runtime Monitoring

Problem: Model checking STL is intractable for complex CPS

Runtime Monitoring

- A more tractable approach: simulation + monitoring

$$\boxed{\text{Model}} \longrightarrow aaaabbbaa\ldots \longrightarrow \boxed{\text{Property } \varphi} \longrightarrow true/false$$

From Model Checking to Runtime Monitoring

Problem: Model checking STL is intractable for complex CPS

Runtime Monitoring

- A more tractable approach: simulation + monitoring

$$\boxed{\text{Model}} \longrightarrow aaaabbbaa\ldots \longrightarrow \boxed{\text{Property } \varphi} \longrightarrow true/false$$

- Think of (incomplete) depth-first search

From Model Checking to Runtime Monitoring

Problem: Model checking STL is intractable for complex CPS

Runtime Monitoring

- A more tractable approach: simulation + monitoring

$$\boxed{\text{Model}} \longrightarrow aaaabbbaa\ldots \longrightarrow \boxed{\text{Property } \varphi} \longrightarrow true/false$$

- Think of (incomplete) depth-first search

$\Rightarrow$ can find counter-examples but usually not *prove* the system correct

Monitoring LTL

Let's evaluate some LTL formula on the trace $w = aaabbaaa$.

w	a	a	a	b	b	a	a	a	$\ldots$
a	$true$	$true$	$true$	$false$	$false$	$true$	$true$	$true$	$\ldots$
b	$false$	$false$	$false$	$true$	$true$	$false$	$false$	$false$	$\ldots$

Monitoring LTL

Let's evaluate some LTL formula on the trace $w = aaabbaaa$.

Recall that a formula is true for a sequence if it is true for the first element.

w	a	a	a	b	b	a	a	a	$\dots$
a	$true$	$true$	$true$	$false$	$false$	$true$	$true$	$true$	$\dots$
b	$false$	$false$	$false$	$true$	$true$	$false$	$false$	$false$	$\dots$
$X\ b$	?								$\dots$

Monitoring LTL

Let's evaluate some LTL formula on the trace $w = aaabbaaa$.

Recall that a formula is true for a sequence if it is true for the first element.

w	a	a	a	b	b	a	a	a	$\ldots$
a	$true$	$true$	$true$	$false$	$false$	$true$	$true$	$true$	$\ldots$
b	$false$	$false$	$false$	$true$	$true$	$false$	$false$	$false$	$\ldots$
$X\ b$	$false$								$\ldots$

Monitoring LTL

Let's evaluate some LTL formula on the trace $w = aaabbaaa$.

Recall that a formula is true for a sequence if it is true for the first element.

w	a	a	a	b	b	a	a	a	$\dots$
a	$true$	$true$	$true$	$false$	$false$	$true$	$true$	$true$	$\dots$
b	$false$	$false$	$false$	$true$	$true$	$false$	$false$	$false$	$\dots$
$X\ b$	$false$	$false$	$false$	$true$	$true$	$false$	$false$	$?$	$\dots$

Monitoring LTL

Let's evaluate some LTL formula on the trace $w = aaabbaaa$.

Recall that a formula is true for a sequence if it is true for the first element.

w	a	a	a	b	b	a	a	a	$\ldots$
a	$true$	$true$	$true$	$false$	$false$	$true$	$true$	$true$	$\ldots$
b	$false$	$false$	$false$	$true$	$true$	$false$	$false$	$false$	$\ldots$
Xb	$false$	$false$	$true$	$true$	$false$	$false$	$false$	$?$	$\ldots$
alw a	$false$	$false$	$false$	$false$	$false$	$true?$	$true?$	$true?$	$\ldots$

Monitoring LTL

Let's evaluate some LTL formula on the trace $w = aaabbaaa$.

Recall that a formula is true for a sequence if it is true for the first element.

w	a	a	a	b	b	a	a	a	$\ldots$
a	$true$	$true$	$true$	$false$	$false$	$true$	$true$	$true$	$\ldots$
b	$false$	$false$	$false$	$true$	$true$	$false$	$false$	$false$	$\ldots$
Xb	$false$	$false$	$true$	$true$	$false$	$false$	$false$	$?$	$\ldots$
alw a	$false$	$false$	$false$	$false$	$false$	$true?$	$true?$	$true?$	$\ldots$
ev b	$true$	$true$	$true$	$true$	$true$	$false?$	$false?$	$false?$	$\ldots$

Monitoring LTL

Let's evaluate some LTL formula on the trace $w = aaabbaaa$.

Recall that a formula is true for a sequence if it is true for the first element.

w	a	a	a	b	b	a	a	a	$\ldots$
a	$true$	$true$	$true$	$false$	$false$	$true$	$true$	$true$	$\ldots$
b	$false$	$false$	$false$	$true$	$true$	$false$	$false$	$false$	$\ldots$
Xb	$false$	$false$	$true$	$true$	$false$	$false$	$false$	$?$	$\ldots$
alw a	$false$	$false$	$false$	$false$	$false$	$true?$	$true?$	$true?$	$\ldots$
ev b	$true$	$true$	$true$	$true$	$true$	$false?$	$false?$	$false?$	$\ldots$
$a\ \mathbf{U}\ b$	$true$	$true$	$true$	$false$	$false$	$false?$	$false?$	$false?$	$\ldots$

Outline

1. Signal Temporal Logic

2. CPSGrader: Writing Temporal Testers

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

Signals are functions from $\mathbb{R}$ to $\mathbb{R}$.

E.g.: positions (x,y,z), orientation θ, sensor values (acc. ax,ay,az), etc.

We denote by $x[\tau]$ the value of signal x at time τ.

STL: Syntax

Signals are functions from $\mathbb{R}$ to $\mathbb{R}$.

E.g.: positions (x,y,z), orientation θ, sensor values (acc. ax,ay,az), etc.

We denote by $x[\tau]$ the value of signal x at time τ.

Atomic predicates are inequalities over signal values at **symbolic** time t

E.g.: $x[t] > 0.5$, $z[t] < 4$, $|lws[t] + rws[t]| > 100$, etc.

STL: Syntax

Signals are functions from $\mathbb{R}$ to $\mathbb{R}$.

E.g.: positions (x,y,z), orientation θ, sensor values (acc. ax,ay,az), etc.

We denote by $x[\tau]$ the value of signal x at time τ.

Atomic predicates are inequalities over signal values at **symbolic** time t

E.g.: $x[t] > 0.5$, $z[t] < 4$, $|lws[t] + rws[t]| > 100$, etc.

Temporal operators are $\mathbf{F}$, $\mathbf{G}$, $\mathbf{U}$, equiped with a time interval

e.g. $\mathbf{F}_{[0,2]}(x[t] > 0.5)$, $\mathbf{G}_{[0,40]}(y[t] < 0.3)$, $\varphi\mathbf{U}_{[1,2.5]}\psi$, etc.

Remark: no "next" X ?

STL Semantics

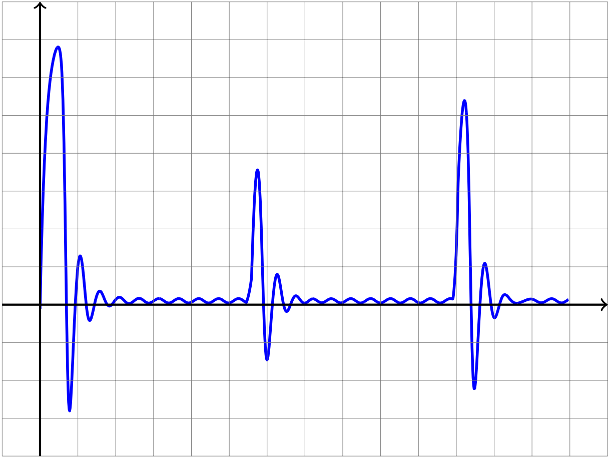
A **formula** φ is true if it is true **at time 0**

A **subformula** ψ is evaluated on **future values** depending on temporal operators

Examples

- $\varphi = (x[t] > 0.5)$ is true iff $x[t] > 0.5$ is true when t is replaced by 0, i.e., at the first value of the signal.
- $\varphi = \mathbf{F}_{[0,1.3]}(x[t] > 0.5)$ is true iff $x[t] > 0.5$ is true when t is replaced by any value in [0,1.3].
- $\varphi = \mathbf{G}_{[0,1.3]}(\psi)$ is true iff ψ is true at all time in $[0, 1.3]$, i.e., for all suffixes of signals starting at a time in $[0, 1.3]$

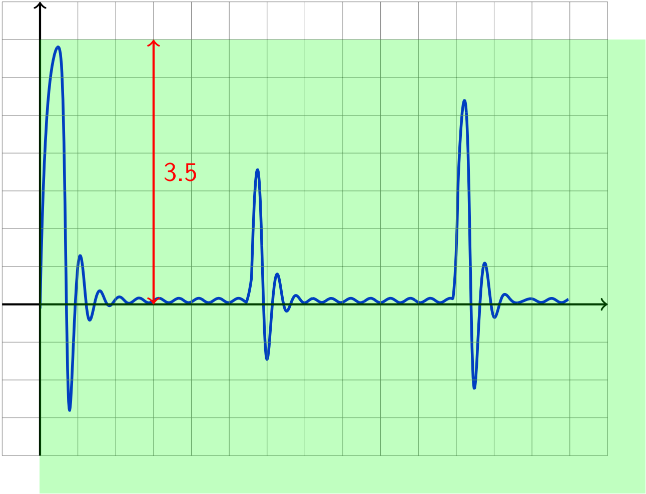
STL Examples



STL Examples

The signal is never above 3.5

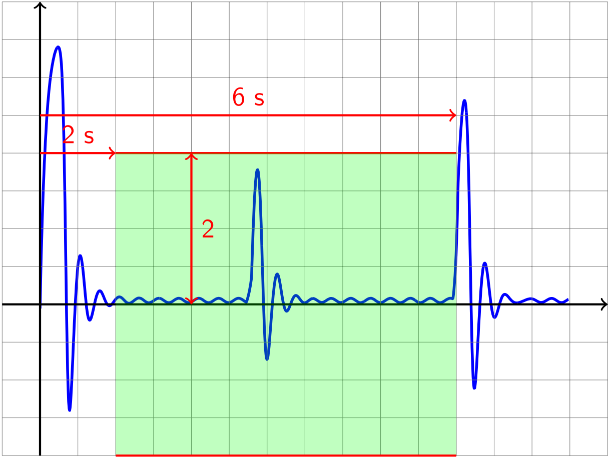
$$\varphi := \mathsf{alw}\ (x[t] < 3.5)$$



STL Examples

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

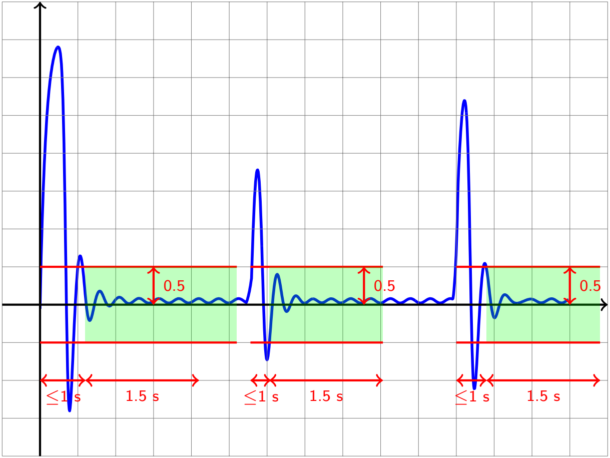
$$\varphi := \text{alw}_{[2,6]} \ \ (|x[t]| < 2)$$



STL Examples

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

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



CPSGrader test plans

Grading is based on test plans comprizing:

Test traces

System traces obtained in a specific environment setting.

They should cover all situations relevant to the design requirement.

CPSGrader test plans

Grading is based on test plans comprizing:

Test traces

System traces obtained in a specific environment setting.

They should cover all situations relevant to the design requirement.

Fault monitors

STL properties characterizing faults in the design.

They should detect any behavior of the design indicative of known faults.

Known faults include not satisfying the design requirement

CPSGrader test plans

The general structure of a test plan is as follows:

```
# signal, parameters and formula declarations
...
# test declarations
test test1 {
   fault1 { ...
   fault2 { ...
   ...
}
test test2 {  ...
}
...
test testN { ...
}
```

CPSGrader: executing test plans

CPSGrader will execute test plans as follows

For each test trace
  Get trace $\mathbf{x}$ from simulator
  For each fault with STL formula φ
    Check whether $\mathbf{x} \models \varphi$
    Print feedback
    If fault is critical **then** return
  end
end

CPSGrader: A concrete example with CyberSim

First, declare signals, parameters and STL formulas and subformulas:

```
# declare signals used in formulas
signal x,y

# Defines some parameters
param y_min= 3., x_max = 5.

# sub formula: defining an (x,y) region which goal is to leave
in_region_to_leave := (y[t]<y_min) or (x[t]>x_max)

# top formula
phi_goal_missed  := alw_[0, 20] (in_region_to_leave)
```

CPSGrader: A concrete example with CyberSim

Second, define tests and faults.

```
# Defining the test
test nav1: "Environment - obstacle south left.xml", 20.1, true
{
  fault_goal_missed                       #name of fault
  {phi_goal_missed,                       #formula to monitor
   "PROBLEM:Couldn't avoid obstacle",     #feedback if true
   "",                                    #feedback if false
   true                                   #feedback is critical?
  }
}
```

Demo

Demo

- CyberSim/CPSGrader autograder and test plans
  ⇒ modify `data\feedback_nav.stl` or `data\feedback_nav_hill.stl`
- Exploring properties using Breach (`www.eecs.berkeley.edu/~donze/breach_page.html`)
  ⇒ at the Matlab prompt `>> Breach('feedback_nav.stl')` Then menu `Files->Import Trajectory from File`

Conclusion

- Tools:
  CPSGrader: `cpsgrader.org`
  - CyberSim
  - Other simulator or data: monitoring trace files

  Breach: `www.eecs.berkeley.edu/~donze/breach_page.html`
  - Simulink models + other simulators via simple wrapper
  - Richer GUI to work with STL formulas
- Suggestions of faults/feedback expressible in STL that would have helped you debug your controller?
- Project idea where CyberSim/CPSGrader or Breach could help with better testing your design?
- Send e-mail to (`donze@berkeley.edu`) for any related question