CoCoSTLib Manual

Yu XIA
2021
Version 1

Abstract

This document describes the capabilities of the `refart` and `refrep` classes for LaTeX 2_ε. These classes do not work with LaTeX 2.09. They contain some improvements over the original `refman` style which may result in different output and minor incompatibilities, but make refman work with paper sizes other than ISO A4, which I consider an improvement.

Contents

1 Introduction

CoCoSTLib is a Simulink Custom Library used for test and verify STL properties.

Note	This library is built based on SignalTemplateLibraryAutogen. Certain components are properties of SignalTemplateLibraryAutogen. `https://github.com/balsini/SignalTemplateLibraryAutogen`

1.1 Application

This library can be used for continuous/discrete/hybrid systems. In two latter cases, it might be asked to adjust sample time settings.

1.2 Index

If you want to use a layout different from the ones distributed with LaTeX, you have to take the following steps:

→ Chapter 2 — You can find detailed information about how change the design in chapter 2.

→ Chapter 3 — STL 3.

→ Chapter 4 — Chapter 4 contains description, characteristics and usage of each block in CoCoSTLib.

→ Chapter 6 — Chapter 6 illustrates.

2 Installation

2.1 Prerequisite

MATLAB and Simulink : version **R2019b** or newer is recommended.

CoCoSim : optional. CoCoSim (Contract based Compositional verification of Simulink models) is a Matlab Toolbox performing verification and validation of Simulink and Stateflow models. For more information see `https://github.com/NASA-SW-VnV/CoCoSim`.

CoCoSTLibis initially designed as component of CoCoSim. It can be used outside of CoCoSim framework.

2.2 Run CoCoSTLib

As a Simulink custom library, CoCoSTLib does not require installation. To use it, simply copy or drag any block(s) from `CoCoSTLib.slx`.

In addition, adding CoCoSTLib to the Library Browser will greatly increase the accessibility. To do so, you need to:

1. Make sure **slblocks.m** and **add2browser.m** are in the same folder as **CoCoSTLib.slx**;
2. Add this folder to path see `https://www.mathworks.com/help/matlab/ref/addpath.html`;
3. Run the script **add2browser.m**;
4. Open the Library Browser by clicking Library Browser in the Simulink Toolstrip. To see the new library in the Library Browser, right-click the library list and select **Refresh Library Browser**.

→ More Info

Mathworks Help Center - addpath

Mathworks Help Center - Add Libraries to the Library Browser

3 Signal Temporal Logics

3.1 Temporal Logics in a Nutshell

In the late 1970s, temporal logic was introduced to specify patterns of timed behaviors of systems. At this stage, temporal logic, namely Linear Temporal Logic or Linear-time Temporal Logic (LTL), deals with discrete sequences of states, i.e. discrete-time and discrete-valued systems. The fundamental elements are logic operators ($\neg$, $\wedge$, $\vee$) and temporal operators: "Next" (denoted as X or $\bigcirc$), "Always" (or "Globally", denoted as G or $\square$), "Eventually" (or "Finally", denoted as F or $\Diamond$) and "until" (denoted as $\mathcal{U}$).

Temporal logics have gained great success in formal verification of programs and hardware digital circuits in a short time, and most on-going researches in model checking are aiming at software, analog/mixed-signal circuits, systems biology, and CPS. In other words, the tendency is to move from discrete-time discrete-valued systems to hybrid (discrete-continuous) systems.

Regarding the relations between MTL and STL, Kapinski et al. explain: "The syntax for the logic MTL is similar to STL. The only difference is that MTL requires that formulas be defined over Boolean signals; continuous-valued signals can be considered by converting them to Boolean signals based on given logical predicates over the continuous signals. A key feature of MTL and STL is that both logics are equipped with quantitative semantics, which is a function mapping a given signal trace and an STL/MTL formula ψ to a real value. This value is an indicator of the degree of satisfaction of ψ; positive values indicate that the trace satisfies ψ, negative values denote violation of ψ, and the magnitude indicates the robustness margin. In other words, a positive value δ indicates that the signal can be perturbed by up to δ before it violates ψ. STL and MTL differ in how they define the signed distance of a signal trace from an atomic predicate, which impacts the computational complexity of the quantitative semantics for these logics."

3.2 STL Syntax and Semantics

In STL, a formula ϕ is evaluated on a sequence of inputs $\mathcal{X} = (x_1, x_2, ..., x_n)$ at a (continuous) time instant t, resulting in the evaluation of $(\mathcal{X}, t)$ pairs. An STL formula ϕ can be:

- p: a proposition that evaluates a state variable.

$$(\mathcal{X}, t) \models p \Leftrightarrow p[t] = TRUE$$

- $\neg\phi$ (Negation): the logical negation of ϕ.

$$(\mathcal{X}, t) \models \neg\phi \Leftrightarrow \neg((\mathcal{X}, t) \models \phi)$$

- $\phi_1 \wedge \phi_2$ (And): the logical and between ϕ_1 and ϕ_2.

$$(\mathcal{X}, t) \models \phi_1 \wedge \phi_2 \Leftrightarrow (\mathcal{X}, t) \models \phi_1 \wedge (\mathcal{X}, t) \models \phi_2$$

- $\phi_1 \mathcal{U} \phi_2$ (Until): a temporal operator that is satisfied if ϕ_1 holds until ϕ_2 becomes true.

$$(\mathcal{X}, t) \models \phi_1 \mathcal{U} \phi_2 \Leftrightarrow \exists t' \geq t : (\mathcal{X}, t') \models \phi_2 \wedge \forall t'' \in [t, t') : (\mathcal{X}, t'') \models \phi_1$$

From the previous primitive operators, it is possible to derive other temporal operators:

- $\Diamond\phi = \text{TRUE}\ \mathcal{U}\phi$ (Eventually): the condition is verified at least once.
$$(\mathcal{X}, t) \models \Diamond\phi \Leftrightarrow \exists t' \geq t : (\mathcal{X}, t') \models \phi$$
- $\Box\phi = \neg(\Diamond\neg\phi)$ (Always/Globally): the condition is always verified.
$$(\mathcal{X}, t) \models \Box\phi \Leftrightarrow \forall t' \geq t : (\mathcal{X}, t') \models \phi$$

In STL, temporal operators may be bounded inside an implicit $[0, +\infty)$ or explicitly specified time interval. The Until operator with an interval bound has the meaning

$$(\mathcal{X}, t) \models \phi_1 \mathcal{U}_{[a,b]} \phi_2 \Leftrightarrow \exists t' \in [t+a, t+b] : (\mathcal{X}, t') \models \phi_2 \wedge \forall t'' \in [t, t'] : (\mathcal{X}, t'') \models \phi_1$$

from which is possible to obtain the following relations.

$$\Diamond_{[a,b]}\phi = \text{TRUE}\,\mathcal{U}_{[a,b]}\phi$$

$$\Box_{[a,b]}\phi = \neg(\Diamond_{[a,b]}\neg\phi)$$

In STL, timed formulas can be nested such as, for example

$$\Diamond_{[0,T]}(\varphi \wedge \Diamond_{[a,b]}\psi)$$

The proposition ψ is nested one level deeper than proposition φ. The semantics is that there has to be one time instant t in $[0, T]$ (the outer Eventually condition) such that φ is satisfied at t, and the proposition ψ is verified at some time in $[t + a, t + b]$. To put it another way, $[a, b]$ is relative to when φ is satisfied.

4 Blocks Reference

Timed Operators,

ϕ is proposition

$\mathcal{X}$ is signal under concern

4.1 Time Bounds

4.1.1 Time Window

Symbol :

Description :

Inputs :

Outputs :

4.1.2 Delayed Time Window

Symbol :

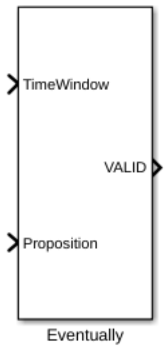
Description :

Inputs :

Outputs :

4.2 Operators

4.2.1 Eventually



Description : Within time interval $[a, b]$, there exits one instant that the signal under concern is TRUE.

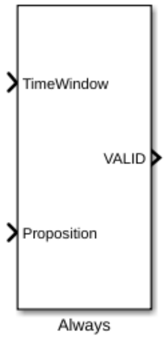
$$\diamond_{TimeWindow}\ Proposition$$

$$(\mathcal{X}, t) \vDash \diamond_{[a,b]}\phi = \exists t' \in [a, b]\ s.t\ (\mathcal{X}, t') \vDash \phi$$

Inputs :

Outputs :

4.2.2 Always



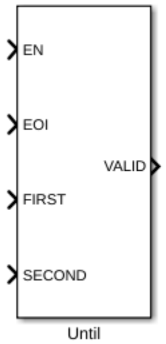
Description : Within time interval $[a, b]$, at any instant, the signal under concern is TRUE.

$$(\mathcal{X}, t) \vDash \Box_{[a,b]}\phi = \forall t \in [a, b] \; s.t \; (\mathcal{X}, t) \vDash \phi$$

Inputs :

Outputs :

4.2.3 Until



Description : The first signal is TRUE until one instant within $[a, b]$ when the second signal is TRUE.

$$FIRST \; Until_{[EN,EOI]} \; SECOND$$

$$(\mathcal{X}, t) \vDash \phi_1 \mathcal{U}_{[a,b]} \phi_2 \Leftrightarrow \exists t' \in [t+a, t+b] : (\mathcal{X}, t') \vDash \phi_2 \wedge \forall t'' \in [t, t'] : (\mathcal{X}, t'') \vDash \phi_1$$

Inputs :

Outputs :

4.2.4 AND

Symbol :

Description :

Inputs :

Outputs :

5 How To

6 Examples

Index