S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems*

Yashwanth Annapureddy[1], Che Liu[1], Georgios Fainekos[1]
Sriram Sankaranarayanan[2]

[1] Arizona State University, Tempe, AZ.
{Yashwanthsingh.Annapureddy,Che.Liu,fainekos}@asu.edu.
[2] University of Colorado, Boulder, CO. srirams@colorado.edu

Abstract. S-TaLiRo is a Matlab toolbox that searches for falsifying trajectories of temporal logic properties of Simulink/Stateflow models. It can analyze arbitrary Simulink models or user defined functions that model the system. At the heart of the tool, we use randomized testing based on stochastic optimization techniques including Monte-Carlo methods and ant-colony optimization. Among the advantages of the toolbox is the seamless integration inside the Matlab environment, which is widely used in the industry for model-based development of control software. We present the architecture of S-TaLiRo and its working on an application example.

1 Introduction

Temporal verification involves the ability to prove as well as falsify temporal logic properties of systems. In this paper, we present our tool S-TaLiRo[1] for temporal logic falsification. S-TaLiRo searches for counterexamples to *Metric Temporal Logic* (MTL) properties for non-linear hybrid systems through global minimization of a *robustness metric* [3]. The global optimization is carried out using stochastic optimization techniques that perform a random walk over the initial states, controls and disturbances of the system [5, 1].

S-TaLiRo supports systems implemented as Simulink/Stateflow (TM) models as well as general *m-functions* in Matlab. Other frameworks can be supported readily, provided a Matlab (TM) interface is made available to their simulators. S-TaLiRo has been designed to be used by developers with some basic awareness of temporal logic specifications. Simulink/Stateflow (TM) models are the *de-facto* standard amongst developers of control software in many domains such as automotive control and avionics. S-TaLiRo also supports the easy input of

* This work was partially supported by a grant from the NSF Industry/University Cooperative Research Center (I/UCRC) on Embedded Systems at Arizona State University and NSF awards CNS-1017074 and CNS-1016994.

[1] S-TaLiRo web-page: https://sites.google.com/a/asu.edu/s-taliro/. The public release of the toolbox does not contained the parts of the software that were developed under I/UCRC funding.

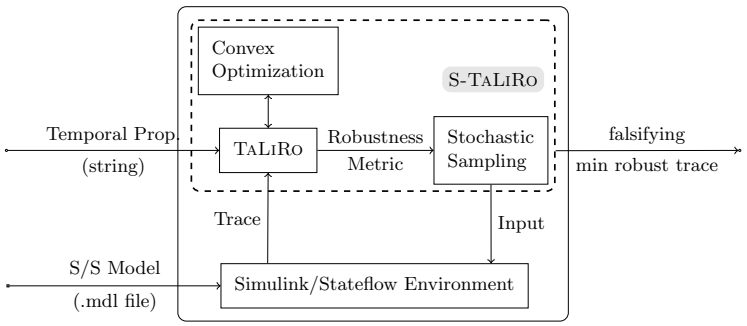


Fig. 1. The architecture of the S-TaLiRo tool.

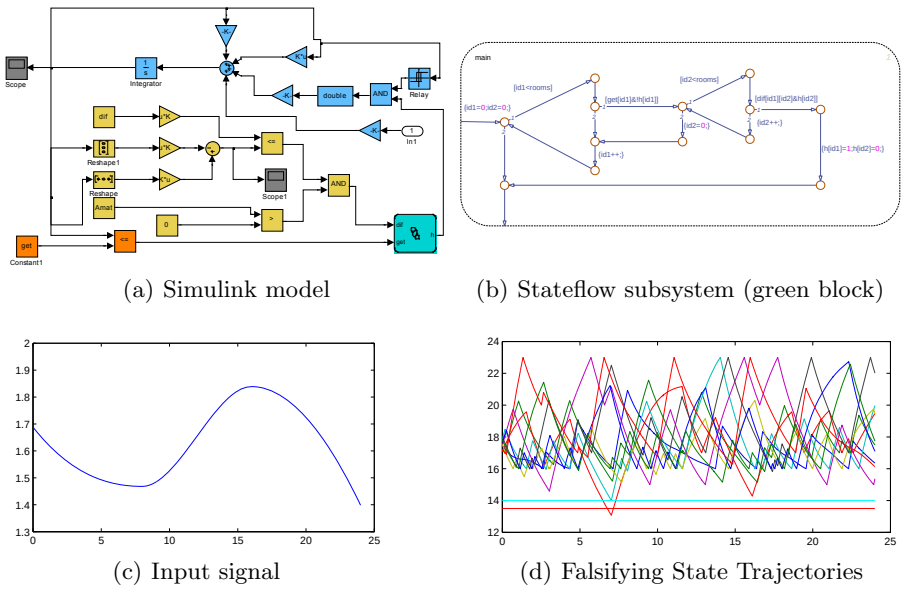
metric temporal logic formulae through an in-built parser. It has been designed and packaged as a Matlab toolbox with a simple command line interface.

At its core, S-TaLiRo integrates robustness computation for traces of hybrid systems (TaLiRo) [3] with stochastic simulation [8]. The search results in simulation traces with smallest robustness values found. In practice, traces with negative robustness are falsifications of temporal logic properties. Alternatively, traces with lower robustness values are closer in distance to falsifying traces using a mathematically well defined notion of distance between trajectories and temporal logic properties. Such traces may provide valuable insight to the developer on why a given property holds or how to refocus a failed search for a counter-example.

S-TaLiRo is based on recent progress in robustness metrics for metric temporal logic properties of continuous systems [3]. These metrics were extended in [5] to hybrid traces that combine continuous state evolution with discrete switches. The application of Monte-Carlo techniques that use sampling biased by robustness for falsification is described in our HSCC 2010 paper [5]. In [1], we report on our experience with other optimization techniques including *Ant-Colony Optimization*. S-TaLiRo also contains an optimized implementation of the robustness metrics (TaLiRo) along with the ability to plug-in other stochastic optimization algorithms.

2 The S-TaLiRo Tool

Figure 1 shows the overall architecture of our toolbox. The toolbox consists of a temporal logic robustness analysis engine (TaLiRo) that is coupled with a stochastic sampler. The sampler suggests input signals to the Simulink/Stateflow (TM) simulator which returns an execution trace after the simulation. The trace is then analyzed by the robustness analyzer which returns a robustness value. The robustness is computed based on the results of convex optimization problems used to compute signed distances. In turn, the robustness score computed is used by the stochastic sampler to decide on a next input to analyze.



(a) Simulink model (b) Stateflow subsystem (green block)

(c) Input signal (d) Falsifying State Trajectories

Fig. 2. Room heating benchmark HEAT30 and results obtained from S-TaLiRo run.

If in this process, a falsifying trace is found, it is returned to the user, who can then proceed to examine it inside the Simulink/Stateflow modeling environment. If the process times out, then the least robust trace found by the tool is output for user examination.

3 Usage

S-TaLiRo has been designed to be seamlessly integrated in the model based design process of Matlab/Simulink (TM). The user designs the model in the Simulink/Stateflow (TM) environment as before. At present, the only requirement is that input signals must be provided to the Simulink model through input ports. Then S-TaLiRo is executed with the name of the Simulink model as a parameter along with the set of initial conditions, the constraints on the input signals (if any) and the MTL specification. Currently, the user may select one of the two available stochastic optimization algorithms: Monte Carlo or Ant Colony Optimization. However, the architecture of S-TaLiRo is modular and, thus, any other stochastic optimization method can be readily implemented.

As a demonstration, we apply S-TaLiRo to the room heating benchmark from [4] (see Fig. 2). We chose the benchmark instance HEAT30. This is a hybrid system with 10 continuous variables (10 rooms) and 3360 discrete locations ($\binom{10}{4}2^4$ where 4 is the number of the heaters). The set of initial conditions is $[17, 18]^{10}$ and input signal u can range in $[1, 2]$. The goal is to verify that no room temperature drops below [14.50 14.50 13.50 14.00 13.00 14.00 14.00 13.00 13.50

$14.00]^T$. The input signal was parameterized using a piecewise cubic Hermite interpolating polynomial with 4 control points evenly distributed in the simulation time. S-TaLiRo found a falsifying trace with robustness value of -0.429. Figure 2 shows the trace and the input signal discovered by S-TaLiRo.

4 Related Work

The problem of testing hybrid systems has been investigated by many others (see the related research section in [5]). Most of the research focuses on *parameter estimation* [7, 2]. Recently, however, the problem of temporal falsification for hybrid systems has received a lot of attention [6, 5]. Unfortunately, the publicly available tool support has been fairly low in this space. The only other publicly available toolbox that supports computation of robustness for temporal logic formulas with respect to real-valued signals is BREACH [2]. However, BREACH currently does not support temporal logic falsification for arbitrary Simulink/Stateflow models. Along the lines of commercial products, Mathworks provides a number of tools such as SystemTest[2] (TM) and Simulink Design Verifier[3] (TM). S-TaLiRo does not attempt to be a comprehensive test tool suite as the above, but rather to solve a very targeted problem, i.e., the temporal logic falsification for hybrid systems. In the future, we hope to extend S-TaLiRo and the theory of robustness to estimate properties such as worst-case timings and integrate it into the statistical model checking framework.

References

1. Y. S. R. Annapureddy and G. Fainekos. Ant colonies for temporal logic falsification of hybrid systems. In *Proceedings of the 36th Annual Conference of IEEE Industrial Electronics*, 2010. (To Appear).
2. Alexandre Donze. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification*, volume 6174 of *LNCS*, pages 167–170. Springer, 2010.
3. Georgios Fainekos and George Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.
4. Ansgar Fehnker and Franjo Ivanĉić. Benchmarks for hybrid systems verification. In *HSCC*, volume 2993 of *LNCS*, pages 326–341. springer, 2004.
5. Truong Nghiem, Sriram Sankaranarayanan, Georgios Fainekos, Franjo Ivančić, Aarti Gupta, and George Pappas. Monte-Carlo techniques for the falsification of temporal properties of non-linear systems. In *Hybrid Systems: Computation and Control (HSCC'10)*, pages 211–220. ACM Press, 2010.
6. E. Plaku, Lydia E. Kavraki, and Moshe Y. Vardi. Falsification of ltl safety properties in hybrid systems. In *TACAS*, volume 5505 of *LNCS*, pages 368 – 382, 2009.
7. A. Rizk, G. Batt, F. Fages, and S. Soliman. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *6th International Conference on Computational Methods in Systems Biology*, number 5307 in LNCS, pages 251–268. Springer, 2008.

[2] `http://www.mathworks.com/products/systemtest/`

[3] `http://www.mathworks.com/products/sldesignverifier/`

8. Reuven Y. Rubinstein and Dirk P. Kroese. *Simulation and the Monte Carlo Method*. Wiley Series in Probability and Mathematical Statistics, 2008.

A S-TaLiRo presentation

The presentation of S-TaLiRo will be divided into three parts which correspond to the major parts of the S-TaLiRo architecture (see Fig. 1). First, a quick introduction to the Matlab/Simulink/Stateflow (TM) will be provided for the members of the audience who might not be familiar with the popular scientific computing platform. Second, we will introduce TaLiRo, which is a toolbox for the computation of the robustness of signals with respect to MTL formulas. Finally, we will present S-TaLiRo through a couple of test cases.

The presentation of S-TaLiRo will be performed using demos and slides. For the demos, we will require Internet connection in order to get access to a Matlab license on our laptop.

A.1 Matlab/Simulink/Stateflow (TM)

The presentation of Matlab/Simulink/Stateflow (TM) will be quite brief and mainly focus on the command interface of Matlab (TM) and on the model based design of hybrid systems using Simulink/Stateflow (TM). The presentation of the script language is necessary in order to demonstrate later on the ease of use of S-TaLiRo.

The concept of model based design and some of the basic building blocks will be introduced using the simple heating model in Fig. 3. This demo is based on the heating model of a house that is provided with Simulink. We assume that the resident may set arbitrarily the thermostat every 6 hours to some temperature in the range $[70, 84]^\circ F$. We also assume that the variation in the ambient temperature can be modeled using Piecewise Cubic Hermite Interpolating Polynomials with control points in the range $[50, 80]^\circ F$ and that the initial temperature in the house is somewhere in the range $[20, 40]^\circ C$. The goal is to find a system behavior where the daily Cost of the heating might exceed 10 or if the difference between the thermostat setting and the indoor temperature is below $-4.5^\circ F$, then the temperature difference is raised above $-4.5^\circ F$ and stays so for 15 min within 30 min.

A.2 TaLiRo

TaLiRo is a software toolbox for the analysis of the robustness of MTL formulas with respect to discretized real-valued signals. The theoretical foundations of TaLiRo are presented in [3].

The current version of TaLiRo can be downloaded from

`https://sites.google.com/a/asu.edu/s-taliro/`

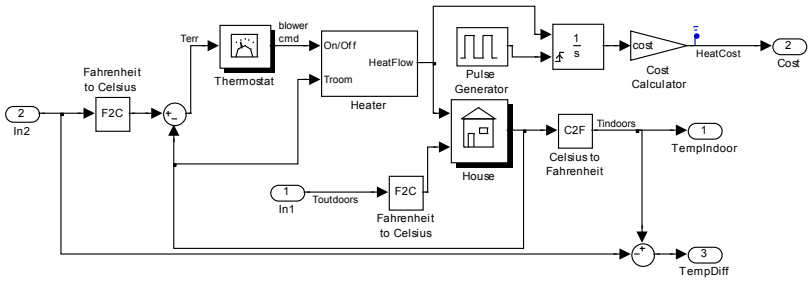


Fig. 3. A simple heating model for a house. This example is a slightly modified Simulink demo.

Version 1.0 is now available as a Matlab toolbox and it can handle multi-dimensional real-valued signals. Furthermore, we have improved the computation of MTL robustness by implementing a dynamic programming based algorithm as opposed to the previous version which was based on formula rewriting techniques. Now the computation time for the robustness is linear in the length of the simulation trace and the size of the formula.

$\neg$	$\vee$	$\wedge$	$\rightarrow$	$\leftrightarrow$	
!	\/	/\	->	<->	
$\bigcirc$	$\mathcal{W}$	$\Box$	$\Diamond$	$\mathcal{U}$	$\mathcal{R}$
X	W	[]	<>	U	R

Table 1. Correspondence between logical operators and ASCII symbols.

The presentation of TaLiRo will start by giving the basic definitions of the temporal logic operators and the theoretical background on temporal logic robustness. Then, we will present the syntax of the logic (Table 1) and how we can define MTL formulas within the Matlab environment.

Figure 4 presents a script for defining an MTL formula, the mapping of the predicates and, finally, calling TaLiRo. In lines 01-08, we define the mappings of the atomic propositions to subsets of the continuous observation space. Essentially, we have the mapping of C and DT to the sets $\mathbb{R} \times (\infty, 10] \times \mathbb{R}$ and $\mathbb{R}^2 \times [-4.5, +\infty)$. The atomic propositions can be mapped to arbitrary polyhedral sets of the form $\{x \mid Ax \leq b\}$. In line 10, we define the MTL formula $\Box(C \wedge (\neg DT \rightarrow \Diamond_{[0,0.5]}\Box_{[0.25]}DT))$, which states that the daily Cost should be always less then 10 and that if the difference between the thermostat setting and the indoor temperature drops below $-4.5°F$, then it should be raised above $-4.5°F$ within 30 min and stays so for 15 min. In line 12, we call TaLiRo on a multi-dimensional signal x and a sequence of time stamps t. TaLiRo returns the Boolean satisfiability value of the formula with respect to the signal along with the corresponding robustness.

The output of TaLiRo on a typical ambient temperature - thermostat setting scenario displayed in Fig. 5 is `[1,0.068333]`. That is, the output signals satisfy the specification with robustness `0.068333`.

```
01. % Cost
02. pred(1).str='C';
03. pred(1).A = [0 1 0];
04. pred(1).b = 10;
05. % Temperature
06. pred(2).str='T';
07. pred(2).A = [0 0 -1];
08. pred(2).b = -4.5;
09. % MTL formula
10. phi='[](C /\ (!T -> <>_[0,0.5] []_[0,0.25]T))';
11. % Calling TaLiRo
12. [B,R] = taliro(phi,pred,x,t);
```

Fig. 4. A Matlab script for calling TaLiRo on state trajectory `x` with time stamps `t`.

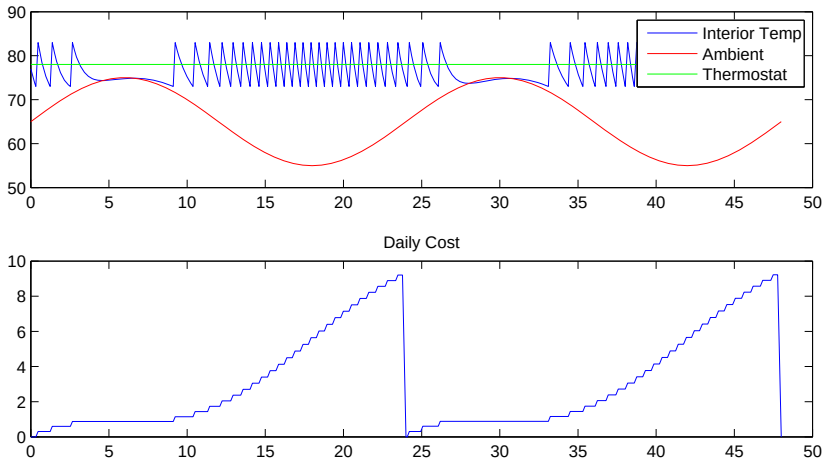


Fig. 5. The output of the model of Fig. 3 on a typical weather - thermostat setting scenario. The x-axis represents hours.

A.3 S-TaLiRo

Next, we will present how S-TaLiRo can be applied to Simulink models. The specifications can be defined either over the state space of the model or over

the output space. This is an explicit option provided to S-TaLiRo. If the model does not have external inputs, then the search for a falsifying trajectory can only be performed over the set of initial conditions. If on the other hand, we would like to verify the MTL property over a Simulink model with external inputs, then these inputs must be defined as input ports to the model (see Fig. 3).

No matter what is the type of the model we provide to S-TaLiRo, we need to also specify the set of initial conditions as well as the constraints to the inputs. In the current version, both must be provided as hypercubes. Note that if we do not want to search for a falsifying trajectory over the set of initial conditions or the system does not have any inputs, then we can set either variable as an empty array. If the system does accept input signals, then in this case we need to parameterize the input function space using a finite set of points in time. For that reason, the user provides two more parameters: the type of the interpolating function and the number of control points in time for each input signal. Currently, S-TaLiRo supports all the interpolating functions provided in Matlab using the `interp1` function as well as well as piecewise constant signals.

The interface of S-TaLiRo is as follows

```
[rob,rtime,nIter,samples] =
s_taliro(model,icond,irange,cparray,phi,pred,tt,opt);
```

The inputs are:

- `model`: a string with the name of the Simulink model, a pointer function or an object of the hybrid automata class
- `icond`: a hypercube defining the set of initial conditions
- `irange`: a hypercube defining the set of constraints on the inputs
- `cparray`: the number of control points for each input signal
- `phi`: a string with the MTL formula
- `pred`: a structure with the atomic proposition mapping
- `tt`: the total simulation time
- `opt`: various S-TaLiRo options

and the outputs are

- `rob`: the minimum robustness value found at each run of the falsification algorithm
- `rtime`: the total running time until a falsifying trajectory is found or the total number of stochastic tests is reached
- `nIter`: the total number of iterations until a falsifying trajectory is found or the total number of stochastic tests is reached
- `samples`: the initial state vector and input control points that produced the falsifying trajectory

The following Matlab script presents how easy is to call S-TaLiRo on the model in Fig. 3.

Example 1 (Using S-TaLiRo *on the house heating example).* The MTL formula and the atomic propositions are defined as in Fig. 4. The following script calls S-TaLiRo with the default options. Namely, the parameterization of the input space is performed using Piecewise Cubic Hermite Interpolating Polynomials and optimization algorithm is ACO [1].

```
% The name of the Simulink model
model = 'househeat01';
% Total simulation time in hours
tt = 48;
% Initial Indoor Temperature and Cost ranges
icond = [20 40; 0 0];
% Control points for Ambient Temp and Thermostat value ranges
irange = [50 80; 70 84];
% Number of control points for each input signal
cparray = [4 8];
[rob,rtime,nIter,samples] =
   s_taliro(model,icond,irange,cparray,phi,pred,tt);
```

If we need to change some of the parameters of S-TaLiRo, then we can use the options class `staliro_options`. For example, if we would like to use the Monte Carlo solver and to have the thermostat input defined as piece-wise constant function, then we simple type

```
opt = set(staliro_options,'optimization_solver','MonteCarlo', ...
   'interpolationtype', {'pchip';'pconst'});
[rob,rtime,nIter,samples] =
   s_taliro(model,icond,irange,cparray,phi,pred,tt,opt);
```

The falsifying trajectories can be reproduced and plotted as follows:

```
[T,XT,YT,IT] = SimSimulinkMdl(model,[size(icond,1) cparray],...
   samples(2,:),tt,{'pchip';'pconst'});
subplot(1,2,1)
plot(T2,YT2(:,1),IT2(:,1),IT2(:,2),'r',IT2(:,1),IT2(:,3),'g')
legend('Interior Temp','Ambient','Thermostat')
subplot(1,2,2)
plot(T2,YT2(:,2))
title('Daily Cost')
```

The above script produces the graphs in Fig. 6 which present the falsifying scenario.

Time permitting, we will present the use of TaLiRo on models defined by *m-functions* in Matlab. Here is an example of a model for aircraft dynamics written as a M-Function.

Example 2 (Aircraft ODE model).

```
function [ret]=aircraftODE(T,X,u)

B0 = 0.07351;
B1 = -1.5E-3;
```

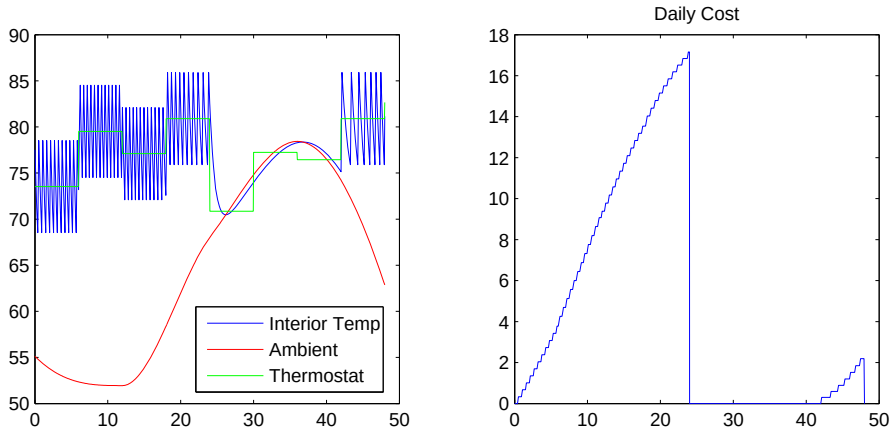


Fig. 6. A falsifying scenario found by S-TaLiRo with robustness −7.156.

```
B2 = 6.1E-4;
C0 = 0.1667;
C1 = 0.109;
m = 74E+3;
g = 9.81;
S = 158;
rho = 0.3804;

mat1 = [(-S*rho*B0*X(1,1)*X(1,1))/(2*m)-g*sin(pi*X(2,1)/180 );
   (S*rho*C0*X(1,1))/(2*m)-g*cos(pi*X(2,1)/180)/X(1,1);
   X(1,1)*sin(pi*X(2,1)/180)];
mat2 = [u(1,1)/m; 0; 0];
mat3 = [(-S*rho*X(1,1)*X(1,1))/(2*m)*(B1*u(2,1)+B2*u(2,1)*u(2,1));
   (S*rho*C1)/(2*m)*X(1,1)*u(2,1);
   0];
ret = mat1+mat2+mat3;
end
```

Hybrid automata can also be defined using a class for hybrid automata that we provide. However, this is not going to be part of the presentation.