

Tool Library Workshop

Marcus Lucas and Paulo Tabuada

Cyber-Physical Systems Laboratory
Department of Electrical and Computer Engineering
University of California at Los Angeles



Background

- Algorithm verification is vital to the development of cyber-physical systems
 - Safety Critical
 - Expensive to test
 - Difficult to reason about
- Verification tools are generally good for the research community
 - Aids reproducability of results
 - Facilitates benchmarking
 - Improves quality of research/knowledge dissemination

Issues

- But verification can be a lot of extra work
 - Finding, understanding, and then using tools is non-trivial
- System designers need to better understand verification and what types of problems can be verified
- Low level of knowledge and confidence regarding existing tools
- Design process is not understood in terms of verification

Goals

- Embed verification into the engineering design process as a first-order objective
- Communicate the circumstances under which verification should be applied
- Normalize the publication and support of tools created by community members
- Grow confidence in verification's utility

How to Get There?

- Reduce barriers to entry through the CPS-VO
 - Tool Repository
 - Tutorials
 - Integrated tools
- Encourage student engagement
- Create benchmarks and standard models for evaluating tools

Verification Tool Library

Verification Tool Library

CPS-VO » CPS TOOLS AND DESIGN STUDIOS » VERIFICATION TOOL LIBRARY » SEARCH FOR TOOLS

Search for Tools

Home

README

Search for Tools →

Browse Tool Library

Members

Tutorials

Tutorial - CORA (Hidden)

Tutorial - SpaceEx

Tutorial - S-TALIRO (Hidden)

Bulk Edit (Hidden)

Taxonomic Information

Forums

Files

COLLABORATE →

EDIT GROUP | TRACK | TAXONOMY | BROADCAST | PANELS | GROUP STATS

Search results for Verification Tool Library

Search for: Types of systems handled

<Any>

Techniques used

- Abstractions
- Agent and Mode Architecture
- Ant Colony Optimization
- Approximate (bi)simulation
- Approximation of solutions to Hamilton-Jacobi partial differential equations
- Bisimulation minimization
- Boolean Equation Systems
- Bounded Model Checking
- Branching-Time Temporal Logics

Type of verification

- Bisimulation generation
- Cost optimal reachability analysis
- Description/modeling language
- Falsification
- Invariant analysis
- Invariant set computation
- Model checking
- Modeling
- Parameter synthesis

Interface

- C++
- Integrated to SpaceEx
- Java
- Matlab
- Model editor integrated
- OMNeT++
- Own interface
- Own language
- Python

Termination guarantees

- Guaranteed to terminate
- Not guaranteed to terminate

OS Free

- Linux
- Mac OS X
- Solaris
- Windows

Apply | Reset | WIKIPAGE

Submitted by [nanli](#) on Tue, 09/25/2018 - 2:53pm

GTTS

Automotive Testing Simulation Validation and Verification Autonomous Driving Simulator test generation verification and validation Tool Free Java Linux Mac OS X Solaris Windows Simulation



Verification Tool Library

- VTL documents mature tools available to the CPS community
- Searchable by taxonomic terms and includes a wiki
- Links to tutorials, integrated tools, more thorough documentation

Verification Tool Library

Repository Demo

Verification Tool Library

Links

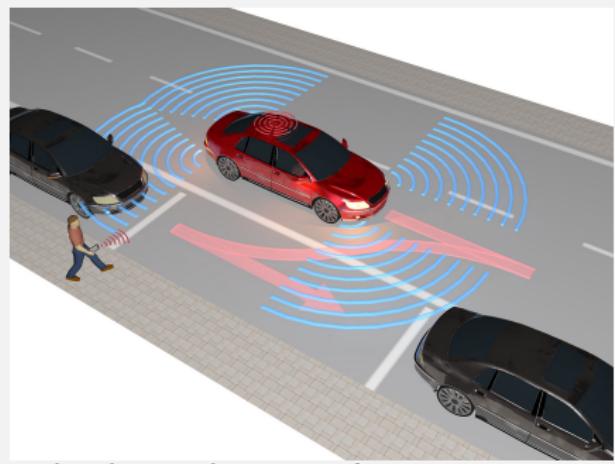
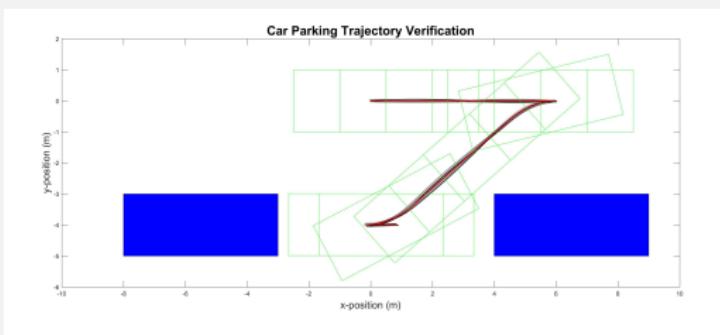
- Tool Library: https://cps-vo.org/group/verification_tools/
- Submission Form: <https://cps-vo.org/group/tools/submit>

Tutorials

- SpaceEx
 - Only handles hybrid systems with affine continuous dynamics
 - Particularly accessible to high-school & undergraduate students
- CORA
 - Formal verification of non-linear systems
- S-TALIRO
 - Falsification of logical constraints

Tutorials

Autonomous Parking Example



- Finding and following parking trajectories is a basic requirement of autonomous cars

Tutorials

Autonomous Parking Example

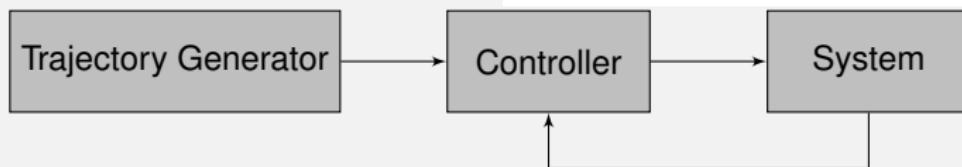
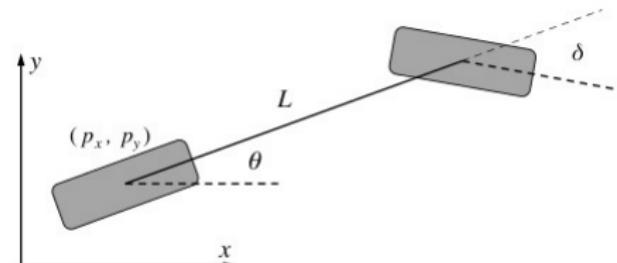
Kinematic Bicycle Model of Car

$$\dot{x} = v \cos(\theta)$$

$$\dot{y} = v \sin(\theta)$$

$$\dot{\theta} = \frac{v}{L} \sin(\delta)$$

$$\dot{\delta} = u$$

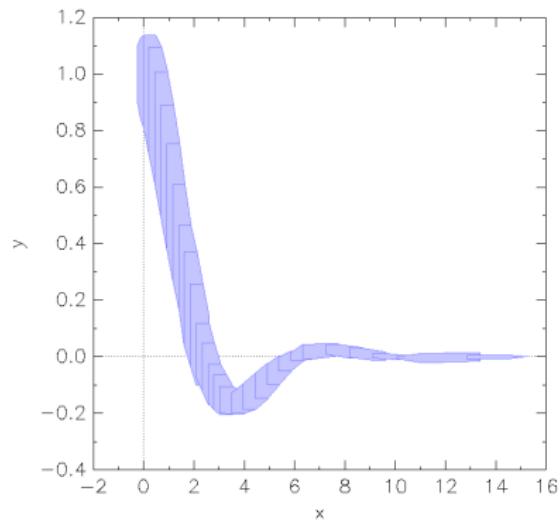
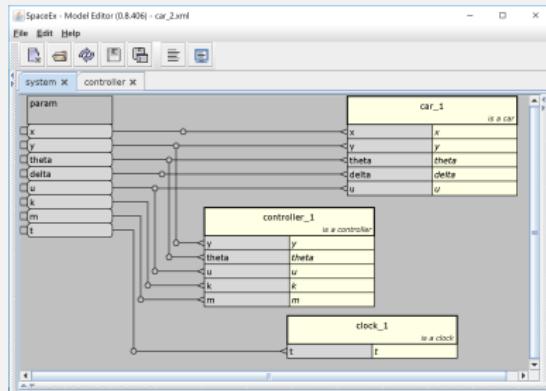


- What if you have a valid trajectory, and want to test if your controller tracks it properly?

Tutorials

SpaceEx

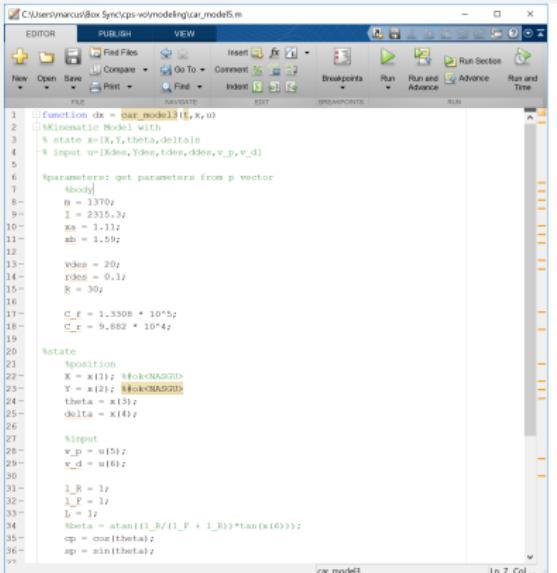
- SpaceEx can be used to evaluate simple control problems such as stabilizing to a horizontal trajectory.



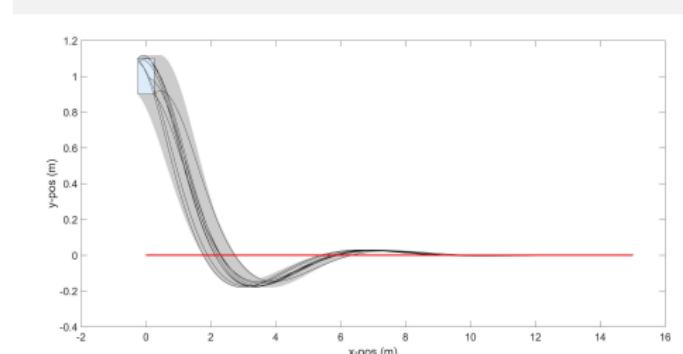
Tutorials

CORA

■ CORA can do the same thing



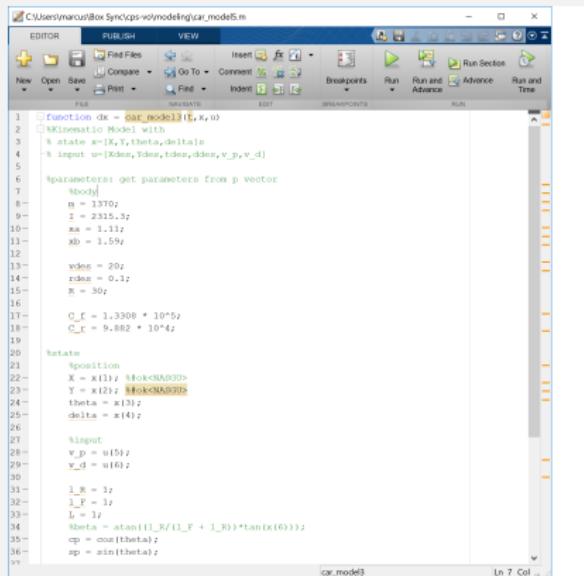
```
C:\Users\marcus\Box Sync\cps-vo\modelling\car_model5.m
EDITOR PUBLISH VIEW
FILE NAVIGATE EDIT BREAKPOINTS RUN
New Open Save Print Insert Go To Comment Breakpoints Run Run and Advance Run and Time
1 %function ds = car_model3(t,x,u)
2 %Kinematic Model with
3 % state x=[x,y,theta,delta]
4 % input u=[delta, ydes, tdes, ddes, v_p, v_d]
5
6 %parameters: get parameters from p vector
7 %load p
8 %    D = 1370;
9 %    I = 2335.3;
10 %    x0 = 1.13;
11 %    xb = 1.59;
12
13 %    vdes = 20;
14 %    rdes = 0.1;
15 %    g = 30;
16
17 %    C_E = 1.3308 * 10^5;
18 %    C_F = 9.882 * 10^4;
19
20 %state
21 %position
22 %    X = x(1); %okOK
23 %    Y = x(2); %okOK
24 %    theta = x(3);
25 %    delta = x(4);
26
27 %input
28 %    v_p = u(5);
29 %    v_d = u(6);
30
31 %    l_R = 1;
32 %    l_F = 1;
33 %    l = l;
34 %    theta = atan((l_R/(l_F + l_R))*tan(x(3)));
35 %    cp = cos(theta);
36 %    sp = sin(theta);
37
```



Tutorials

CORA

- But it is also capable of analyzing non-linear systems, making it much more useful for verifying systems like the full parking controller.



CV:\Users\marcus\Box Sync\cps-vo\modelling\car_model5.m

EDITOR PUBLISH VIEW

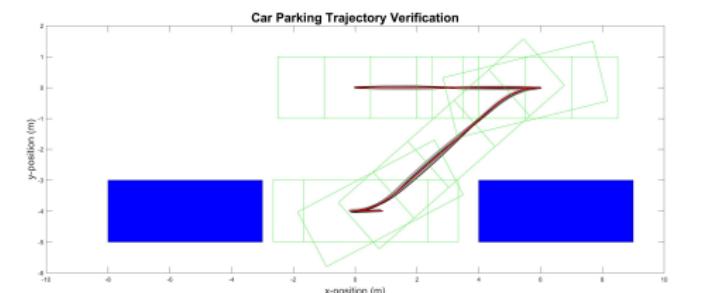
New Open Save Print Comment Insert Go To Breakpoints Run Run and Advance Run and Advance Run Time

FILE NAVIGATE EDIT

```
1 function dx = car_model3(t,x,u)
2 %Kinematic Model with
3 % state: x[X,Y,theta,delta]
4 % input: u[delta_dot,tdot,ddot,v_p,v_d]
5
6 %parameters: get parameters from p vector
7 %Model
8 % n = 1370;
9 % I = 2315.3;
10 % ka = 1.112;
11 % kb = 1.594;
12
13 %vdes = 20;
14 %rdash = 0.1;
15 %B = 30;
16
17 %C_f = 1.3308 * 10^5;
18 %C_r = 9.882 * 10^4;
19
20 %state
21 %position
22 % X = x(1);
23 % Y = x(2);
24 %theta = x(3);
25 %delta = x(4);
26
27 %input
28 % v_p = u(1);
29 % v_d = u(2);
30
31 % l_R = 3;
32 % l_P = 3;
33 % L;
34 %theta = atan((l_R/(l_F + l_R))*tan(x(3)));
35 %cp = cos(theta);
36 %sp = sin(theta);
```

car_model3

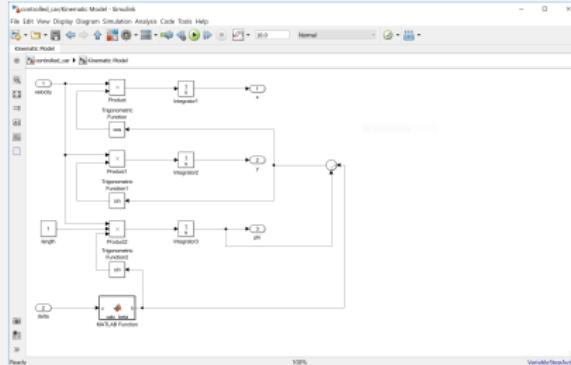
Ln 7 Col 1



Tutorials

S-TALIRO

- S-TALIRO does not provide formal guarantees, but can be used to falsify the same classes of problems



```
C:\Users\marcus\MATLAB\Toolbox\staliro\demos\Falsification\demostaliro_demo_automtrans_01.m
EDITOR PUBLISH VIEW
New Open Save File Files Compare Go To EDIT Breakpoints
File Print Find Breakpoints
Run Run and Advance Run and Advance Run and Time
22
23 disp(' ')
24 disp('The constraints on the initial conditions defined as a hypercube')
25 init_cond = []
26
27 disp(' ')
28 disp('The constraints on the input signal defined as a range')
29 input_range = [0 100]
30 disp(' ')
31 disp('The number of control points for the input signal')
32 cp_array = 7
33
34 disp(' ')
35 disp('The specification')
36 phi = '1<=x1 /\ \>x2'
37
38 ii = 1z
39 predo(ii).x1e = 'x1>1'
40 predo(ii).A = [-1 0]
41 predo(ii).b = -120
42 predo(ii).lo = [1;7]
43
44 ii = ii+1z
45 predo(ii).x1e = 'x1>1'
46 predo(ii).A = [0 -1]
47 predo(ii).b = -4500z
48 predo(ii).lo = [1;7]
49
50 disp(' ')
51 disp('Total Simulation time')
52 time = 30
53
54 disp(' ')
55 disp('Create an staliro_options object with the default options')
56 opt = staliro_options()
57
```



Embedded Tools

- Currently have **SpaceEx** running

The screenshot shows the SpaceEx software interface running on a virtual machine. The main window has a title bar "SpaceEx [Virtual Machine Server]" and a toolbar with "Home", "About SpaceEx", "Documentation", "Run SpaceEx", "Downloads", and "Contact".

Console:

```
192.168.1.84/ YouTube 192.168.1.84
Model Specification Options Output Advanced
System system Update
system
  Controlled : t, x, y, theta, delta, u
  Base-components : clock_1, car_1, controller_1
Initial states
t=0 & x=0 & 0.8<y<=1.2 & -0.1<theta<=1 & delta=0 & u=0
Computing reachable states...
Iteration 0... 1 sym states passed, 0 waiting 0.01s
Found fixpoint after 1 iterations.
Computing reachable states done after 0.011s
Output of reachable states... 0.048s
0.10s elapsed
1724KB memory
SpaceEx output file : output_t-x (gen),
SpaceEx output file : output_t-y (gen),
SpaceEx output file : output_x-y (gen),
Graph output file : plot_t-x (qf),
Graph output file : plot_t-y (qf),
Graph output file : plot_x-y (qf).
```

Reports:

Graphics:

- Plot 1: A line graph of x vs t. The x-axis ranges from 0 to 16, and the y-axis ranges from 0 to 16. The plot shows a single straight line starting at (0,0) and ending at approximately (15, 15).
- Plot 2: A line graph of y vs t. The x-axis ranges from 0 to 16, and the y-axis ranges from -0.2 to 1.4. The plot shows a blue line starting at (0, 1.2), dropping to a minimum of about -0.1 at t=4, and then oscillating between 0 and 0.5 for the rest of the time.
- Plot 3: A line graph of x vs y. The x-axis ranges from 0 to 16, and the y-axis ranges from -0.2 to 1.4. The plot shows a blue line starting at (0, 1.2), dropping to a minimum of about -0.1 at y=0.5, and then oscillating between 0 and 0.5 for the rest of the time.



Running Tools on the CPS-VO

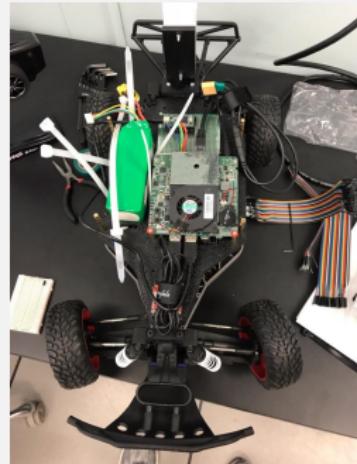
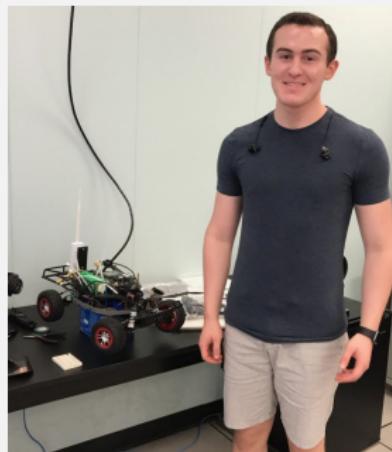
SpaceEx Example

SpaceEx Demo



Student Engagement

- Want to normalize the use of verification tools in design process
- Targeting undergraduate and even high-school students
- Ideal tools integrate easily with familiar software (i.e. Matlab/Simulink)



Future Work

Benchmarks

- We're announcing a new benchmarking competition!
- Expansion of the "Friendly Competition" held during the ARCH Workshop
 - <https://cps-vo.org/group/ARCH>
- Tentative structure:
 - Participants integrate verification/synthesis tools into VO
 - Test period of about a month
 - Competition organizers evaluate tools against benchmarks using VO resources
- Competition results will be advertised through the CPS-VO