CHUCHU FAN, BOLUN QI, PARASARA S. DUG-
GIRALA, SAYAN MITRA, MAHESH VISWANATHAN

C2E2 USER'S GUIDE

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Compiled on January 23, 2016

Contents

1
Acknowledgements

Matthew Potok○ Zhenqi Huang ○ Taylor Johnson ○ Karthikeyan Manamcheri Sukumar ○ Le Wang ○ Department of Computer Science and Department of Electrical and Computer Engineering at the University of Illinois at Urbana Champaign ○ Coordinated Science Laboratory ○ The National Science Foundation ○ Air Force's Office of Scientific Research.

2
Introduction

C2E2 is a tool for verifying bounded-time invariant properties of Stateflow^{TM} models. It supports models with nonlinear dynamics, discrete transitions, and sets of initial states. The invariant properties have to be specified by conjunctions of linear inequalities. Internally, C2E2 implements the simulation-based verification algorithms described in the sequence of publications Fan and Mitra [2015], Duggirala et al. [2013, 2014], Sukumar and Mitra [2011]. The new version of C2E2 uses an on-the-fly discrepancy computation algorithm Fan and Mitra [2015] to automatically generate neighborhoods that conservatively contain all the behaviors of neighboring trajectories. In a nutshell, C2E2 parses and transforms the Stateflow^{TM} model to a mathematical representation, it generates faithful numerical simulations of this model using a validated numerical simulator, it then bloats these simulations using on-the-fly discrepancy computation to construct over-approximations of the bounded time reachable set, and it iteratively refines these over-approximations to prove the invariant or announce candidate counterexamples.

C2E2 has a GUI for loading and editing of Stateflow^{TM} models and properties, launching the verifier, and for plotting 2D sections of the reach set computed by the verifier. It saves the properties and the models in an internal HyXML format. The reach tubes computed for verification are stored in a machine-readable format.

3
Installation

We have tested the installation scripts on Linux/Ubuntu (ver 12, 64 bit). Currently we do not support any other operating systems.

Installing C2E2 should be straightforward.

- Download the latest distribution of C2E2 distribution from: `http://publish.illinois.edu/c2e2-tool/download/`.
- Unzip the files in a local directory, say `/C2E2/`.
- Go into `/C2E2/` and run `sudo ./installRequirements`. This should install all the packages needed and may take a while.
- If the above step succeeds then run `./installC2E2` to install the program.
- If the above step succeeds, then run `./testC2E2` and you should be able to pass all the test if the installation is successful.
- After passing the tests, run `./runC2E2` and C2E2's graphical user interface (GUI) should appear and you should be ready to load models.
- If any of the above steps fail then you can try to install the packaged separately. The list of all the needed packages are given in Appendix A.
- If any of the above steps fail, you have the second option to install the a VMware Workstation image of a 64 bit Ubuntu 14.04.10 which has the newest version of C2E2 already correctly installed. (password: c2e2admin)

4
Getting Started

In this chapter, we give a quick tour of some of the features of C2E2 using a couple of examples that are distributed as part of the package.

4.1 Opening a Model

Once C2E2 is launched, go ahead and open one of the example models from the `File` menu. For this tutorial, we will use a simple model of a adaptive cruise control (See Examples page) which is stored in the file `TotalMotion40s.hyxml`.

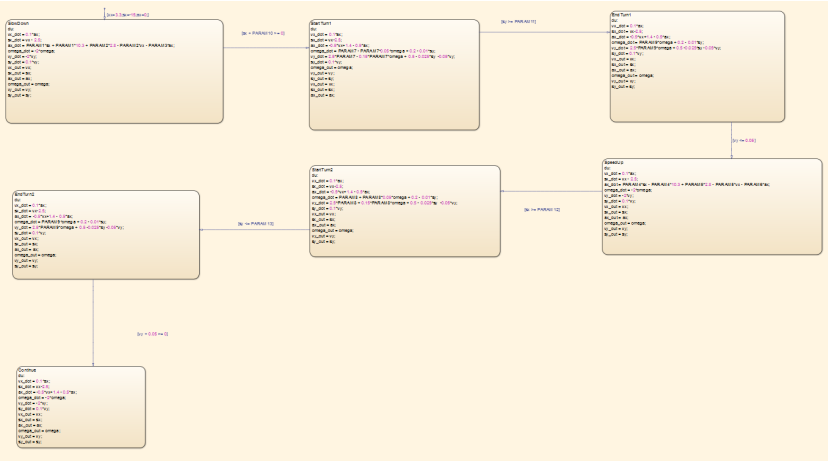

Figure 4.1: Stateflow model of a simple adaptive cruise control the 2D-plane.

The Stateflow™ model in this file is shown in Figure 4.1. In brief, the model has six continuous variables $v_x, s_x, a_x, v_y, s_y, omega$ and seven modes *SlowDown*, . . . , *Continue*, and in each mode the behavior of the adpative curise control is described by a linear differential equation with different inputs. There are six transitions, each going from one mode to another. The starting state is defined as $v_x = 3.3$

and $s_x = -15$ and $a_x = 0$ in *SlowDown*. You can generate numerical simulations of this model in Matlab. The initial state specified in the .mdl file is useful for generating simulations of the model in Matlab but it will be largely ignored by C2E2. You will have to specify the *set of* initial states in C2E2.

Upon opening the file the C2E2 window should look Figure 4.2.

The left hand side of this window shows the parse tree of the

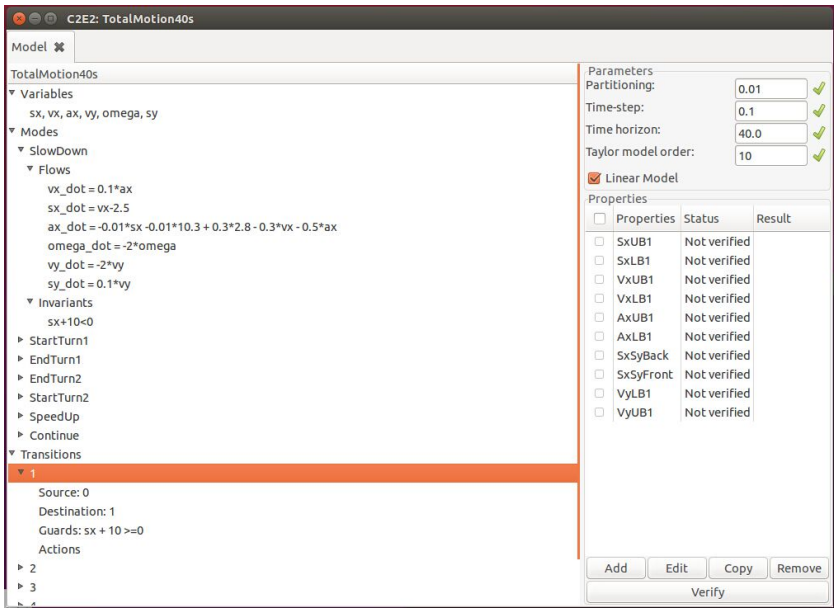

Figure 4.2: *Left:* Model parse tree. *Right:* Verification pane.

model and the right hand side is the *verification pane*. You can expand the tree to see the variables, transitions and modes of the automaton by clicking on the arrows left. In the near future, you will be able to edit the items in the parse tree. For now, this is a convenient representation of the model. Since we are using different discrepancy computation engines for linear systems and nonlinear systems, please note the following step for different models.

- Linear Models
  Please make sure you check the *Linear Model* box as shown in the red box in Figure 4.3 if the system is linear.

- Nonlinear Models
  Make sure you leave the *Linear Model* box as blank. If you are using old examples from the website, please make sure to delete the annotations from `.mdl` files, or change the value K in `.hyxml` to 2000. This is important since the wrong K value will influence the final result. You can leave *Gamma* value as it is.

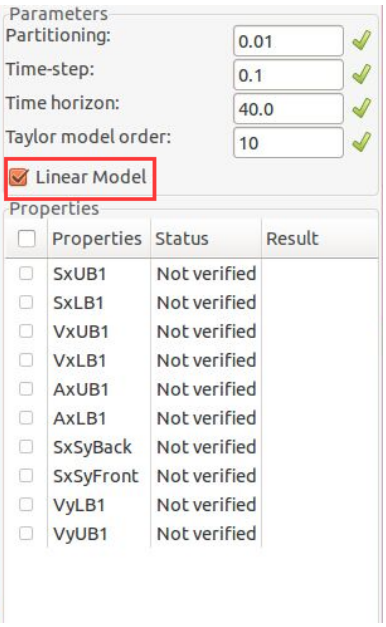

Figure 4.3: Linear model check box.

4.2 Properties

The right hand side of the main window is the verification pane. This is where you can add, edit, and copy properties and launch the verifier. Currently C2E2 verifies *bounded time linear invariant properties from linear bounded initial sets*. Such properties are specified by the time bound (T), the initial set and the unsafe set. The *Time horizon* parameter listed at the top of the verification pane is the time bound. Currently C2E2 requires both the initial and the unsafe sets to be described by a conjunction of linear inequalities involving the model variables. The model you load already has a couple of sample properties. Here we will walk you through the steps involved in creating a new property.

1. Click `Add` in the property pane. This launches the `Add Property` dialog box.
2. Enter a name for the property, say `SxUB1`, in the first textbox.
3. Enter a linear predicate on the variables to specify the *initial set* or the *starting states* in the second textbox. Currently, the syntax for specifying the initial state is as follows:
   ⟨ `mode-name` ⟩ : ⟨ (`linear-inequality` &&)+ ⟩.
   For example, for the above model:
   `SlowDown`: sx>=-15.0 && sx<=-14.95 && vx>=3.25 && vx<=3.3 && ax==0 && vy==0 && omega==0 && sy==0
   Is a valid expression for specifying the set of initial states.
4. Enter the unsafe set in the last textbox. Currently, the syntax for specifying the unsafe set is a &&-separated sequence of linear inequalities:
   sx>=50
5. Press `Add`.

If all the expressions are syntactically acceptable then there will be little green checks next to the textboxes and you will be able to add the property. Otherwise there will be a cross next to the textbox. **Both the unsafe set and the initial set should be described by a collection of linear inequalities and in addition the initial set should be bounded.**

Once the property is added the name of the property appears in the property pane. You may add several properties in the same way. You may also make copies of existing properties to save yourself some typing. The added properties can be saved with the model. See section 4.5.

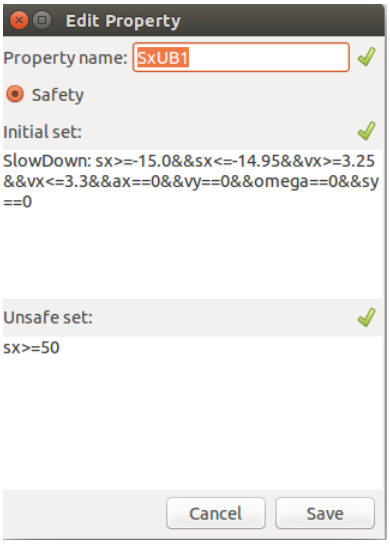

Figure 4.4: Dialog box for adding properties checks the syntax of the initial and unsafe sets.

4.3 Verifying

Once you have created a model and added a property (see Section 4.2) you can launch the verification engine by selecting the property and then clicking the `Verify` button.

C2E2 is sound which means that you can trust the Safe/Unsafe answer proclaimed by it. In principle, C2E2 is also complete for robust properties Duggirala et al. [2013]. That is, if the model satisfies the property robustly[1], and if the numerical precision supported by the algorithm is adequate then C2E2 should terminate with a Safe/Unsafe proclamation. In practice, the time it takes to verify is sensitive to the time horizon (T), the initial partition. You may want to first run the verification with small values of T.

[1] Robustness: the requirement that the actual reachable set of the model does not skim the boundary of the unsafe set.

The reachable set over-approximation computed by C2E2 is stored in the `/wd/`.

If you have multiple properties, then you may select one or more of them to be verified. Multiple properties are verified one at a time. When the verification is in progress clicking the `Abort` button aborts it.

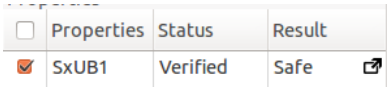

Figure 4.5: One or more properties can be selected by checking the boxes to the left of the property name. The `Verify` button launches the verification engine to verify one property at a time.

4.3.1 *Changing Verified Properties*

Once a property is verified the status of the property changes to `Verified`, the result `Safe/Unsafe` appears next to it, and a small box icon appears next to the result to launch the plotter (see Section 4.4). At this point, if you change any of the parameters associated with the property, say the Time horizon, then the status of the property changes to `Verified*`. This (*) indicates that the property and parameters verified is outdated.

4.4 Plotting

Once the verification of a property is complete, a small box icon appears next to the result. Click this icon and this opens a new tab with the same name as the property. This is the plotting window for this property: it enables us to plot various projections of the reach set that has been computed in verifying the property. The plot

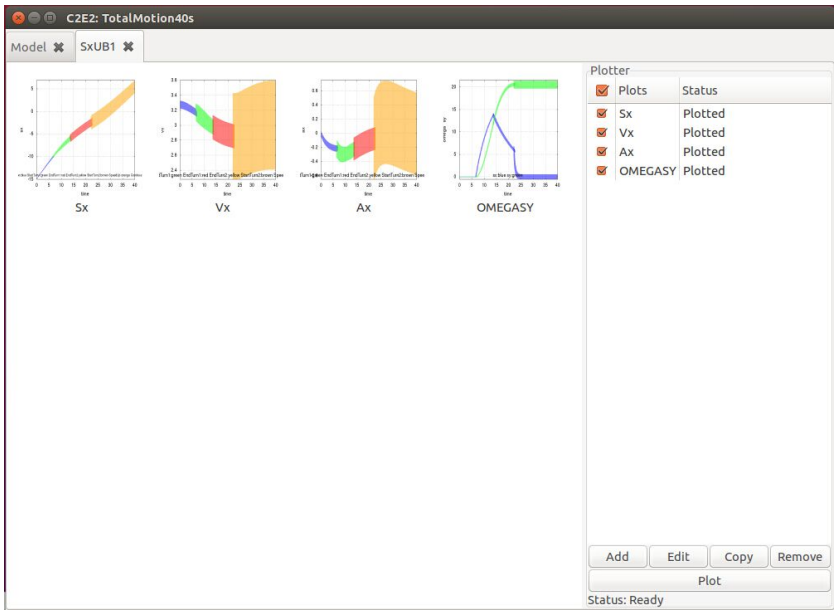

Figure 4.6: The plotting tab for property 1 without any plots.

window has two parts. The left pane shows all the plots icons and the right pane is used to create new plots. The steps for creating new plots is similar to that for creating properties. C2E2 can currently create two dimensional plots. As before, you can add more plots or edit/copy/remove/plots multiple plots. When you add/edit the plots, you will see a dialog box where you can select the variables would like to plot. You can plot a state variable with respect to time (for example, x vs. t) or two state variables (x vx. y). You can also plot multiple state variables with respect to time (x and y vs. t).

Once you are satisfied with your plots, you can click the plot button generate the plots.

As the program plots the reach sets, you will see icons appear on the left hand side of the window with a preview of what the plot looks like as well as the plot name below it. You can expand the plot by double clicking on these icons.

You can navigate the the first window and other opened plot windows by clicking on the tabs along the upper portion of the window. You can save the model as well as the properties you have

created in an `.hyxml` file.

Clicking on the `Plot` button will create the plots one by one and show the resulting icons. Double click on an icon to expand a plot. The plot window allows you to pan and zoom over the reach tube and also generate `.png` figures of the plot in wd/plotresult folder.

4.4.1 *Plots with multiple variables and modes*

If plotting a multiple variables with respect to time then the reach tubes of each variable is shown in a different color and the axes are labled with the corresponding colors. For example, in Figure 4.9 the x-axis is time and the y-axis shows the reach tubes of two variables ω and v_x which are shown in blue and green respectively, and the axis for v_x is labeled by green and that of ω is labeled by blue.

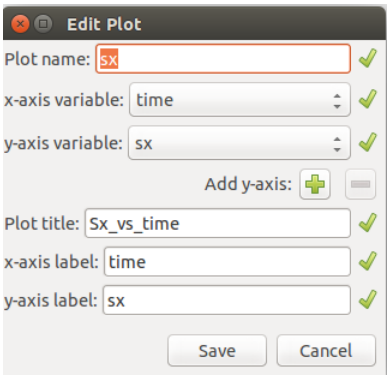

Figure 4.7: Add Plot dialog box.

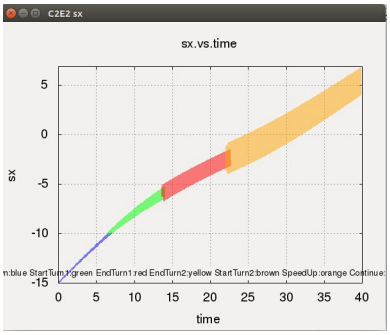

Figure 4.8: A plot of a reach tube of one variable with respect to time.

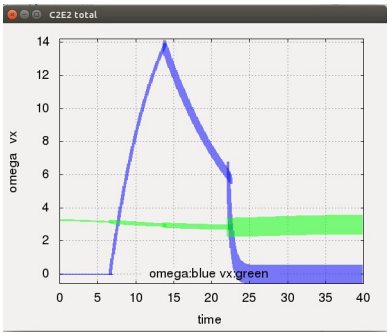

Figure 4.9: A plot of a reach tube of two variables with respect to time.

4.5 Loading and Saving

Currently C2E2 provides very basic functionality for loading and saving models and properties. You can save a model and its properties from the file menu. The saved file is in the `.hyxml` format as shown below Sukumar and Mitra [2011]. Once saved, a model and the properties in the `.hyxml` file can be loaded from the file menu.

Changes made to the model in the C2E2 fronted are **not** saved but only the edited properties are saved. However, you can edit the `.hyxml` file using a text editor to change the model and the properties. The reach sets computed during verification are stored in the working directory `/wd/` but currently they cannot be loaded.

A
Required Libraries

The following is a complete list of packages needed for installing C2E2.

1. GNU Linear Programming Kit along with Python bindings, GLPK and PyGLPK (http://www.gnu.org/software/glpk/) (http://tfinley.net/software/pyglpk/)
2. GNU parser generator, Bison (http://www.gnu.org/software/bison/)
3. The Fast Lexical Analyzer, Flex (http://flex.sourceforge.net/)
4. Python (http://www.python.org/)
5. Python parsing libraries, Python-PLY (http://code.google.com/p/ply/)
6. GTK libraries for Python (http://www.pygtk.org/)
7. Plotting libraries for Python, Matplotlib (http://matplotlib.org/)
8. Packing configurations library (http://www.freedesktop.org/wiki/Software/pkg-config/)
9. GNU Autoconf (http://www.gnu.org/software/autoconf/)
10. Python xml library, lxml (http://lxml.de/installation.html)
11. Parma Polyhedron Library (http://bugseng.com/products/ppl/)

If you get any errors while installing PyGLPK, please visit the following website:
`http://tfinley.net/software/pyglpk/building.html` and install it manually.

Bibliography

Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. Verification of annotated models from executions. In *EMSOFT*, pages 1–10, 2013.

Parasara Sridhar Duggirala, Le Wang, Sayan Mitra, Mahesh Viswanathan, and César Muñoz. Temporal precedence checking for switched models and its application to a parallel landing protocol. In *FM 2014: Formal Methods - 19th International Symposium, Singapore, May 12-16, 2014. Proceedings*, volume 8442 of *Lecture Notes in Computer Science*, pages 215–229. Springer, 2014. ISBN 978-3-319-06409-3.

Chuchu Fan and Sayan Mitra. Bounded verification with on-the-fly discrepancy computation. *13th International Symposium on Automated Technology for Verification and Analysis, AVTA'15, Shanghai, China*, 2015.

Karthik Manamcheri Sukumar and Sayan Mitra. A step towards verification and synthesis from simulink/stateflow models. In *Tools paper in Hybrid Systems: Computation and Control (HSCC 2011)*, 2011.