CHUCHU FAN, BOLUN QI, PARASARA S. DUGGIRALA, SAYAN MITRA, MAHESH VISWANATHAN

C2E2 USER'S GUIDE

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Compiled on May 12, 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 12.04, 64 bit).

- C2E2 is already installed in the virtual machine image for the Artifact Evaluation.
- We provide a sequence of examples in form of `.hyxml` files in the `Examples` folder. The user can modify the models (e.g. dynamics, invariant, guards etc.) easily in the `.hyxml` files. Check out the updates on our website[1] for further information to support Matlab Stateflow model (`.mdl`) files.
- **Disclaimer:** this artifact evaluation submission (the `.ova` file) is generated using VM Workstation Player and might have some issues with VirtualBox.

[1] https://publish.illinois.edu/c2e2-tool/

C2E2 can also be installed on Mac. Instead of using the installation scripts, the packages will have to be manually installed with the help of brew (`http://brew.sh/`) and pip. If you do not already have brew, install it as directed by the website.

- Using "brew install <package-name>", install python, gmp, ppl, pygtk, gnuplot, boost, swig.
- Using "pip install <package-name>", install psutil, sympy, lxml. If lxml fails install command line developer tools by running "xcode-select –install" and then try again.
- Run installGLPK. If you run into errors, refer to this guide.
- Install PyGLPK by using this guide with slight modifications. DO NOT install GLPK using home-brew and install ply version 3.4 by running "pip install ply==3.4".
- Install GnuPlot by downloading the zip here and running "python setup.py install".

- Finally run "mkdir -p /.local/share/". You should now have all the dependencies and the program should run smoothly.

Note: please do not use the old example files from the website since the `.hyxml` file format has changed.

Note: Look into `wd/warnings` for a more detailed log of errors.

4
Getting Started

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

4.1 Opening a Model

In the c2e2 folder, type command ./runC2E2 to launch C2E2 and you should be able to see the front end of the tool as Figure 4.1. Once C2E2 is launched, go ahead and open one of the examples from the `File` menu (or use `Ctrl + O`). All examples are stored in the `Examples` folder inside `c2e2` folder. For this tutorial, we will use the model of an adaptive cruise control system (see the example webpage [1]) which is stored as `TotalMotion100s.hyxml`. For the description of other examples, please refer to the examples webpage [2].

[1] https://publish.illinois.edu/c2e2-tool/example/adaptive-cruise-control/

[2] https://publish.illinois.edu/c2e2-tool/example/

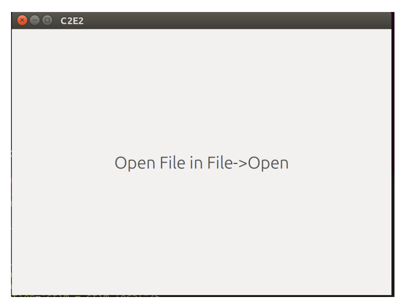


Figure 4.1: Frond end of C2E2 when launched

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

The left hand side of this window shows the *parse tree* of the model and the right hand side is the *verification pane*. You can expand the tree to see the variables, dynamics, 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.

If you want to use the default settings, please skip this section and go to Section 4.2 directly.

The coordinate transformation step K for nonlinear models has been set to 2000 by default at the top right corner of the verification pane. This value is important since the inappropriate K value will influence the final result. The user can change the K value to see different outputs.

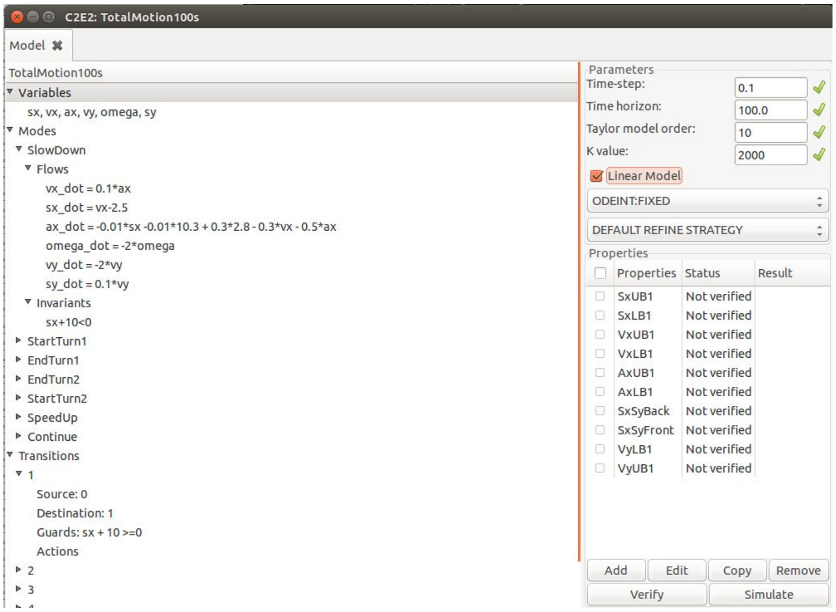


Figure 4.2: *Left:* Model parse tree. *Right:* Verification pane.

- Simulators
  Current version of C2E2 supports Odeint constant time step simulator, Odeint adaptive time step simulator and CAPD simulator. The default simulator is Odeint constant time step simulator, and it is been compiled when you opening the model. You can change the simulator by selecting different simulator from the simulator drop down menu as shown in Figure 4.3. Note that the CAPD simulator may not work for several models due to the integration method used in the CAPD library. For example, the CAPD simulator for `cardiacComposition.hyxml`, `Powertrain.hyxml`, `helicopter.hyxml` will fail simulating.

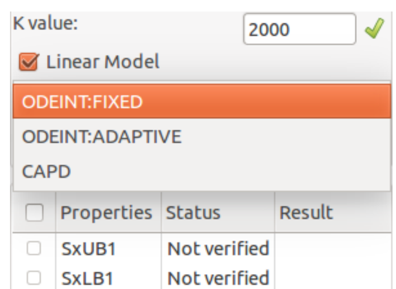


Figure 4.3: Simulator drop down menu

- Refinement strategy
  When the initial set needs refinement, C2E2 provides different refinement strategies. The default refinement strategy will refine the dimensions within the unsafe set for four times, then iteratively refine the dimension with the largest uncertainty size. C2E2 also supports user defined strategy, which can be found at Section 4.3.2. To select the strategy, please use the drop down menu on GUI as shown in Figure 4.4.

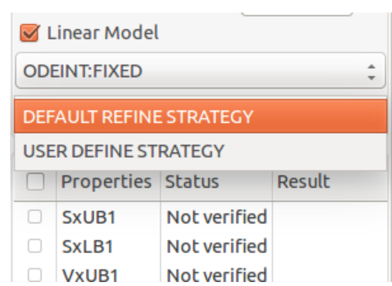


Figure 4.4: Refinement drop down menu

4.2 Properties

The right bottom corner of the main window is the property pane. This is where you can add, edit, copy properties and launch the verifier or simulator. 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 models in the `Examples` folder have already has a couple of sample properties. Here we will walk you through the steps involved in creating a new property like in Figure 4.5.

1. Click `Add` in the property pane. This launches the `Add Property` dialog box.

2. Enter a name for the property, say `VxLB1`, 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 set 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:
   vx<=2.1

5. Press `Add`.

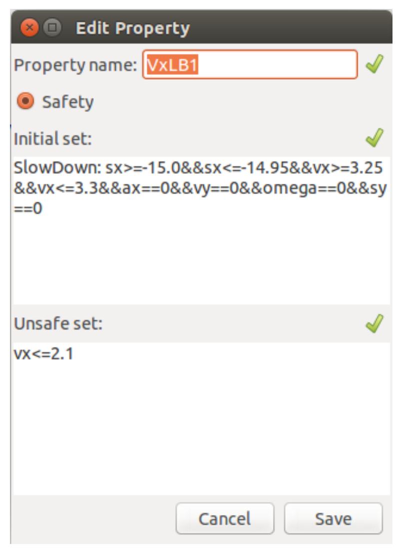


Figure 4.5: Dialog box for adding properties checks the syntax of the initial and unsafe sets.

If all the expressions are syntactically acceptable then there will be little green checks next to the textboxes and you will be able to save 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 and edit them by clicking the `Edit` button. The added properties can be saved with the model. See section 4.5.

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[3], 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 and initial set with small size.

[3] 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/ReachSet<Property name>`. You can also check the log file at `/wd/log` to check the progress of the verification. Once the verification is done, the result (Safe/Unsafe/Unknown) will show up at the Result column (as in Figure 4.6). Note that if you see verification result as `Unknown`, it is because of the following reason:

1. The system is neither robustly safe nor robustly unsafe Duggirala et al. [2013].

2. Reachable set computed bloats up and thus the number of refinements needed is too large. Please go back and check the model dynamic and properties, or simulate first to see whether the system trajectories bloats up.

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.

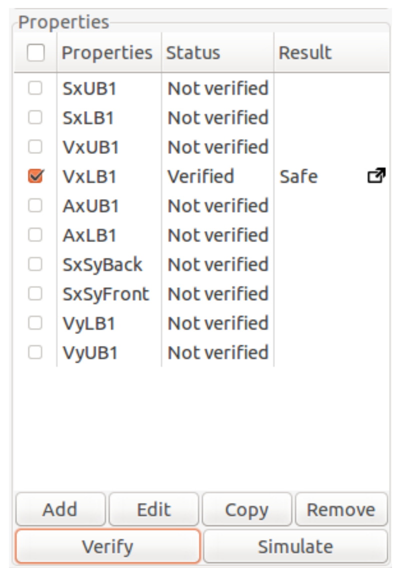


Figure 4.6: One or more properties can be selected by checking the boxes to the left of the property name. The `Verify` button launches the verification engine to verify one property at a time.

4.3.1 *Simulation*

C2E2 also allows users to generate pure simulation traces from initial sets. Once you have created a model and added a property (see Section 4.2) you can launch the simulation engine by selecting the property and then clicking the `Simulate` button. C2E2 will select several states from initial set and generate simulation traces from those initial states. Note the Safe/Unsafe result shown in this case only stands for the safety of the simulation traces instead of all the reachable states from the initial set.

4.3.2 *User defined refinement strategy*

C2E2 supports user defined refinement strategy (see Section 4.1). You can select USER DEFINE STRATEGY from the drop down menu as shown in Figure 4.4. In this case, you need to write down your strategy in a file named `refineorder.txt` and store it in the `/wd` folder. The file should be look like Figure 4.7. In each line, you should write down the index of the variables, the order of which is the same as the variables that are shown in the front end GUI. That is, "2“ means the second variable shown in the Variables list in the front end. Indexes that are larger than the dimension of system will be ignored automatically. C2E2 will refine the initial set according to the order written in `refineorder.txt` iteratively. For example, if use the refinement strategy as in Figure 4.7, the dimension corresponding to the second variable will be refined three times, then the dimension corresponding to the first variable will be refined once, then go back to the first line of `refineorder.txt` if the verification process has not terminated.

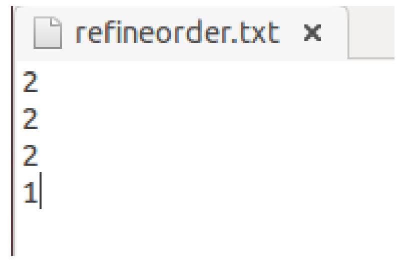


Figure 4.7: The user defined refine strategy file

4.3.3 *Change Verified Properties*

Once a property is verified the status of the property will change to `Verified`, and the result `Safe/Unsafe/Unknown` will appear next to it. The verification result comes with a small box icon appears next to it to launch the plotter (see Section 4.4). once the result has shown up, the property will be frozen temperately. At this point, if you change any of the parameters associated with the property, say the initial set or unsafe set, then the status of the property will change to `Verified*`. This (*) indicates that the property and parameters verified is outdated and you can launch the verifier again.

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 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 (as in Figure 4.9) is similar to that for creating properties:

1. Click `Add` right bottom corner of the window. This launches the `Add Plots` dialog box.

2. Enter a name for the plot, say `Sx`, in the first textbox.

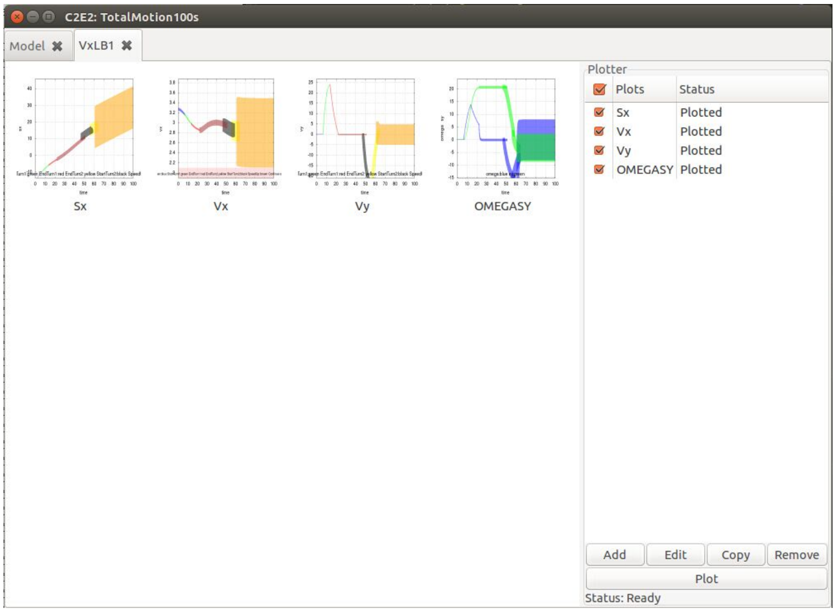


Figure 4.8: The plotting tab for property 1 without any plots.

3. Select the x-axis and y-axis variable using the drop down menu, and add more y-axis variable if needed.

4. Click the `Save` button.

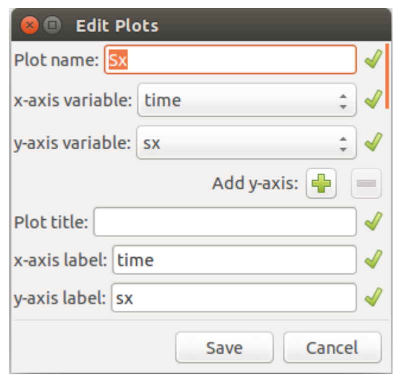


Figure 4.9: Add Plot dialog box.

C2E2 can currently create two dimensional plots as Figure 4.10. 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.

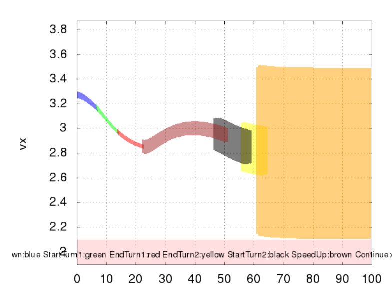


Figure 4.10: A plot of a reach tube of one variable with respect to time.

4.4.1 Plots with multiple variables and modes

If plotting multiple variables on the y-axis, the reach tube of each variable will be shown in different colors with labels. For example, in Figure 4.11 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. The axis for v_x is labeled by green and that of ω is labeled by blue.

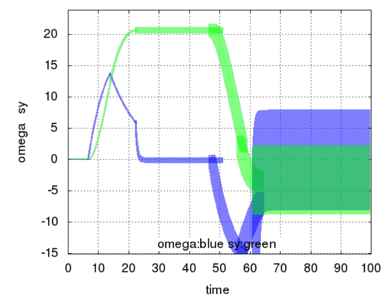


Figure 4.11: A plot of a reach tube of two variables with respect to time.

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.

4.6 Format of HyXML

Currently, C2E2 only accepts models of a certain form. The model can be defined in the HyXML file format - many examples of this can be found in the Examples/ directory. Multiple attributes must be defined for it to be a valid model. The attributes marked by * must be defined but are not used currently.

- automaton
  - variable
    - name
    - scope → LOCAL_DATA, INPUT_DATA, OUTPUT_DATA
    - type* → Real, Integer
  - mode (unique id's, consecutive order - 0 to n)
    - name
    - initial → True, False
    - dai → An equation of the form v_out = algebraic formula if v is a local variable, else v = algebraic formula if v is an output variable

    * invariant → Equation must be linear and use only local variables (Use < for <, etc)
  - transition
    * source, destination → IDs of modes
    * id* → id of transition
    * guard → Equation must be linear and use only local variables (Use < for <, etc)
    * action (optional) → Can only use local variables and must be of the form v = linear formula

- composition → automata should be a semicolon separated list of automata names you are composing

- property
  - unsafeSet → Conjunction of linear inequalities
  - initialSet → <mode_name>: Conjunction of linear inequalities
  - name → verification/simulation data can be found at wd/ReachSet<name>
  - type*
  - parameters
    * taylororder → Taylor terms you want in the expansion
    * timestep → Only used in ODE_FIXED and CAPD
    * timehorizon → Length of time you want to simulate for
    * delta

5
Using Command Line Version C2E2

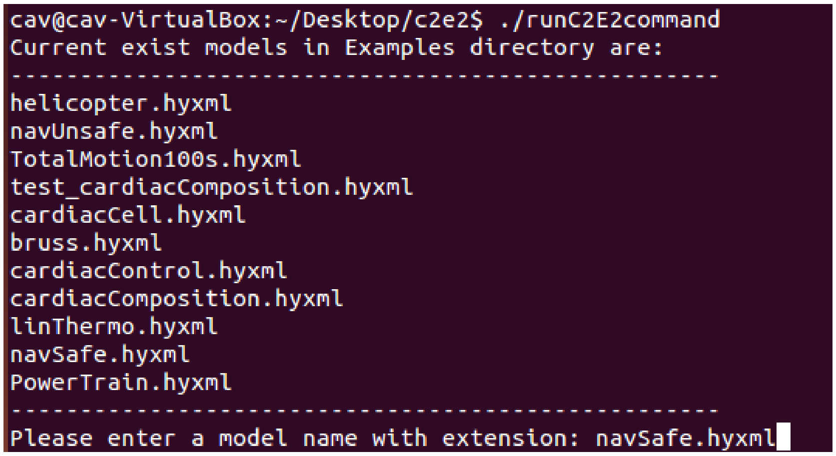
In this chapter, we give a quick introduction on running C2E2 using command line. We will use an example in the example folder.

In the c2e2 folder, type command ./runc2e2command to launch the command version C2E2 and you should able to see the model list (as in Figure 5.1). Command line version of C2E2 can only read models in the `Examples` folder. If you have customized models, please place it in the `Examples` folder as well.

Once you load a file, C2E2 will start to compile simulator, and it will ask you to identify if it is a linear model or nonlinear model (see Figure 5.2). Type Y or N to answer, then the model will be successfully loaded.

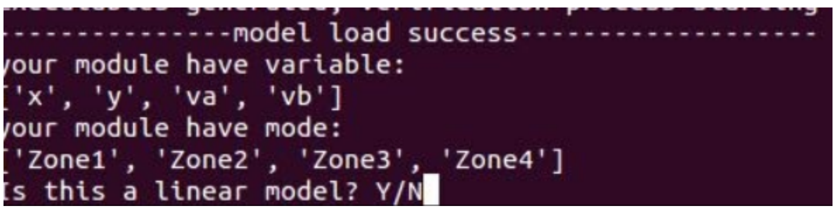
Once the model is loaded, there will be various of commands you can use. Type `help` command to check the all the commands (see Figure 5.3). These commands are similar to the functionality we have described in Section 4.1. To check the properties that come with the `.hyxml` file, type `printprop` command. This command will expand all the built in properties as shown in Figure 5.4.By typing the `verify` command, you can verify all the properties in the property list. Results will be shown once the verification ends as in Figure 5.5.

When you are finishing with the command line version C2E2, you can type `quit` command to exit.



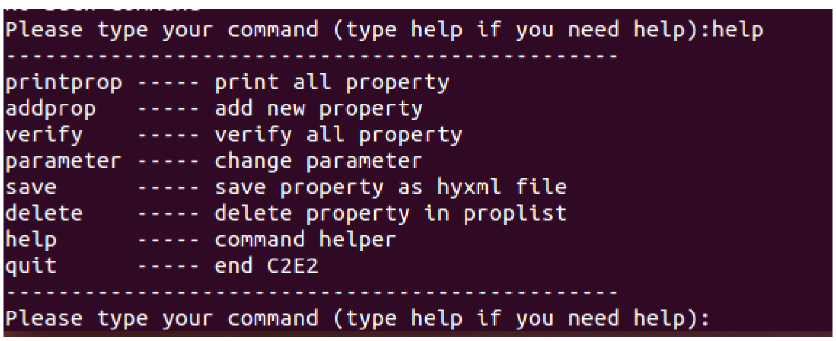
```
cav@cav-VirtualBox:~/Desktop/c2e2$ ./runC2E2command
Current exist models in Examples directory are:
---------------------------------------------------
helicopter.hyxml
navUnsafe.hyxml
TotalMotion100s.hyxml
test_cardiacComposition.hyxml
cardiacCell.hyxml
bruss.hyxml
cardiacControl.hyxml
cardiacComposition.hyxml
linThermo.hyxml
navSafe.hyxml
PowerTrain.hyxml
---------------------------------------------------
Please enter a model name with extension: navSafe.hyxml
```

Figure 5.1: Command line version C2E2 interface



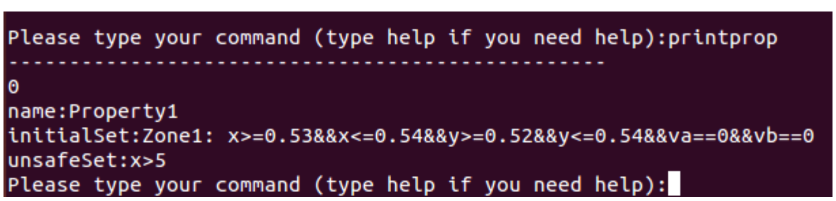
```
--------------model load success------------------
your module have variable:
['x', 'y', 'va', 'vb']
your module have mode:
['Zone1', 'Zone2', 'Zone3', 'Zone4']
Is this a linear model? Y/N
```

Figure 5.2: Type Y or N to tell C2E2 if the model is linear



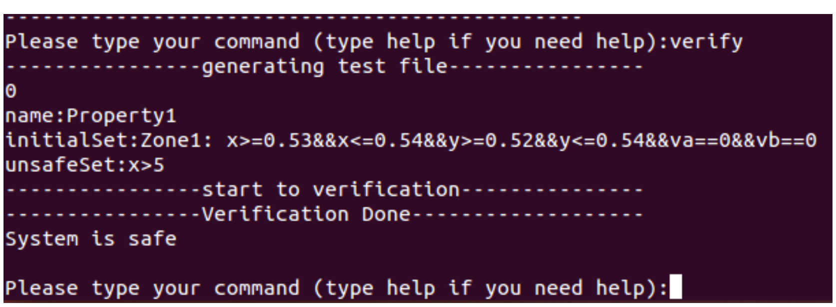
```
Please type your command (type help if you need help):help
-----------------------------------------------
printprop ----- print all property
addprop   ----- add new property
verify    ----- verify all property
parameter ----- change parameter
save      ----- save property as hyxml file
delete    ----- delete property in proplist
help      ----- command helper
quit      ----- end C2E2
-----------------------------------------------
Please type your command (type help if you need help):
```

Figure 5.3: All command options



```
Please type your command (type help if you need help):printprop
-----------------------------------------------
0
name:Property1
initialSet:Zone1: x>=0.53&&x<=0.54&&y>=0.52&&y<=0.54&&va==0&&vb==0
unsafeSet:x>5
Please type your command (type help if you need help):
```

Figure 5.4: Expand properties come with the model `.hyxml` file



```
-----------------------------------------------
Please type your command (type help if you need help):verify
---------------generating test file---------------
0
name:Property1
initialSet:Zone1: x>=0.53&&x<=0.54&&y>=0.52&&y<=0.54&&va==0&&vb==0
unsafeSet:x>5
---------------start to verification--------------
---------------Verification Done------------------
System is safe

Please type your command (type help if you need help):
```

Figure 5.5: Verify the properties in the property list

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/)
12. Python Sympy library (http://www.sympy.org/en/index.html)
13. Boost libraries (http://www.boost.org)

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.