PARASARA SRIDHAR DUGGIRALA, MATTHEW POTOK, SAYAN MITRA, AND MAHESH VISWANATHAN

C2E2 USER'S GUIDE

Compiled on October 7, 2014

Contents

1 Acknowledgements

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 Duggirala et al. [2013, 2014], Sukumar and Mitra [2011]. In a nutshell, it 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 user-provided annotations 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.

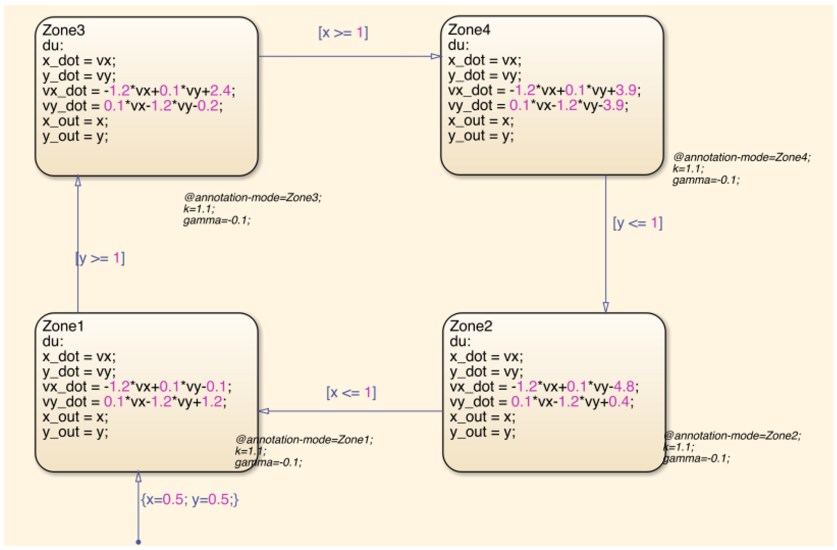
- Download the latest distribution of C2E2 distribution from: `http://publish.illinois.edu/c2e2-tool/download/`.
- Unzip the files in a local directory, say `/C2E2/`.
- Go into `/C2E2/` and run `sudo ./installRequirements`. This should install all the packages needed and may take a while.
- If the above step succeeds then run `./installC2E2` to install the program. 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.

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. In Chapter **??** we will describe creation of new Stateflow^{TM} models.

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 robot moving in the 2D plane which is stored in the file `FourNav.mdl`.



```
Zone3
du:
x_dot = vx;
y_dot = vy;
vx_dot = -1.2*vx+0.1*vy+2.4;
vy_dot = 0.1*vx-1.2*vy-0.2;
x_out = x;
y_out = y;
```

```
Zone4
du:
x_dot = vx;
y_dot = vy;
vx_dot = -1.2*vx+0.1*vy+3.9;
vy_dot = 0.1*vx-1.2*vy-3.9;
x_out = x;
y_out = y;
```

```
Zone1
du:
x_dot = vx;
y_dot = vy;
vx_dot = -1.2*vx+0.1*vy-0.1;
vy_dot = 0.1*vx-1.2*vy+1.2;
x_out = x;
y_out = y;
```

```
Zone2
du:
x_dot = vx;
y_dot = vy;
vx_dot = -1.2*vx+0.1*vy-4.8;
vy_dot = 0.1*vx-1.2*vy+0.4;
x_out = x;
y_out = y;
```

```
@annotation-mode=Zone3;
k=1.1;
gamma=-0.1;
```

```
@annotation-mode=Zone4;
k=1.1;
gamma=-0.1;
```

```
@annotation-mode=Zone1;
k=1.1;
gamma=-0.1;
```

```
@annotation-mode=Zone2;
k=1.1;
gamma=-0.1;
```

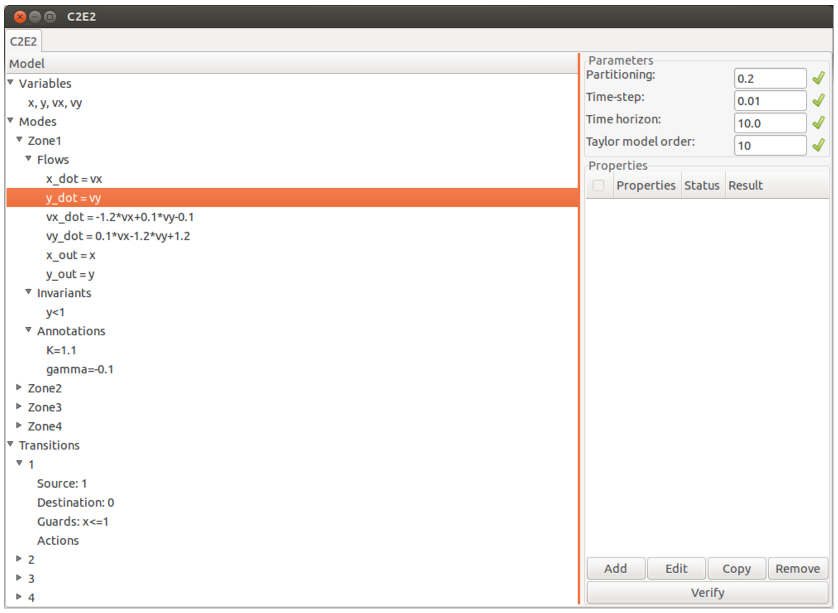
Figure 4.1: Stateflow model of a simple robot moving the 2D-plane.

The Stateflow^{TM} model in this file is shown in Figure 4.1. In brief, the model has four continuous variables x, y, vx, vy and four modes *Zone1*, ..., *Zone4*, and in each mode the behavior of the robot is described by a linear differential equation with different inputs.

There are four transitions, each going from one mode to another. The starting state is defined as $x = 0.5$ and $y = 0.5$ in *Zone1*. You can generate numerical simulations of this model in Matlab. You may notice one additional piece of information in the model: The the textboxes with strings like `@annotation-mode=Zone3; k=1.1; gamma=-0.1`. These *annotations* are used by C2E2 for computing reach sets and, for now, they have to be provided by the user. See Chapter 5 for more on annotations. Also, 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



```
Model
▾ Variables
    x, y, vx, vy
▾ Modes
  ▾ Zone1
    ▾ Flows
        x_dot = vx
        y_dot = vy
        vx_dot = -1.2*vx+0.1*vy-0.1
        vy_dot = 0.1*vx-1.2*vy+1.2
        x_out = x
        y_out = y
    ▾ Invariants
        y<1
    ▾ Annotations
        K=1.1
        gamma=-0.1
  ▸ Zone2
  ▸ Zone3
  ▸ Zone4
▾ Transitions
  ▾ 1
      Source: 1
      Destination: 0
      Guards: x<=1
      Actions
  ▸ 2
  ▸ 3
  ▸ 4
```

Parameters

Partitioning:	0.2 ✓
Time-step:	0.01 ✓
Time horizon:	10.0 ✓
Taylor model order:	10 ✓

Properties

	Properties	Status	Result

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 a convenient representation of the model. Notice that for each mode, the corresponding annotations also appear in the parse tree.

4.2 Editing

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 `safe1`, in the first textbox.

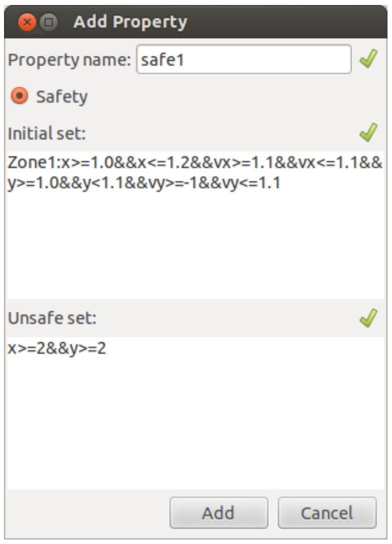
3. Enter a linear predicate on the variables to specify the *initial set* or the *starting states* in the second textbox. Currently, the syntax for specifying the initial state is as follows:
   ⟨ `mode-name` ⟩ : ⟨ (`linear-inequality` &&)+ ⟩.
   For example, for the above model:
   `Zone1`: x>=1.0 && x<=1.1 && y==1.0 && vx>=2.0 && vx<=2.2 && vy>=-1.0 && vy<=1.1
   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:
   x>=5.1 && y>=4

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 propery 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.



Property name: safe1 ✓

● Safety

Initial set: ✓

Zone1:x>=1.0&&x<=1.2&&vx>=1.1&&vx<=1.1&&y>=1.0&&y<1.1&&vy>=-1&&vy<=1.1

Unsafe set: ✓

x>=2&&y>=2

Figure 4.3: Dialog box for adding properties checks the syntax of the initial and unsafe sets.

4.3 Verifying

Once you have created and annotated 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.

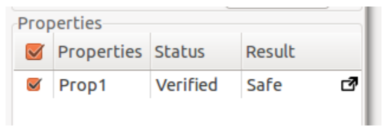
Warning! Verification may take more than a few minutes.

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 annotations, 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.


Properties

✓	Properties	Status	Result
✓	Prop1	Verified	Safe

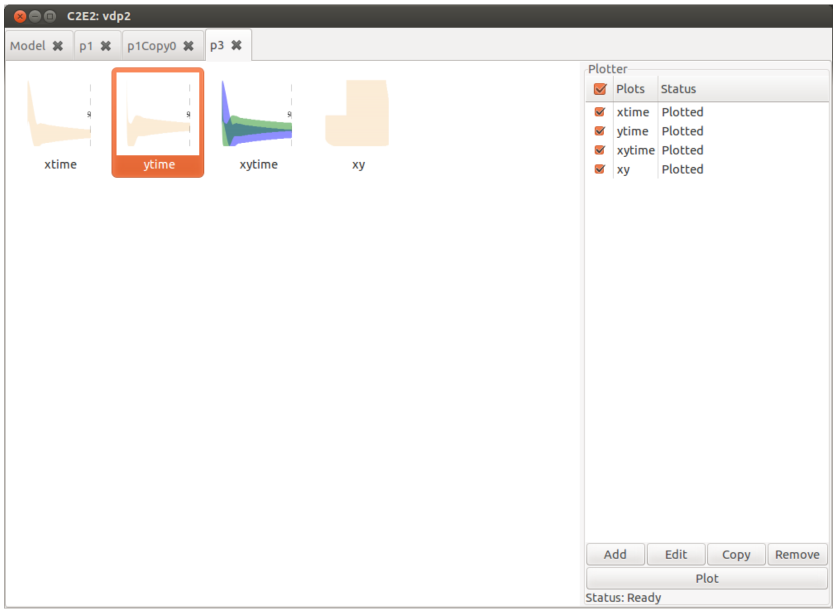
Figure 4.4: One or more properties can be selected by checking the boxes to the left of the property name. The `Verify` button launches the verification engine to verify one property at a time.

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



Plots	Status
xtime	Plotted
ytime	Plotted
xytime	Plotted
xy	Plotted

Status: Ready

Figure 4.5: 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. C3E2 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.

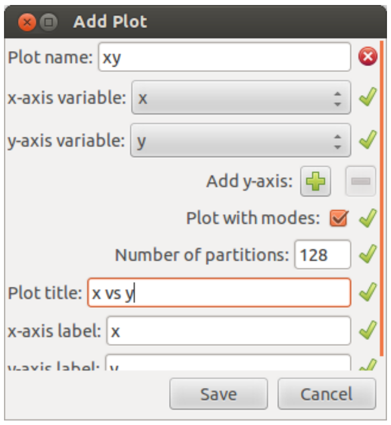
As the program plots the reach sets, you will see icons appear on the left hand side of the window with a preview of what the plot looks like as well as the plot name below it. You can expand the plot by double clicking on these icons. <insert an image here> You can navigate the the first window and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

Warning! Plotting can take time but the results can be pretty.

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 `.pdf` figures of the plot.

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 4.8 the x-axis is time and the y-axis shows the reach tubes of two variables x and y which are shown in blue and green respectively, and the axis for x is labeled by blue and that of y is labeled by green.



Plot name: xy
x-axis variable: x
y-axis variable: y
Plot with modes: ✓
Number of partitions: 128
Plot title: x vs y
x-axis label: x
y-axis label: y

Figure 4.6: Add Plot dialog box.

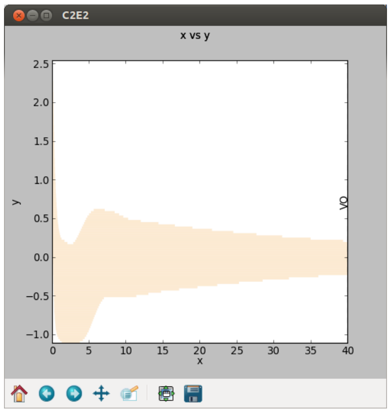


Figure 4.7: A plot of a reach tube of one variable with respect to time.

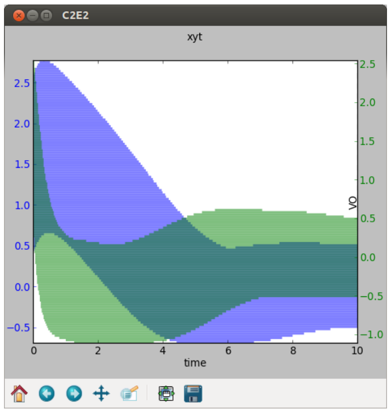


Figure 4.8: A plot of a reach tube of two variables with respect to time.

If plotting a single variable with respect to time across different modes, then the reach tube in each mode is plotted in a different color as in Figure 4.9. Vertical dashed lines are shown to deliniate the mode(s) visited by the system over different intervals of time. Owing to nondeterminism—in the initial state and transitions—and over-approximations in C2E2's reach set computation, at a given point in time the reach tube may be in multiple modes. For example, in Figure 4.9, the system starts out in `Zone1`, and then after 1.2 seconds it enters `Zone3`, but between 1 second and 1.2 seconds, it may be either zone. Similarly, between 1.7 and 2 seconds it may be in `Zone3` or `Zone4`.

If plotting multiple variables with respect to time across different modes, then the reach tube of each variable is plotted in one color across each mode to avoid overloading the plot with color information (see Figure 4.10). The vertical dashed lines serve the same purpose as in the case of single variables.

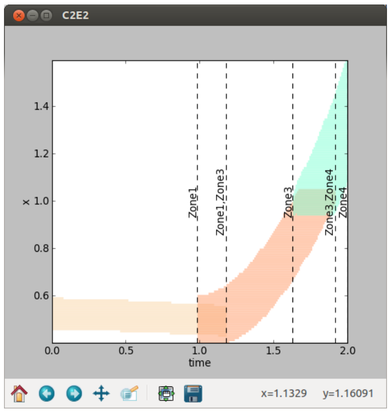


Figure 4.9: A plot of a reach tube of a single variable across multiple modes.

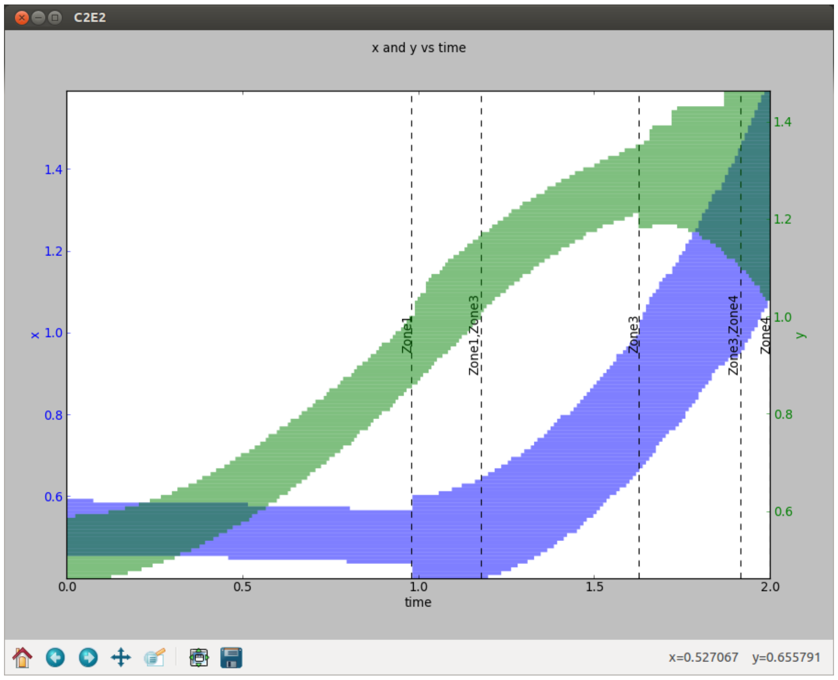


Figure 4.10: A plot of a reach tube of state variables x and y with respect to time across multiple modes.

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 or the annotations in the C2E2 frontned 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, annotations, and the properties. The reach sets computed during verification are stored in the working directory `/wd/` but the currently they cannot be loaded.

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hyxml>
<hyxml>
 <variable scope="LOCAL_DATA" type="Real" name="x"/>
 <variable scope="LOCAL_DATA" type="Real" name="y"/>
      ...
 <automaton name="default_automaton">
  <mode initial="True" id="0" name="Zone1">
   <dai equation="x_dot = vx"/>
   <dai equation="y_dot = vy"/>
   <dai equation="vx_dot = -1.2*vx+0.1*vy-0.1"/>
   <dai equation="vy_dot = 0.1*vx-1.2*vy+1.2"/>
   <invariant equation="y&lt;1"/>
   <annotation mode="Zone1">
    <K value="1.1"/>
    <gamma value="-0.1"/>
    <type string="exponential" value="1"/>
   </annotation>
  </mode>
  ...
    <mode initial="False" id="3" name="Zone4">
   <dai equation="x_dot = vx"/>
   <dai equation="vx_dot = -1.2*vx+0.1*vy+3.9"/>
   <invariant equation="y&gt;1"/>
   <annotation mode="Zone4">
    <K value="1.1"/>
    <gamma value="-0.1"/>
    <type string="exponential" value="1"/>
   </annotation>
  </mode>
  <transition source="1" destination="0" id="1">
   <guard equation="x&lt;=1"/>
  </transition>
  <transition source="2" destination="3" id="2">
   <guard equation="x&gt;=1"/>
  </transition>
            ...
 </automaton>
 <property unsafeSet="x&gt;=4" type="0" name="Prop1"
      initialSet="Zone1: x&gt;=0.5&amp;&amp;x&lt;=0.55&amp;&amp;
      y&gt;=0.5&amp;&amp;y&lt;=0.55&amp;&amp;vx==0&amp;&amp;vy==0"/>
</hyxml>
```

5 Model Annotations

Each mode of the hybrid automaton model has to be annotated with a *discrepancy function* Duggirala et al. [2013] by the user. Roughly, a discrepancy function gives a measure of the distance between two neighboring trajectories as a function of time. The current version of C2E2 supports only exponential discrepancy functions. Suppose the dynamics of the model in a given mode is:

$$\dot{x} = f(x). \tag{5.1}$$

For any initial state x_0 and time t, let $\xi(x_0, t)$ be the solution of the system at time t. An exponential discrepancy function for this system is specified by two parameters $K > 0$ and a real number γ that satisfy the following for any two neighboring initial states x_1, x_2 and any time t:

$$||\xi(x_1, t) - \xi(x_2, t)|| \leq K||x_1 - x_2||e^{\gamma t}. \tag{5.2}$$

So, you have to annotate each mode of the model with K and γ such that Equation (5.2) is satisfied. These annotations have to be added in the `.mdl` file as a text label. See the provided examples.

5.1 Finding Annotations

For linear dynamics you can choose $K = 1$ and γ as the largest eigenvalue of the system matrix. For nonlinear models, several heuristic approaches for finding discrepancy functions have been outlined in Duggirala et al. [2013], Huang et al. [2014], Huang and Mitra [2012]. In a future version of this manual, we will provide more details on this issue of finding annotations.

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.

Zhenqi Huang and Sayan Mitra. Computing bounded reach sets from sampled simulation traces. In *In The 15th International Conference on Hybrid Systems: Computation and Control (HSCC 2012), Beijing, China.*, 2012.

Zhenqi Huang, Chuchu Fan, Alexandru Mereacre, Sayan Mitra, and Marta Z. Kwiatkowska. Invariant verification of nonlinear hybrid automata networks of cardiac cells. In *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2014. ISBN 978-3-319-08866-2.

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.