

Recitation 11: KeYmaera X Tips and Tricks
15-424/15-624/15-824 Logical Foundations of Cyber-Physical Systems

Notes by: Brandon Bohrer

Edits by: Yong Kiam Tan (yongkiat@cs.cmu.edu)

1 Announcements

- Project proposal due 11:59PM on Sunday.

2 Motivation and Learning Objectives

Now that you are done with the finals, the only thing left for you to do in this class is finishing your final project and preparing to compete in the CPS V&V Grand Prix. This (short) recitation focuses on KeYmaera X tips and tricks for modeling projects, and also how to prove advanced properties of ODEs.

If you are doing a theory or implementation project, please come talk to us separately if you need help/advice.

3 Lab 4 Veribot

Note: In class, we explored the ideas here in the context of my solution Lab 4 Question 2. The specific solutions for Lab 4 are omitted for these notes but the ideas should be broadly applicable.

3.1 Creating the Model

Here are some simple tips for creating your model. They should have appeared on your Betabot/Veribot feedback already but here is a quick summary:

1. Always **document** your model. Adding comments to your model makes it much easier for others (and for yourself in the future) to understand what you were trying to model.
2. Structure your model, e.g., using braces and spaces (no tabs) and make use of the KeYmaera X's support for definitions with the Definitions block. Use braces to group relevant parts of terms/formulas/programs together. For example, the controller for your hybrid program model could be explicitly put in between braces: they do not cost you anything and greatly improves readability.

Creatively grouping parts of your formulas together could also help to make your proofs more straightforward. For example, if you have a conjunction $P \wedge Q \wedge R$ in your postcondition and you know that P, R are the “straightforward” ones, then perhaps writing your postcondition as $(P \wedge R) \wedge Q$ would make it easier to use $\square \wedge$ afterwards.

3. Matter of taste: you can use the special functions `max`, `min`, `abs` in your model which are interpreted with their standard mathematical meaning in KeYmaera X. This not only improves readability, but could also be more intuitive compared to encoding these functions with formulas.

The downside of this is that tactics like `dI` and `Mathematica` do not play very nicely with these special functions. Make sure to remove them in your proof before calling `QE` if your `QE` call gets stuck.

4. Matter of taste: it could help to introduce additional variables or change the meaning of existing ones if it helps to simplify your model.
5. Think backwards: start from the “worst case” for your controller/model and then design the rest of your controller so that it ensures that even in this “worst case” your controller is still safe.

In addition, this “worst case” can often feature as the default option for your controller that immediately ensures that your controller is never vacuous (just think of the braking option in all of the labs you have done).

6. Start from an easy model e.g., with an easy controller. Verify it before moving on to more complicated ones. The simplification might even be “good enough” if you can justify it.
7. Make sure that you ask the right questions in your model i.e., can you justify why the safety postcondition truly reflects your intuition as to what it means for the robot to be safe?

3.2 Proving the Model

Here are some simple tips for working on the proof of your model after you have created it:

1. When iterating between updating your model and proving it, do not bother with branches that you already know will work.

However, remember to keep partial tactics so that it is easier to piece together your proof attempts afterwards.

2. Be cognizant of where you are “in your model”. Since most of your `dL` proofs work by removing `dL` operators step by step, you should have a high-level intuition e.g., of which branch of a choice you are currently working on for your controller.
3. Try the `ODE` tactic for simple looking `ODE` goals. If that doesn’t work, then go back and redo the `ODE` proofs more manually. We will see some particularly useful `ODE` proof rules that KeYmaera X implements later.
4. Remember that `dI` can work directly with disjunctions. There is no need to manually split disjunctive cases that you know will prove by `dI`.

5. Help `QE` along manually if necessary. It is not the best at dealing with complicated goals. If you have an intuition for why your goal is going to be true, then make it more obvious e.g., by cutting or hiding goals. Sometimes, even manually splitting disjunctions can help to speed it up.

4 Advanced ODE Proofs

For the early labs (Labs 1 and 2) we dealt with ODEs using KeYmaera X's support for solving them. However, remember that the term language of `dL` only allows you to write down *polynomials* over the variables of concern. In particular, the `solve` tactic will only solve your ODEs if the solution can be expressed as a polynomial in the time variable.

Even very simple ODEs of the kind you might encounter in your projects might no longer have polynomial solutions. For example, consider the following ODE:

$$x' = x$$

Its general solution is given by $x(t) = x_0 e^t$, where x_0 is the initial value of x , which is already outside our polynomial term language.

We saw techniques in `dL` for dealing with these more difficult ODEs, namely `dI`,`dC`,`dG`, and we practiced using `dI`,`dC` in the later labs (Labs 3 and 4) for robots moving on a circle. These labs, however, were specifically designed so that you would not require `dG` for your proofs. For your more open-ended projects this might not be the case. Our goal here is to get an intuitive understanding for when you might need to use differential ghosts in your proofs. We will also look at the existing support in KeYmaera X for making effective use of differential ghosts.

On Assignments 4 and 5 you already saw how differential ghosts can help you with ODE reasoning. We proved the following bounds on e^t for the time interval $0 \leq t \leq 1$:

$$1 + t + \frac{t^2}{2} + \frac{t^3}{6} \leq e^t \leq 1 + t + \frac{t^2}{2} + \frac{t^3}{4}$$

The lower bound could be proved by a series of `dI`,`dC` steps as you saw on Assignment 4. In fact the lower bound works for all $t \geq 0$. In contrast, the right bound made essential use of the `dbx` tactic from Assignment 5. You proved the following `dL` formula:

$$x = 1 \wedge t = 0 \rightarrow [\{x' = x, t' = 1 \ \& \ t \leq 1\}]x \leq 1 + t + \frac{t^2}{2} + \frac{t^3}{4}$$

In particular, this would allow you to give an approximate upper bound on x in the postcondition:

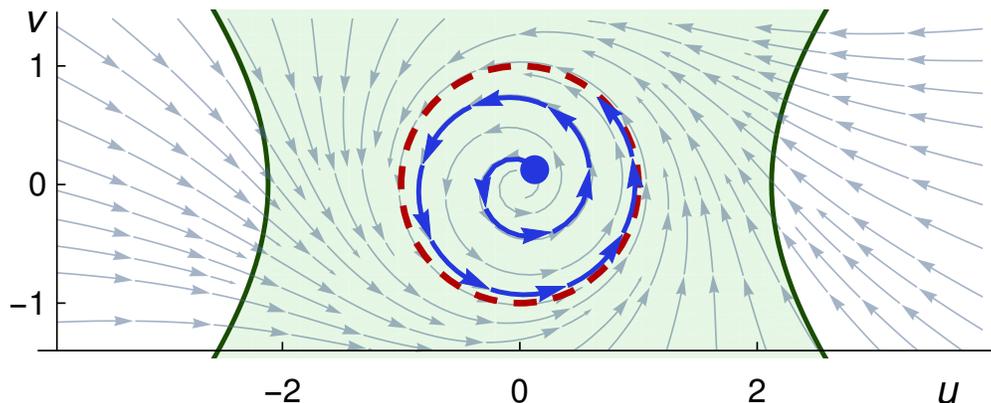
$$x = 1 \wedge t = 0 \rightarrow [\{x' = x, t' = 1 \ \& \ t \leq 1\}]x \leq \frac{11}{4}$$

4.1 Running Example

We will use the following ODE as a running example:

$$\alpha \equiv u' = -v + \frac{1}{4}u(1 - u^2 - v^2), v' = u + \frac{1}{4}v(1 - u^2 - v^2)$$

Here is a visualization of the ODE:



There are a few properties of interest:

- All trajectories (e.g., in blue) except the one from the origin spiral towards the (red dotted) unit circle. However, notice that trajectories starting in the unit circle will stay in it, trajectories starting outside the circle stay out, and trajectories starting on the circle stay on the circle. More formally, we might say that the following formulas are all valid (the strict inequalities can be replaced with non-strict ones as well):

$$u^2 + v^2 < 1 \rightarrow [\alpha]u^2 + v^2 < 1$$

$$u^2 + v^2 > 1 \rightarrow [\alpha]u^2 + v^2 > 1$$

$$u^2 + v^2 = 1 \rightarrow [\alpha]u^2 + v^2 = 1$$

- The origin is itself invariant because it is an equilibrium point of the ODEs (just plug $u = 0, v = 0$ into the RHS and simplify which will give $u' = 0, v' = 0$):

$$u = 0 \wedge v = 0 \rightarrow [\alpha](u = 0 \wedge v = 0)$$

- The shaded green region characterized by $u^2 \leq v^2 + \frac{9}{2}$ has a very similar property although less visually striking than the spiral: all trajectories starting in the green region will stay in it. Intuitively, this works because all the arrows in the plot along the boundary point “inwards”. Formally again:

$$u^2 \leq v^2 + \frac{9}{2} \rightarrow [\alpha]u^2 \leq v^2 + \frac{9}{2}$$

Note: In fact, most of these basic properties can be proved automatically using the powerful ODE tactic. However, we will look at the individual tactics that ODE internally calls.

4.2 Darboux (In)equalities

Let us consider the first group of formulas mentioned above ($>$, $<$ can also be replaced by \geq , \leq):

$$u^2 + v^2 < 1 \rightarrow [\alpha]u^2 + v^2 < 1$$

$$u^2 + v^2 > 1 \rightarrow [\alpha]u^2 + v^2 > 1$$

$$u^2 + v^2 = 1 \rightarrow [\alpha]u^2 + v^2 = 1$$

Surprisingly, none of these formulas are easy to prove with just dI,dC.

Exercise 1:

Try to use dI to prove $u^2 + v^2 < 1 \rightarrow [\alpha]u^2 + v^2 < 1$.

Answer:

$$\frac{\vdash -\frac{1}{2}(u^2 + v^2)(u^2 + v^2 - 1) \leq 0}{\vdash \dots}}{\text{dI } u^2 + v^2 < 1 \vdash [\alpha]u^2 + v^2 < 1}$$

The real arithmetic at the end does not work prove. Our usual trick of using a dC would not really work either. Intuitively, the problem is what we observed in the visualization: even if we started inside the unit disk, the trajectory spirals towards leaving the unit disk i.e., the property that we want to show holds true is getting “less” true over time. This is one way in which you can recognize that dG would be needed.

Fortunately, KeYmaera X knows about these sorts of situations as well. It provides you with a tactic called **dbx** (which you used on Assignment 5) that automatically uses differential ghosts to prove such properties. Formally, the derived proof rule that **dbx** implements looks very similar to dI. It lets you prove an invariant inequality property (g is a polynomial that you can choose, \succcurlyeq stands for either \geq or $>$, and the proof rule works symmetrically for \leq , $<$ as well):

$$(\text{dbx}\succcurlyeq) \frac{Q \vdash [x' := f(x)](p)' \geq gp}{p \succcurlyeq 0 \vdash [\{x' = f(x) \& Q\}]p \succcurlyeq 0}$$

Exercise 2:

How could you use this proof rule for the problem in the previous exercise?

Answer: We already arranged it nicely! Notice that if we let $p \equiv u^2 + v^2 - 1 < 0$, then $(p)'$ (which we have calculated above) will simplify to $-\frac{1}{2}(u^2 + v^2)p$ and so we can apply **dbx** \succcurlyeq by choosing $g \equiv -\frac{1}{2}(u^2 + v^2)$.

The similarities to dI do not just end there. For example, the **dbx** tactic also allows you to use this proof rule (g is again a polynomial that you can choose):

$$(\text{dbx}) \frac{Q \vdash [x' := f(x)](p)' = gp}{p = 0 \vdash [\{x' = f(x) \& Q\}]p = 0}$$

Using these we can prove all of the properties that we wanted about the invariants involving the unit circle/disk.

4.3 Equilibria and Equational Postconditions

The second property we observed for this system was that the origin is an equilibrium point. In particular, when we set the RHS to $u = 0, v = 0$, we noted that $u' = 0, v' = 0$. Intuitively, if we started at an equilibrium point, then we should stay at the equilibrium point because the ODE has no movement at all. Thus, this formula is valid:

$$u = 0 \wedge v = 0 \rightarrow [\alpha](u = 0 \wedge v = 0)$$

This, again, will turn out to be surprisingly difficult to prove with just dI,dC.

Exercise 3:

Try to use dI to prove $u = 0 \wedge v = 0 \rightarrow [\alpha](u = 0 \wedge v = 0)$.

Answer: This will fail fairly obviously because the resulting premise would require proving $-v + \frac{1}{4}u(1 - u^2 - v^2) = 0 \wedge u + \frac{1}{4}v(1 - u^2 - v^2) = 0$ which is not true without any additional assumptions on u, v .

Exercise 4:

Try to use dI to prove the “smarter” way of asking the same question:

$$u^2 + v^2 = 0 \rightarrow [\alpha](u^2 + v^2 = 0)$$

Answer: We actually already did most of the calculations earlier. In fact, the proof can be completed using dbx by choosing g appropriately. This illustrates a property that we already saw for dI: even though $u = 0 \wedge v = 0$ and $u^2 + v^2 = 0$ are equivalent formulas in real arithmetic, they have very different differential structure.

This approach of rewriting the postcondition, in fact, will always work for proving equational postconditions. The technical details are advanced¹ (you can ask me if you are interested). KeYmaera X already implements a tactic **dRI** that handles equational reasoning. You can access it from the browsing menu, just like **dbx**. It will be able to prove the aforementioned question from $u = 0 \wedge v = 0$ directly. If you have a postcondition that contains only equations (possibly with conjunctions), give it a try and let me know if you find that it fails to prove your property.

4.4 Barrier Certificates

The final property we saw with the visualization was that the green region has arrows all pointing “inwards”. Formally, we claimed that the following formula is valid ($u^2 \leq v^2 + \frac{9}{2}$ is

¹For those of you who were at my talk, the implementation in KeYmaera X actually does something different than what I presented there.

the green region):

$$u^2 \leq v^2 + \frac{9}{2} \rightarrow [\alpha]u^2 \leq v^2 + \frac{9}{2}$$

Straightforward dI fails because a simple calculation shows that we need to prove:

$$u(-v + \frac{u(1 - u^2 - v^2)}{4}) \leq v(u + \frac{v(1 - u^2 - v^2)}{4})$$

This property is in fact provable with `dbx` using a polynomial choice of g but the details are advanced (again, ask me if you are interested). Instead, we will make use of the barrier certificates proof rule:

$$(BC) \frac{Q, p = 0 \vdash [x' := f(x)](p)' > 0}{p \not\approx 0 \vdash [\{x' = f(x) \ \& \ Q\}]p \not\approx 0}$$

Notice that, unlike all of the proof rules from before, this rule allows you to assume $p = 0$ in the antecedent of the premise. In return, we are required to prove a *strict* inequality in the postcondition. Making this inequality strict is *essential* as the rule is unsound with the non-strict inequality (see e.g., Assignment 3). The `barrier` tactic implements this rule. In fact, this is one of the first things that the default `ODE` tactic would try, so more often than not, you could just click on `ODE` right away.