Recitation 9: 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 and Katherine Cordwell (kcordwel@cs.cmu.edu).

1 Announcements

- Lab 3 Veribot will be graded after Lab 4 Betabot. The proof redo will be due Tuesday, Nov. 12 before 11:59 pm (submit by email).
- Lab 4 Betabot extended to be due Saturday, Nov. 2 before 11:59 pm; Veribot extended to Saturday, Nov. 9 before 11:59 pm.
- Assignment 5 due Thursday, Nov. 14 before 11:59 pm.
- Proposal due Friday, Nov. 15 before 11:59 pm. While the White Paper was graded mostly for completion and feedback, we will be looking for more specific things on the Proposal. You should ensure that you have made concrete progress since the White Paper, as we will be looking for preliminary results.

2 Motivation and Learning Objectives

This (short) recitation focuses on KeYmaera X tips and tricks. We'll discuss how to prove advanced properties of ODEs and how to do proofs for hybrid games. While the general goal is to give you a broader/deeper sense of the capabilities of KeYmaera X, you may also find the extra practice helpful for Lab 4., and it may be especially relevant if you are doing a modeling project. (Of course, no matter what type of project you are doing, you can come talk to us for help/advice!)

Specifically, we will:

- Briefly discuss how one might handle complicated motion like circular motion in KeYmaera X.
- Introduce the `dbx` tactic for differential ghosts and some other ways of handling ODEs.
- Explore proving hybrid games in KeYmaera X.
- Recap with some general tips and tricks for writing and proving models.

3 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.

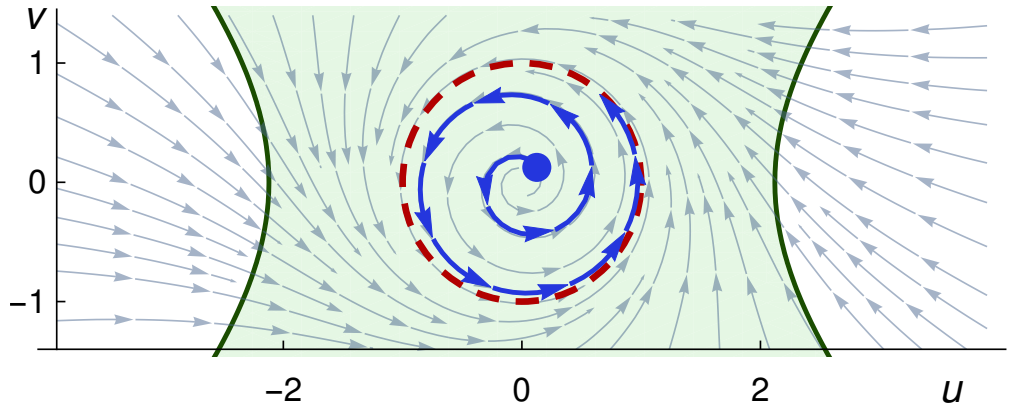
We saw techniques in dL for dealing with these more difficult ODEs, namely dI,dC,dG. Our goal here is to get an intuitive understanding for when you might need to use differential ghosts in your proofs, and how to use the existing support in KeYmaera X for making effective use of differential ghosts.

3.1 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 $u' = 0, v' = 0$):

$$u = 0 \land v = 0 \to [\alpha](u = 0 \land v = 0)$$

- 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:

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

Note: In fact, most of these basic properties can be proved automatically using the powerful `ODE` tactic. However, we will look at some of the individual tactics that `ODE` internally calls. Some relevant KeYmaera X models can be found in the"code" file corresponding to this recitation on the course webpage.

3.2 Darboux (In)equalities

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

$$u^2 + v^2 < 1 \to [\alpha]u^2 + v^2 < 1$$
$$u^2 + v^2 > 1 \to [\alpha]u^2 + v^2 > 1$$
$$u^2 + v^2 = 1 \to [\alpha]u^2 + v^2 = 1$$

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

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

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

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):

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

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):

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

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

3.2.1 Bounding Taylor Series and using dbx in KeYmaera X

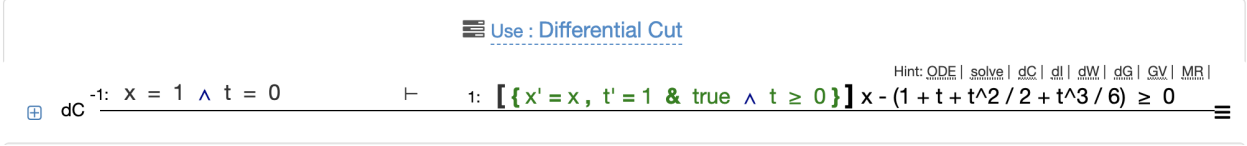
On Assignment 4 you proved some lower bounds on e^t with a series of dI,dC steps.

Now we'll discuss how to use the `dbx` tactic to prove bounds on Taylor series. Let's prove the following formula in KeYmaera X:

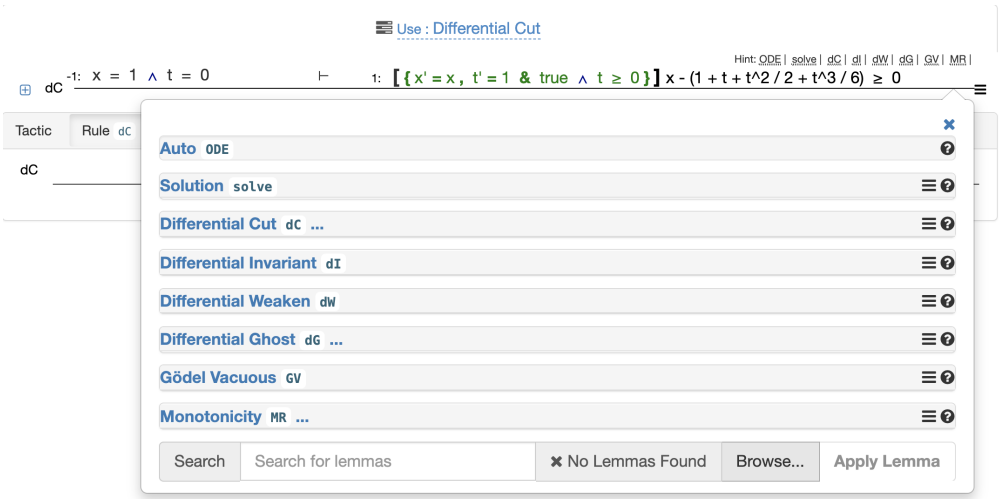
```
x=1&t=0->[{x'=x,t'=1}]x-(1+t+t^2/2+t^3/6)>=0
```

Here is a step-by-step guide to the proof:

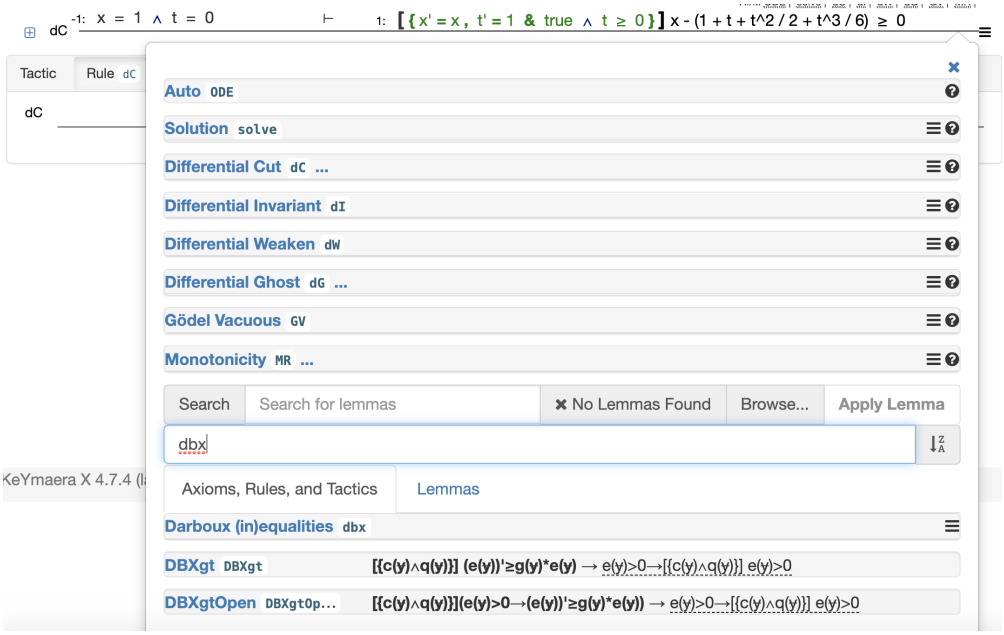
1. First, use a differential cut to add $t \geq 0$ to the domain constraint. Your goal should look like this after the cut:



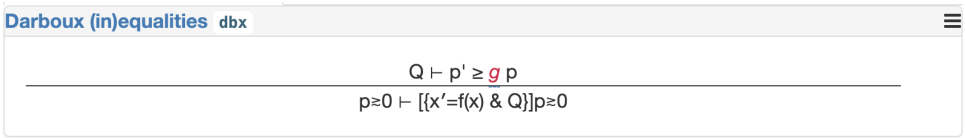
2. Now, right-click on the ODE in the succedent, and select `Browse...`



3. In the search box where it says "Filter lemmas", type `dbx` and you should see a proof rule called "Darboux (in)equalities".



This is a tactic that automatically uses differential ghosts to prove advanced properties of differential equations. Expand the rule information for `dbx` by clicking on ☰:



This rule says that in order to prove $p \geq 0$ (or $p > 0$) invariant, it suffices to prove that $[x' := f(x)](p)' \geq gp$ for some cofactor polynomial g.[1] Notice that the cofactor polynomial g has to be supplied to the tactic as an argument.

In this case, we can do a rough calculation (assuming $x' = x, t' = 1$ and using the domain constraint $t \geq 0$ for the last inequality):

$$\overbrace{(x - (1 + t + \frac{t^2}{2} + \frac{t^3}{6}))'}^{p'} = x - (1 + t + \frac{t^2}{2}) \geq \overbrace{1}^{g} \cdot \overbrace{(x - (1 + t + \frac{t^2}{2} + \frac{t^3}{6}))}^{p}$$

So we can apply the rule by choosing g to be 1.

4. Click on g, type `1`, press "Enter", and click on the tactic name to run it. The proof should be completed.

Exercise 3:
Since e^t grows faster than any polynomial function as $t \to \infty$, it is impossible to prove a polynomial upper bound on it for all $t \geq 0$. It is possible, however, if we restrict ourselves to a bounded time interval like $t \leq 1$. Prove the following formula in KeYmaera X:

```
x=1&t=0->[{x'=x,t'=1&t<=1}]x<=1+t+t^2/2+t^3/4
```

Hint: A very similar proof should work for this question, but you have to normalize the postcondition to have 0 on one side of the inequality first in order to use `dbx` in your proof. (How?)

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

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

Together with the previous part of this question, this proves the following approximation to e^t on the interval $0 \leq t \leq 1$:

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

This bound can be used, for example, to show $1 \leq e^t \leq 2.75$ for $0 \leq t \leq 1$.

[1] Recall that dI would require you to prove $(p)' \geq 0$ in the postcondition instead.

3.3 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:

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

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

Exercise 4:
Try to use dI to prove $u = 0 \land v = 0 \rightarrow [\alpha](u = 0 \land v = 0)$.
Answer: This will fail because the resulting premise would require proving $-v + \frac{1}{4}u(1 - u^2 - v^2) = 0 \land u + \frac{1}{4}v(1 - u^2 - v^2) = 0$ which is not true without any additional assumptions on u, v.

Exercise 5:
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 $u = 0 \land v = 0$ 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 advanced; if you are interested, Yong Kiam Tan is the person in the lab to talk to. 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 $u = 0 \land v = 0$ directly. If you have a postcondition that contains only equations (possibly with conjunctions), give it a try and let us know if you find that it fails to prove your property.

3.4 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 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$\succcurlyeq$ 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:

$$\text{(BC)} \quad \frac{Q, p = 0 \vdash [x' := f(x)](p)' > 0}{p \succcurlyeq 0 \vdash [\{x' = f(x) \,\&\, Q\}]p \succcurlyeq 0}$$

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.

3.5 Miscellaneous and Explore!

KeYmaera X has many more useful ODE tactics. For example, `diffUnpackEvolDomain` allows you to assume the evolution domain constraint is initially true for an ODE. As another example, the relatively new tactic `odeInvC` is an especially powerful ODE workhorse tactic. But there are many more! You should feel free to explore KeYmaera X's tactics yourself, especially if you are doing a course project involving modeling and complicated dynamics. You can find them in the same way that we found the `dbx` tactic in Section 3.2.1 of these notes.

4 Proving Hybrid Games

Let's revisit a familiar question. Let

$$\alpha_1 \equiv \{x' = v, v' = a, t' = 1 \,\&\, t \leq T\}$$
$$\alpha_2 \equiv \{x' = v, v' = -B, t' = 1 \,\&\, v \geq 0\}$$

and consider the following game:

$$\alpha \equiv t := 0; a := *; ?(0 \leq a \wedge a \leq A); T := *^d; ?(T > 0)^d; (\alpha_1 \cup \alpha_2)$$

This means that Angel picks an acceleration between 0 and A, Demon picks a positive timestep, and Angel gets to accelerate with acceleration a for time T or apply the brakes at $-B$ indefinitely until stopping. Angel gets to decide how many times to play.

Demon has a winning strategy, i.e. the following is valid:

$$A > 0 \wedge B > 0 \wedge v = 0 \wedge x < station \rightarrow [\alpha^*]x < station$$

Note: We discussed the invariant and went through the KeYmaera X proof in class, but will not be releasing the solution/code.

5 General tips (may be useful on lab 4!)

5.1 Creating the Model

Here are some simple tips for creating your model. You are likely familiar with all of these, but you may wish to review them before Lab 4 and the project.

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 $[]\wedge$ 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. Consider if there are quantities which may be helpful that you do not already have variables for.

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?

5.2 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, focus on the branches that you are most unsure about.

   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, and what case it corresponds to in the system.

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. You may be surprised at the speed-up that you can achieve by doing this. Sometimes, even manually splitting disjunctions can help to speed things up.