15-424/15-624/15-824 Recitation 3
Manual Proofs and Tactical Proofs in KeYmaera X

1 Quick notes on assignment 1

These were some fairly common mistakes that are worth pointing out:

1. Don't forget that π is not a piece of syntax, so while $x := 3.14159$ is a program, $x := \pi$ is not a program!

2. The functions cos or sin are also not in the syntax. That's because it's hard to reason about them symbolically, like we do in differential dynamic logic. That doesn't mean they can't be represented! As you know, the differential equation $x' = -y, y' = x$ gives us those functions. And we know how to reason symbolically about ODEs! The critical realization is that while we are talking about cos and sin, we understand them through their definition according to the differential equation. Thus, when reasoning symbolically about cos and sin, we can only use the fact that $x' = -y, y' = x$, nothing else.

3. Remember that $\langle\alpha\rangle\phi$ means *some* execution of α takes us to a state where ϕ is true. The problem that asked for an α that does not mention n and also makes $\langle\alpha\rangle x = n$ true was a tricky one! But it's not so bad if you remember the meaning of the diamonds – just continuously evolve backward and then forward. There are lots of ways of doing that which do not result in states where $x = n$, but that doesn't matter – we just need one state!

2 Manual Proofs and Tactical Proofs

Let's consider a simple program that inverts negative values and increments non-negative values:

$$[?(x < 0); x := -x \ \cup \ ?(x \geq 0); x := x + 1]x > 0$$

Now that we've seen proof rules in lecture, we can write down a proof of this property on paper:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{*}{x<0\vdash -x>0}\,\text{FOL}_{\mathbb{R}}\ \texttt{QE}}{x<0\vdash [x:=-x]x>0}\,[:=]\ \texttt{assignb(1)}}{\vdash x<0\to[x:=-x]x>0}\,\to\text{R}\ \texttt{implyR(1)}}{\vdash [?(x<0)][x:=-x]x>0}\,[?]\ \texttt{testb(1)}}{\vdash [?(x<0);x:=-x]x>0}\,[;]\ \texttt{composeb(1)}
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{*}{x\geq 0\vdash x+1>0}\,\text{FOL}_{\mathbb{R}}\ \texttt{QE}}{x\geq 0\vdash [x:=x+1]x>0}\,[:=]\ \texttt{assignb(1)}}{\vdash x\geq 0\to[x:=x+1]x>0}\,\to\text{R}\ \texttt{implyR(1)}}{\vdash [?(x\geq 0)][x:=x+1]x>0}\,[?]\ \texttt{testb(1)}}{\vdash [?(x\geq 0);x:=x+1]x>0}\,[;]\ \texttt{composeb(1)}
$$

$$
\dfrac{\vdash [?(x<0);x:=-x]x>0\wedge[?(x\geq 0);x:=x+1]x>0}{\vdash [?(x<0);x:=-x\ \cup\ ?(x\geq 0);x:=x+1]x>0}\,\wedge R\ \texttt{andR(1)}\ \ [++]\ \texttt{choiceb(1)}
$$

You'll notice this proof looks slightly different than the ones we've done in class and in recitation. Each step in the proof is labeled twice. The first label is the name of the proof step that we've used, and the second is the name of the tactic that corresponds to that step[1].

Notice that almost all of the tactics are *applied* to a positional indiciator; in this case, the number `1`. These numbers indicate positions within the sequent; positive numbers indicate formulas to the left of the turnstile ($\vdash$) while negative numbers indicate positions to the right of the turnstile. In this case, we never have more than one formula in the succedent[2]

Okay, let's translate our dℒ formula into a KeYmaera X `.kyx` file and then try to prove this property!

Step 1: Translate the formula into a `.kyx` file

```
ProgramVariables.
  R x.
End.
Problem.
 [
  ?(x < 0); x := -x;
 ++
  ?(x >= 0); x := x + 1;
 ] x > 0
End.
```

Step 2: Upload to KeYmaera X and start a new proof.

Step 3: Prove the mode! At this point we have three options:

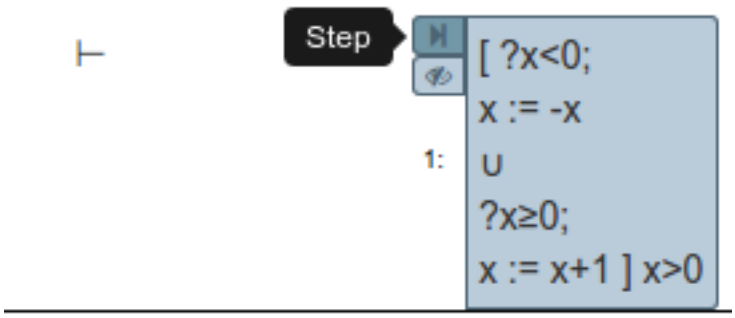
[1]Recall from the Lab 0 handout that **tactics** are scripts that direct KeYmaera X's proof construction facilities.

[2]Recall from lecture that sequents $\Gamma_{-n},\ldots,\Gamma_{-1}\vdash\Delta_1\ldots\Delta_n$ have an antecedent (the Γ's) and a succedent (the Δ's), both of which are just lists of dℒ formulas. The $\vdash$ is called a turnstile and means "assuming ALL the things on the left (|), one of the things on the right is true (−)."

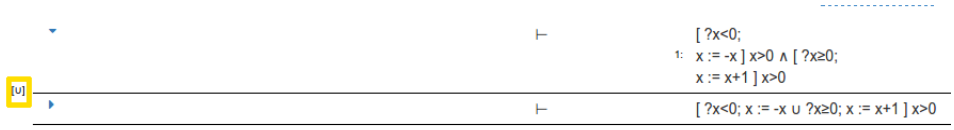
- Run the Auto tactic (a.k.a. `master`). For very simple models this will finish the proof in a reasonable amount of time. But by lab 3 `master` isn't going to work anymore, so we better get practice with the other methods of proving things in KeYmaera X!
- Manual proofs: KeYmaera X has a nice user interface, so we can point and click on formulas and rules to build up the proof manually like we did on the white board
- Tactical proofs: When manual proofs become a little bit too big, we can write down tactic programs instead of spending all day and night pointing and clicking every time we make a small chagne to our model. But we better learn how to do that in a small setting first!

2.1 Manual Proof

To prove a property manually, just click on the formula and KeYmaera X will take a single step of the `master` / auto tactic:

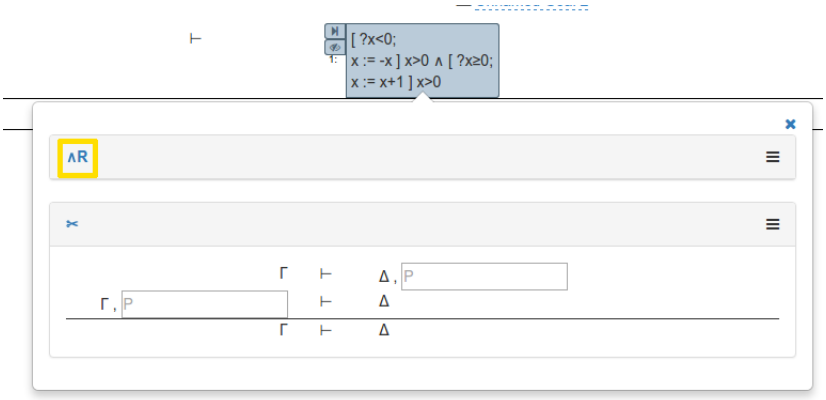


After clicking on the step button we'll have a new goal to prove:

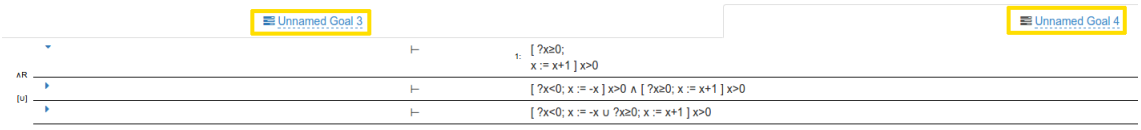


Notice that KeYmaera X decided to do exactly what we did in our manual proof – apply the [∪] proof step to the formula in the succedent (a.k.a. execute the `choiceb(1)` tactic). You can tell this is what the step tactic did by observing the label next to the horizonal line in the proof.

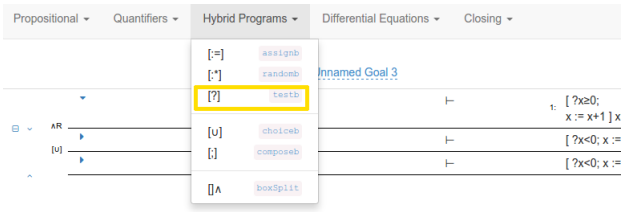
The top-level operator on our new formula is a ∧, so let's split the proof into two subgoals by running the `step(1)` tactic or by highlighting the formula and clicking the step button or by right-clicking on the formula and clicking on the name of the ∧R rule:



Now we have two new subgoals – one for each side of the conjunction. In the sequent proof above this was represented by two new goals on top of a single old goal. To keep things readable, KeYmaera X places each subgoal in a separate tab:



And now we can continue on with the proof in each branch. One quick note – the KeYmaaera X developers have placed a lot of the available proof steps / tactics in the menus. So we can search the hybridp program menu and find the [?] tactic we know from the paper proof we want to use next:



Convienantly, the menu options also indicate the name of the tactic that's applied when the UI executes the proof step.

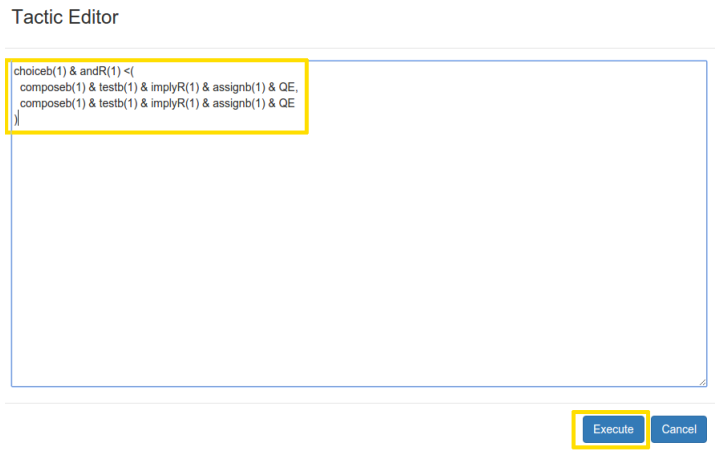
2.2 Tactics

UI proofs are fun and easy for small models, but for larger models and for models in flux, we might want to write down our proof as a program so that small changes to the model can be accompanied by small changes to the proof. The tactic for this model is:

```
choiceb(1) & andR(1) <(
  composeb(1) & testb(1) & implyR(1) & assignb(1) & QE,
  composeb(1) & testb(1) & implyR(1) & assignb(1) & QE
)
```

The `&` symbols mean "do the left and then the right (but only if the left succeded" and the $<$ symbol is used whenever the proof branches from on goal to two (or more) goals. Notice that we do the exact same thing on both branches of this proof, and instead of re-clicking through the entire proof all we have to do is copy/paste! That's super useful if the proofs for each branch are hundreds of tactics long instead of just 5 tactics long! For more details on tactics combinators (&, $<$, etc.) see the KeYmaera X cheat sheet http://www.ls.cs.cmu.edu/KeYmaeraX/KeYmaeraX-sheet.pdf.

Let's save this proof as `discrete.kyt` for later. If you want to execute this tactic, just open the tactic editor in the KeYmaera X proof UI, copy/paste, and click execute:



3 More Tactics Practice

Consider the model:

$$\equiv y = y_0 \land x > 0 \rightarrow [\{y' = x\}]y \geq y_0$$

Convert this formula into a `.kyx` file, load it into KeYmaera X, and start a new proof. The

proof is pretty easy – move the assumption into the antecedent, solve the ODE, the run the decision procedure for $\mathbb{R}$ (a.k.a. `QE`):

```
implyR(1) & diffSolve(1) & QE
```

4 Combining Tactical Proofs

Let's now consider a formula that combines the discrete and continuous systems we've just completed proofs for:

$$y = y_0 \rightarrow [\{?(x < 0); x := -x \cup ?(x \geq 0); x := x + 1\}; \{y' = x\}]y \geq y_0$$

Remember the post-condition of our discrete system was $x > 0$, which is convienantly the only assumption we're missing such that following the ODE results in our desired post-condition! So let's splice our two tactical proofs together into a proof for the combined discrete/continuous system:

```
implyR(1) & composeb(1) & choiceb(1) & andR(1) <(
  composeb(1) & testb(1) & implyR(1) & assignb(1) & diffSolve(1) & QE,
  composeb(1) & testb(1) & implyR(1) & assignb(1) & diffSolve(1) & QE
)
```

Note we needed an extra `composeb(1)` to handle the semi-colon between the discrete program and the continuous program, but otherwise we've basically just appended the continuous tactic onto the end of each of the branches in our discrete tactic.

4.1 Cut - bring your knife! haha so fun

It is a truth universally acknowledged that computers are in want of intelligence! Sometimes, they need guidance when we tell them to find proofs for a given formula. They get easily confused by formulas that say what we mean in strange ways.

To illustrate, let's start working with the following:

$$\big((x - y)^2 \leq 0 \land \phi(x)\big) \rightarrow \phi(y)$$

The proof starts easily enough...

$$\to_R \dfrac{\wedge_L \dfrac{\dfrac{?}{(x-y)^2 \leq 0, \phi(x) \vdash \phi(y)}}{(x-y)^2 \leq 0 \wedge \phi(x) \vdash \phi(y)}}{\vdash ((x-y)^2 \leq 0 \wedge \phi(x)) \to \phi(y)}$$

But now what? well, if ϕ is a FOL formula and it's small and simple enough, it might be possible to get KeYmaera X and its QE procedure to do the work for us.

Unfortunately, that's often not the case. Formulas are long, complicated, and involve lots of variables, and so KeYmaera X is going to really struggle.

What can we do? Looking at the assumption $(x-y)^2 \leq 0$, we can draw some conclusions. We know that $(x-y)^2$ as to be non-negative, which with our assumption gives us $(x-y)^2 = 0$. in turn, this tells us that $x = y$. But wait! If they are equal, then the formulas $\phi(x)$ and $\phi(y)$ are equivalent, making them trivial to prove!

That is a lot easier for KeYmaera X to realise, so we are going to cut in the fact that $x = y$. Furthermore, we will also use the weakening or hiding rules to make sure KeYmaera X focuses on the right subproblems of our proof, instead of worrying with extraneous stuff.

$$\to_R \dfrac{\wedge_L \dfrac{\dfrac{\text{cut} \dfrac{W_L \dfrac{W_R \dfrac{\text{QE} \dfrac{*}{(x-y)^2 \leq 0 \vdash x = y}}{(x-y)^2 \leq 0 \vdash x = y, \phi(y)}}{(x-y)^2 \leq 0, \phi(x) \vdash x = y, \phi(y)} \quad W_L \dfrac{\dfrac{*}{x = y, \phi(x) \vdash \phi(y)}}{(x-y)^2 \leq 0, x = y, \phi(x) \vdash \phi(y)}}{(x-y)^2 \leq 0, \phi(x) \vdash \phi(y)}}{(x-y)^2 \leq 0 \wedge \phi(x) \vdash \phi(y)}}{\vdash ((x-y)^2 \leq 0 \wedge \phi(x)) \to \phi(y)}$$

This practice of weakening/hiding and then applying QE is extremely helpful to save time in your proof attempts!