Lab 1: Charging Station
15-424/15-624 Foundations of Cyber-Physical Systems
Course TAs: João Martins (jmartins@cs), Annika Peterson (apeterso@andrew)

Betabot Due Date: 9/10/14, worth 20 points
Veribot Due Date: 9/17/14, worth 70 points

1. **Interacting with KeYmaera**

   Recall formula $\forall x(x > 0 \wedge x < 1)$ from lab 0. You will manually interact with this formula now, by trying to prove it. Since the formula is false you won't be able to, but you will see how doing so affects the proof tree.

   Highlight the $\forall x$, and use the "$\forall$ r all right" rule. This gets rid of the $\forall$ symbol, leaving you only with something like $x_1 > 0 \wedge x_1 < 1$. Notice how a new node appeared under the Proof Tree panel in the lower left, and it has the name of the rule you just applied. The proof tree keeps track of everything you are doing to the original formula!

   The formula under the Current Goal panel is now a conjunction, $\wedge$. To prove a conjunction, both conjuncts need to be true. In this case, for $x_1 > 0 \wedge x_1 < 1$ to be true, you'd need to prove both $x_1 > 0$ as well as $x_1 < 1$. Hover your mouse over the $\wedge$ symbol, and select "$\wedge$ r and right". This rule applies the reasoning we just made, where to prove $x_1 > 0 \wedge x_1 < 1$, you must prove each conjunct separately.

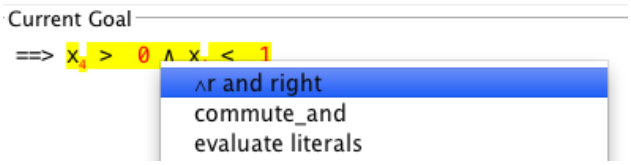

Figure 1: Applying the $\wedge$ r and right rule.

   Look at the Proof Tree panel! Two cases were added, one for each conjunct. The tree is now *actually* a tree, not just a sequence of nodes! Yay!

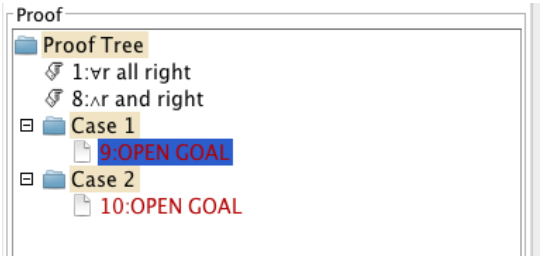

Figure 2: The proof tree is actually a tree now!

   Since you haven't proven either of the cases, they are called *open goals*. Select each open goal, and notice how KeYmaera updates the Current Goal panel. Each goal should represent one of the conjuncts! In a bigger proof, you'd now continue to try to prove each conjunct separately, growing that part of the tree.

   If you go back to your proof for lab0.key, in the previous lab, you'll be able to see just how quickly these trees can grow!

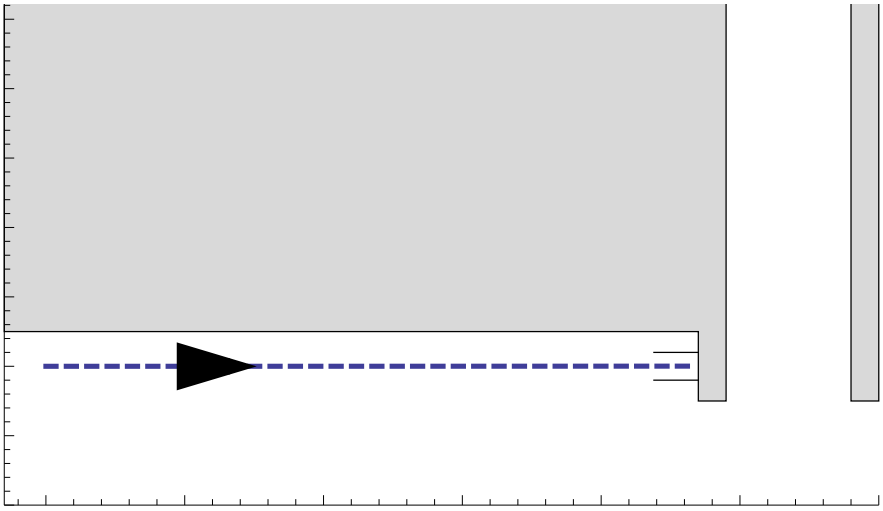

Figure 3: Charging station & wall.

2. **Autobots, Roll Out**

   As the n00b in your robotics group, the senior members gave you the broken robot. Its steering is jammed, so it can only move in a straight line and control acceleration.

   1. Fill in the missing continuous dynamics in the hybrid program below that will model your robot accelerating in a straight line, with position `pos`, velocity `vel`, and acceleration `acc`.

   2. Fill in the safety condition that will ensure the velocity of the robot is never negative. Save this file as `L1Q1_name.key` (where name is replaced with your first and last name).

   3. Use KeYmaera to prove the velocity of the robot is always positive. Save the resulting proof file as `L1Q1_name.proof`. All of the tasks in this lab should prove automatically - but keep in mind that proving manually will significantly help be necessary in the later labs and it helps if you start early!

```
(vel = 0 & acc >= 0)      /* Requires */
->
\[{--------}\]           /* Continuous dynamics */
(--------)                /* Ensures (safety condition) */
```

3. **Charging Station, Single Control**

   Oh no! Your robot's battery is almost dead! Luckily the robot is already on a straight trajectory to a wall charging station and has positive velocity. With its remaining power, your robot has just *one chance* to engage the brakes properly to bring the robot to a stop at the right place.

   - If the robot brakes too hard, it will not make it to the charging station and will have to wait for a human to plug it in. Running out of power is *inefficient*.
   - If the robot doesn't brake hard enough to stop at the charging station, it will dent the wall behind the station. When building manager Jim Skees finds out, he'll have your robot banned from the building! Running into walls is *unsafe*.

   Hints: think carefully about all the variables that the acceleration/braking should depend on. If you find you can't prove the property, it could be that you missed something and your property is falsifiable. Don't forget you learned how to find counter-examples in lab0!

1. Specify the missing safety and efficiency requirements that the hybrid program below should ensure. Then fill in the missing control and continuous dynamics. Save the resulting file as `L1Q2_name.key`.

2. Use KeYmaera to prove that the hybrid program is safe and efficient. Save the resulting proof as `L1Q2_name.proof`.

3. **Question:** What is the evolution domain for the continuous dynamics in this hybrid program? Why is it necessary? (Your response to this question does not need to be submitted.)

```
(pos < station & vel > 0) /* Requires */
->
\[
    acc := _________;      /* Assign a safe acceleration. */
    {____ & vel >= 0}      /* Use continuous dynamics from part 1. */
\]
(____________              /* Ensures (safety condition) */
 &____________)            /* Ensures (efficiency condition) */
```

4. **Charging Station, Double Control**

In part 2, you are always able to coast the robot all the way to the charging station, since it is already moving. But what if the robot is stopped? In this question, the goals are the same as in part 2; however, the robot starts with zero velocity. You have two chances to control the robot, first to get it moving by accelerating, and then to bring it to a stop at the right point by braking.

The robot also has a time limit `T` on how long it can accelerate before exhausting the remaining battery life. Once the brakes are engaged, they will stay engaged until the robot comes to a complete stop.

1. Save the filled in formula below as `L1Q3_name.key`.

2. Prove the formula and save your proof file as (`L1Q3_name.proof`).

3. **Question:** What is your efficiency condition? Is it different from part 2, why or why not? (Your response to this question does not need to be submitted.)

```
(pos < station & vel = 0 & 0 < T)        /* Requires */
->
\[
    t := 0;
    acc := _________;                    /* Assign a safe acceleration. */
    {____, t' = 1 & vel >= 0 & t <= T};
    ?(t > 0);
    acc := _________;                    /* Assign a safe deceleration. */
    {____, t' = 1 & vel >= 0}
\]
(____________                            /* Ensures (safety condition) */
 &____________)                          /* Ensures (efficiency condition)*/
```

5. **Submission Checklist**

(a) **BetaBots:** submit a zip file on autolab containing your preliminary .key files for each of the tasks. This will enable us to give you feedback halfway through the assignment, so that you don't get stuck! If you want, you can include some *small* comments about your approach and questions you might have. *Due Wed 09/10.*

(b) **Final submission:** the final submission works the same way, but you must include your final model in a .key file as well as completed proof in a .proof file. You can save a proof while it is in progress or when it is completed by clicking on File → Save in KeYmaera. To receive credit,

proofs must be complete (i.e. the "Property Proved" window has appeared, so all branches are closed and there are no remaining goals). *Due Wed 09/17*

Use the provided templates, and *do not forget to fill in the section at the top.* It gives us important information when grading your submission!

The zip file should contain:

- `L1Q1_name.key`
- `L1Q1_name.proof`
- `L1Q2_name.key`
- `L1Q2_name.proof`
- `L1Q3_name.key`
- `L1Q3_name.proof`