Lab 3: Robots on Racetracks
15-424/15-624/15-824 Logical Foundations of Cyber-Physical Systems
TA: Katherine Cordwell (kcordwel@cs.cmu.edu)

Checkpoint Due Date: **Friday, October 11th BEFORE 12:00 noon with NO late days**, worth 10 points
Betabot Due Date: **Tuesday, October 15th BEFORE 12:00 noon with NO late days**, worth 20 points
Veribot Due Date: Tuesday, October 22nd, 11:59PM (2 late days, max 6 per semester), worth 70 points

Lab Resources: `https://lfcps.org/course/lfcps19/lab3.zip`

1 Checkpoint: Differential Invariants, Cuts, and Weakening

In this lab, you will utilize two essential techniques:

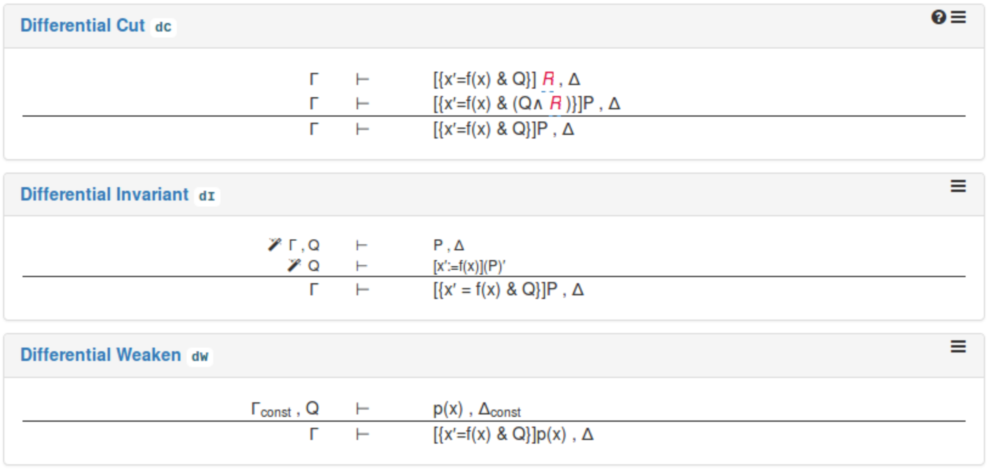
1. Reasoning about ODEs with differential invariants when solving them is difficult.
2. Reasoning about arithmetic with lower bounds when exact solutions are difficult.

By putting these techniques together, we will be able to verify a robot on a racetrack!

The purpose of this checkpoint submission is to ensure that you have the necessary practice with ODE tactics in KeYmaera X for the rest of this lab. Your task is to prove the following dL formula using KeYmaera X:

$$x \geq y \land y \geq z \land z \geq 0 \rightarrow [x' = y, y' = z, z' = x^2]x \geq 0$$

Special restriction: we want to check your thinking, not that of KeYmaera X. You are not allowed to use master/auto or the automatic ODE tactic for this question. You will receive no points if your proof uses any of these. Instead, make use of the Differential Cut, Differential Invariant, and/or Differential Weakening rules that you have seen in class (use the Right-Click context menu):



Differential Cut dc

Γ ⊢ [{x'=f(x) & Q}] R , Δ
Γ ⊢ [{x'=f(x) & (Q∧ R)}]P , Δ
Γ ⊢ [{x'=f(x) & Q}]P , Δ

Differential Invariant dI

Γ , Q ⊢ P , Δ
Q ⊢ [x':=f(x)](P)'
Γ ⊢ [{x' = f(x) & Q}]P , Δ

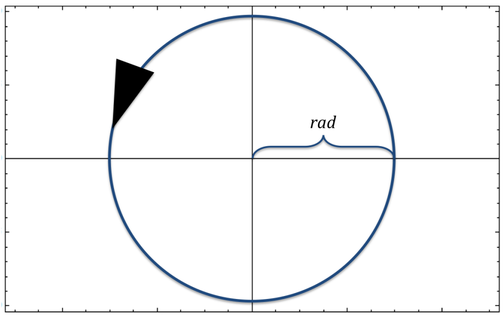
Differential Weaken dW

Γ_{const} , Q ⊢ p(x) , Δ_{const}
Γ ⊢ [{x'=f(x) & Q}]p(x) , Δ

1. (Checkpoint). Prove the formula and save the resulting archive as L3Q1.kyx.

   Note that this proof is due for the Checkpoint submission.

   If you are working in pairs, both group members should attempt this question to get familiar with proving properties of ODEs in KeYmaera X. *However, only one of you needs to submit on Autolab.*



For the remaining questions, you will design controllers to drive your robots around a circular racetrack centered at the origin and with radius *rad*.

The properties you prove are safety properties (e.g., stay on the track and do not hit the obstacle), but you should design your controller to satisfy a basic efficiency condition as well (i.e., it should eventually come near the obstacle). So, for example, a robot that eventually comes near the obstacle slower than necessary is fine, but a robot that never ever comes close to the obstacle because it just stays still is useless.

Since using polar coordinates would make this lab trivial (and cause you to fail Labs 3 and 4), you are **required to use Cartesian coordinates**. Cartesian coordinates will also generalize better to Lab 4.

2 Keep The Robot on the Racetrack

The goal of this problem is to focus on getting your continuous physics model correct. Your robot must be able to drive **counterclockwise** on the track **without leaving it**. In other words, the robot must always be on the circle with radius *rad* centered at the origin. The racetrack is completely empty, so the robot may choose to travel at any speed v. Note that in Cartesian coordinates, v means *linear* velocity/ground speed, and not angular velocity.

Design a hybrid program to keep your robot on the track. This initial hybrid program has no controller! Your robot should move at a *non-negative*, constant speed with *positive*, constant radius.

Create a formula which, if proved, would show that your robot never drifts off the circle. You will need to determine appropriate initial conditions to make this property provable; however, try to prove the strongest property you can by making the initial conditions as general as possible.

1. (Betabot). Model the situation described above and save your model as L3Q2.kyx.

2. (Betabot). **Question:** Solve the differential equation you used to model the physical dynamics of the system from initial point $(0, rad)$. Could you rewrite an equivalent hybrid program using this solution and the assignment operator instead of a continuous evolution? Why or why not? Submit a *brief* answer in lab3.txt. Also *briefly* explain the intuition behind your differential equations.

   Hint: If you aren't confident about your physical dynamics, you can use your solution to check it! For example, substitute your solution into an expression for the linear speed to check that it is indeed v.

3. (Veribot). Use KeYmaera X to prove that your robot stays on the track **using differential invariants**.[1] Submit your proof archive as L3Q2.kyx.

4. (Veribot). **Question:** How does the arithmetic involved in your differential invariance proof compare to that of the solution to the differential equations in your earlier answer? Are there properties about the motion of your robot that you can deduce from the solution but not from the differential invariant? Submit *brief* answers (2-4 sentences) in lab3.txt.

[1] You can use `master` to prove this automatically, but make sure you try the differential invariant calculations yourselves.

3 Keep An Accelerating Robot on the Loop with a Loop

Augment your model so the robot can change its speed (by accelerating) and use KeYmaera X to prove that your controller keeps the robot on the track. Your HP must include a loop so that the acceleration controller may execute more than once.

Since there are no obstacles to avoid yet, you can set the acceleration to whatever you want. The goal is just to make sure you can still model the physics correctly when you add acceleration. This should not be much harder than the previous problem, but is easy to get wrong if you are not careful.

1. (Betabot). Submit your augmented model as L3Q3.kyx.
2. (Veribot). Submit your completed proof archive as L3Q3.kyx.

4 Lower Bounds for Circular Motion

Now that we are sufficiently warmed up, our goal is to verify that the robot will safely stop before hitting an obstacle on the circular track. That will be the subject of Question 5.

For this question, you will devise and discuss a proof approach to help you for that question. All parts of this question should be answered in `lab3.txt` for the **Betabot** submission.

1. In Lab 2, you could brake safely by computing the total distance traveled by the controlled car in a given time. Briefly explain why this approach is not feasible for a robot moving in a circle.
2. An alternative approach is to establish a lower bound l on the remaining distance d which your robot has to move *on the circular path* before hitting the obstacle. In particular, with this lower bound, it is sufficient to prove $0 < l$ in order to establish safety for your robot.

   Choose and write down a term l which is a lower bound on the remaining distance to travel on the circular path. You will want l to be simple so that your proof will also be simple.

   Hint: You may assume that the robot is currently at coordinates (x, y), and the obstacle is at (ox, oy).
3. Because you are using a lower bound, the resulting controller will not be perfectly efficient: it will sometimes accelerate less than the maximum safe amount. Discuss any inefficiencies in the resulting controller: under what circumstances is it most inefficient? How practical or impractical are these inefficiencies? Can you think of a way to reduce the inefficiencies by choosing a more complicated lower bound l?

   Note: You should still do the proof in the next question using a very simple l even if it is inefficient. Only attempt to prove a more efficient controller once this proof is complete, because you may find that the proof is quite challenging even for simple l.

5 Racetrack Debris (circular-path static obstacle avoidance)

In this problem you will combine everything you learned in the previous problems to verify safety of a robot driving on a circular track. A frustrated student left their computer on the track while waiting for KeYmaera X to find a safety proof. Your goal is to develop a robot that moves counterclockwise around the track but stops before running over the student's computer (because you feel their pain).

Of course, you cannot start your robot driving at high speed right next to the computer and expect the system to still be safe, so you will need to define some initial conditions on the distance between your robot and the relative position of the computer. If these initial conditions are satisfied, your controller must be able to bring your robot to a stop before it hits the computer. The bounds on acceleration and braking for your controlled robot are $A > 0$ and $-B < 0$ respectively. **Your controller must be time-triggered.**

The computer is at the leftmost position on the track because that is closest to the nearest place where the student could get coffee. Following the previous problem, **you should start with a very conservative controller with very loose lower bounds**. You should only attempt to make the controller less conservative once it works, and you **should not** expect to make your bounds exact.

1. (Betabot). Design a hybrid program to keep your robot on the track and stop before the position of the obstacle. Create a formula which, if proved, would guarantee that your time-triggered controller satisfies these safety properties. Submit your model as L3Q5.kyx.

2. (Betabot). Briefly explain your control strategy in lab3.txt.

3. (Veribot). Use KeYmaera X to prove the safety property and submit your proof archive as L3Q5.kyx.

4. (Veribot). Explain any tricks used in your proof in lab3.txt. If you did not manage to complete your proof, briefly outline where you got stuck and briefly speculate why e.g., do you believe that your controller is correct but you could not figure out how to prove it?

5. (Veribot). Suppose you have proven this controller to be safe. Based on your model and your proof, should we believe that your controller would work if the obstacle was instead dropped at an arbitrary spot on the circle rather than just its leftmost point? In lab3.txt write whether the controller ought to work and briefly argue why or why not.

   Hint: Obviously, your controller would need minor tweaks, e.g., changing the location of the obstacle. This question is asking whether you would need to change your general control strategy.

Meta-arguments like in Q5.5 are not formally proved in KeYmaera X. For extra credit (*only for Veribot*), do not assume that the student's computer was dropped at the leftmost position of the racetrack. The befuddled student might have dropped it anywhere on the racetrack!

6. (Veribot). **Bonus:** Update your model to satisfy this requirement and prove it safe in KeYmaera X. Submit the resulting file as L3Q5_bonus.kyx.

6 Tips and Tricks

In this lab (and in the next one), you might run into an issue where "master" or "QE" fail to prove very simple goals in a reasonable amount of time. In this case, it is likely that your Mathematica has gotten stuck. From the drop down user interface, try selecting Tools → Restart. If that does not solve the problem, try restarting KeYmaera X.

Also, if you are doing QE on something very complicated and it doesn't close but you are sure it is right, try to manually simplify things (e.g. hide unnecessary assumptions). This can really help out Mathematica.

7 Submission Checklist

If you are working with a partner, then you must submit a file called andrewids.txt containing both of your Andrew IDs. *To make the grading infrastructure happy, please put them on a single line separated by a space, e.g.*:

```
kcordwel aplatzer
```

Make sure you submit this file when working in a group. Otherwise, one of you will not get credit because there is no record of your submission. Additionally, ONLY one of you should submit on Autolab so that we do not end up grading your submissions twice.

For all submissions, remember to check the Autograder's output on Autolab to ensure that your files were submitted in the right format, parse correctly, etc. If you are working in a group, please also ensure that the Autograder correctly reports your Andrew IDs in its output, e.g.:

```
==> Group Andrew IDs: kcordwel, aplatzer
```

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!

1. **Checkpoint submission**. Submit a zip file on Autolab containing your proof archive for L3Q1.

   The Checkpoint zip file should contain:

   - `L3Q1.kyx`
   - `andrewids.txt` (only if working in pairs)

2. **Initial submission (Betabot)**. Submit a zip file on Autolab containing your preliminary .kyx files for each of the tasks as well as the Betabot discussion file. This will enable us to give you feedback halfway through the assignment, so that you do not get stuck! If you want, you can include some *small* comments about your approach and questions you might have.

   While a proof is not required at the Betabot due date, you should, nevertheless, strive to get the model and controller correct, because that will give you a better basis for the Veribot that you will be proving. It will also result in a higher Betabot grade and allow us to provide more useful feedback.

   The Betabot zip file should contain:

   - `L3Q2.kyx`
   - `L3Q3.kyx`
   - `L3Q5.kyx`
   - `lab3.txt` (containing answers to Questions 2.2, 4.1-4.3, 5.2)
   - `andrewids.txt` (only if working in pairs)

3. **Final submission (Veribot)**. The final submission works the same way, except you submit `.kyx` files (which contain both the model and the proof) and your Veribot discussion file. To receive full credit, proofs must be complete (i.e., a successful "Proof Result" window appears after running the tactic you have submitted).

   The Autograder for this lab (and all remaining labs) will not check that your proof works automatically because the models and proofs are a lot more complicated. Thus, you will not see any `==> Succeeded at proving ...` lines in its output.

   The Veribot zip file should contain:

   - `L3Q2.kyx`
   - `L3Q3.kyx`
   - `L3Q5.kyx`
   - `L3Q5_bonus.kyx` (only for bonus credit)
   - `lab3.txt` (containing answers to Questions 2.4, 5.4, 5.5)
   - `andrewids.txt` (only if working in pairs).