

**Lab 3: Robots on Racetracks**  
**15-424/15-624/15-824 Foundations of Cyber-Physical Systems**  
**Course TAs: Nathan Fulton (nathanfu@cs), Anastassia Kornilova (akornilo@andrew)**

Betabot Due Date: 2/26/16, worth 20 points  
Checkpoint Due Date: 3/1/16, worth 10 points  
Veribot Due Date: 3/03/16, worth 70 points

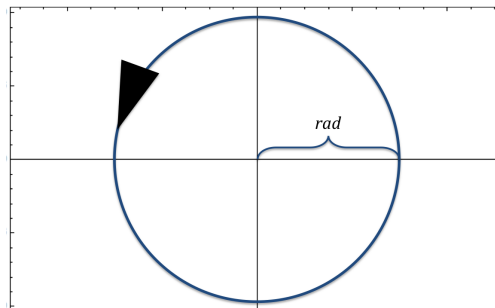
**1. Practice with Diff Cuts, Invariants, and Weakening in KeYmaera X**

Prove the following  $d\mathcal{L}$  formula using KeYmaera X. You **may not** use master/auto! You also may not use the ODE Solve rule. Instead, use the Differential Invariant, Differential Cut, and Differential Weakening rules. Save your resulting model and tactic as L3Q1\_andrewid.kyx and L3Q1\_andrewid.kyt.

$$\vdash (x \geq 0 \wedge y \geq 0 \wedge z \geq 0) \rightarrow [x' = y, y' = z, z' = x^2]x \geq 0$$

The model is due for the BetaBot portion of the lab and the tactic is due on the Checkpoint due date.

In questions 2 and 3, you will design controllers to drive your robots around a circular racetrack centered at the origin and with radius  $rad$ . In both questions, you will design a controller that satisfies the safety specifications given. The properties you prove are safety properties (e.g. stay on the track and don't hit the obstacle), but you should also keep in mind the efficiency of your controller (e.g. can your controlled robot reach the obstacle). So, for example, a robot that just stays still will satisfy all the safety properties, but would be rather useless. To prepare you for lab 4, we **strongly encourage you to use cartesian coordinates** (as opposed to polar coordinates) for this problem.



**2. Lap time (empty racetrack)**

In this question, your robot should be able to drive *counter-clockwise* on the track without leaving it. In other words, the robot should 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. This should be *linear* velocity, not angular. Your HP should include a loop so that the controller may execute more than once.

- (a) Design a hybrid program to keep your robot on the track, then create a formula which, if proved, would guarantee that your controller never causes the robot to drift 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.
- (b) Use KeYmaera X to prove that your controller keeps the robot on the track. Submit your model and tactic as L3Q2\_andrewid.kyx and L3Q2\_andrewid.kyt.

- (c) **Question:** Solve the differential equation you used to model the physical dynamics of the system. 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 `L3_andrewid.txt`. Also *briefly* explain the intuition behind your differential equations (**BetaBots**).

*Hint:* Since the next question is more complicated, try allowing your robot to accelerate, and proving it still remains on the racetrack. This ensures that he does not spiral away from the racetrack.

### 3. Racetrack Debris (static obstacle avoidance)

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 counter-clockwise around the track but stops before running over the student's computer (because you feel their pain). Of course, you can't start your robot at 100 mph just feet behind the computer and expect the system to still be safe, so you may 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 should 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  $-B < 0$ . *Your controller should be time-triggered.*

The computer is at the leftmost position on the track because that's closest to the nearest place where the student could get coffee. **We strongly suggest you start with very conservative controllers that you can prove**, and only then try to make them more efficient.

- 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 controller satisfies these safety properties.
- Use KeYmaera X to prove the safety properties. Submit the `.kx` file you proved and the tactic as `L3Q3_andrewid.kyx` and `L3Q3_andrewid.kyt`.
- In `L3_andrewid.txt`, briefly explain your control decision (**Betabots**) and any tricks used in your proof (**Veribots**).
- (**Veribots**) Suppose you've proven this controller to be safe. Would it work on a circle in which the obstacle was dropped at an arbitrary spot? How and why? Write your answer in `L3_andrewid.txt`.

### 4. Racetrack Debris (Extra credit, Veribots only)

Meta-arguments like in Q3.d aren't formally proved in KeYmaera X. For extra credit (*only for Veribots*), don't 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!

Submit `L3Q4_andrewid.kyx` and `L3Q4_andrewid.kyt` for the Veribots deadline. In `L3_andrewid.txt`, briefly explain how proving this generalisation was different from the previous question.

### 5. Submission Checklist

Submit a zip file on autolab (include all andrewid of members of your group).

- `L3Q1_andrewid.kyx`
- `L3Q1_andrewid.kyt` (Checkpoint only)
- `L3Q2_andrewid.kyx`
- `L3Q2_andrewid.kyt` (VeriBot only)
- `L3Q3_andrewid.kyx`
- `L3Q3_andrewid.kyt` (VeriBot only)
- `L3_andrewid.txt`

- `keymaerax_exported_andrewid.sqlite`
- `L3Q4_andrewid.kyx` (VeriBot only, extra credit)
- `L3Q4_andrewid.kyt` (VeriBot only, extra credit)