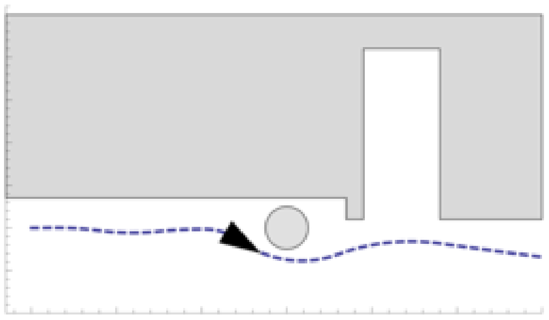
Lab 4 Robots in a Plane (Obstacle Avoidance)
15-424/15-624/15-824 Foundations of Cyber-Physical Systems
Course TA: Brandon Bohrer (bbohrer@cs)



Betabot Due Date: **Saturday 03/25/17 11:59pm**, worth 20 points
Veribot Due Date: **Thursday 03/30/17 11:59pm**, worth 80 points

In this lab you will design a controller which may move freely (non-deterministically) on a plane, rather than just around fixed to a circular track. The robot should be able to move anywhere, but it must always avoid a single, static (not moving) obstacle.

Modeling the free motion of a robot in 2D can be thought of as an extension of lab3, but now with discrete control of steering as well as acceleration. When steering is changed, you might think of it as the robot switching from one circular track to another, but the new track must be tangent to the old one at the position of the robot so that the robot can maintain its position, direction and velocity. The new track's radius and whether the track is on the left or the right of the robot (in other words, whether the robot is traveling clockwise or counter-clockwise around the track) may change at each discrete transition. To help you visualize what is happening, we have created this youtube video: `http://youtu.be/C_pyRQT6bBw`

To help you get started with this assignment, you may download a template for the kyx file here: `http://symbolaris.com/course/fcps17/lab4.zip` Ths template file contains variables that may be helpful, but you are of course always free to choose your own variable names (but please put a comment explaining what they do).

1. **Part 1** For the first portion of the lab, model the robot's movement without considering obstacles or time-triggers. The robot must always be on *some* circular track, but it may choose which circular track to follow at every control point.

   Constraints

   - The robot should have a non-deterministic controller. That is, if there are multiple valid control decisions, then your model should strive to allow as many of them as possible, because this makes the model more general.
   - You **may not** have discrete changes in the position, direction, or linear velocity of the robot (or any other variable which would implicitly cause such a discrete change), because that would not be faithful to physics.
   - You may (and should) discretely control the track radius and acceleration of the robot. You can switch between clockwise and counter-clockwise motion by switching between positive and negative radius.
   - Since you are now freely moving in 2D, your robot now has a shape, which can be over-approximated by a circle of radius $r > 0$.

- The robot should always have a non-zero turning radius (i.e. it can't spin in place).

Suggestions and hints

- Using *guarded* non-deterministic assignment like $\{var := *; ?(P(var))\}$ will be very useful in this problem. You may use it to choose the radius *trackr* of the conceptual "track" that the robot is moving on. Remember that in KeYmaera X syntax, $:= *$ must not have any spaces after the =.
- You will likely find the functions `max(x,y)` and `abs(x)` of use in this assignment. While they exist in KeYmaera X, they do not always work reliably, so we recommend you "implement" these functions using $\wedge$ and $\vee$ instead.

Turn-in Instructions

1.1 **(BetaBots)** Fill in the missing parts of the provided template to model the hybrid program and check that it is safe (i.e., stays on the currently chosen track). Submit this file as `L4Q1.kyx`.

1.2 **(Veribots)** Use KeYmaera X to prove that your hybrid program is safe. Submit the resulting proof as `L4Q1.kya`.

2. **Part 2** In this part of the lab you will add a static obstacle and a time-triggered controller to your model.

Constraints

- All constraints from Part 1 still apply.
- The robot should have a non-deterministic controller (i.e. it should be able to drive anywhere that does not cause it to come too close to the obstacle).
- The robot should be time-triggered.
- The obstacle also can be over-approximated as a circle of radius $obsr > 0$ (hint: you may still model your system using points, so long as a sufficient buffer is kept between the robot's point and the obstacle's point).

Suggestions and Hints

- All the hints from Part 1 are still relevant.
- We recommend that you use braking to enforce safety, and leave steering entirely non-deterministic. That is, your safety argument should not care about your steering decisions at all. As in Lab 3, this is tradeoff between efficiency and easy of proof. A controller that uses steering to provide safety would be more efficient, but much harder to verify due to the difficulty of reasoning about curved paths.
- You may also simplify your model to represent an infinitesimal point (x, y) for the position of the robot and point $(obsx, obsy)$ as the point of the obstacle **provided** you ensure that the two never get within a symbolic *buffer* distance of each other.

Turn-in Instructions

2.1 **(Betabots)** Fill in the missing parts of the provided template to model the hybrid program above and verify that it is safe. Submit this file as `L4Q2.kyx`.

2.2 **(Veribots)** Use KeYmaera X to prove that your hybrid program is safe. Submit the resulting proof as `L4Q2.kya`.

3. **Part 3** This exercise is identical to the above, except that the obstacle is now a *rogue-bot* which drives around with constant velocity rather than a stationary obstacle. Here are some extra rules for this problem:

   - The robot should never turn with a turning radius less than $minr > 0$ (i.e. it can't spin in place or turn too sharply).
   - The rogue-bot's shape can be over-approximated as a circle of radius $rogr > 0$ (hint: you may still model your system using points, so long as a sufficient buffer is kept between the robot's point and the rogue-bot's point).
   - The acceleration and braking of the robot are bounded by $A > 0$ and $B > 0$ respectively.
   - The rogue-bot maintains constant velocity $rogv \geq 0$.

   Turn-in Instructions

   3.1 **(Veribots)** Find a controller that you can convince yourselves in safe. You only have to submit the `L4Q3.kyx` file. **You do not need to prove this!**

   3.2 **Extra-credit:** prove it and submit the proof file `L4Q3.kya`

   3.3 **(Veribots) question:** What if your robot has a top speed that's less than *obsv*? What if instead of moving on an unbounded 2D plane your robot is constrained to a bounded space? In these cases, even if your robot is stopped, the obstacle could still hit you! Propose and discuss a few possible safety properties for these new scenarios which, if proved, would guarantee that your robot still exhibits reasonable behavior even in the presence of unreasonable obstacles. What are the pros and cons of each of your proposed safety properties? Submit your answer to this question in `L4.txt`.

4. **Submission checklist.** Submit a zip file on Autolab. The Makefile distributed with the assignment can make this for you. **Remember to include a file named andrewids.txt with your names in the following format, otherwise I will not know that you both submitted:**

   ```
   aplatzer bbohrer
   ```

   Additionally, when working in groups, *only one person should submit, otherwise I will end up grading you twice.*

   Test submission (Due Saturday 03/25):
   - `L4Q1.kyx`
   - `L4Q2.kyx`
   - `andrewids.txt (if working in pairs)`

   Final submission (Due Thursday 03/30):
   - `L4Q1.kya`
   - `L4Q2.kya`
   - `L4Q3.kyx`
   - `L4Q3.kya` (extra-credit)
   - `L4.txt`
   - `andrewids.txt (if working in pairs)`