15-424/15-624/15-824 Lab 2
15-424/15-624/15-824 Foundations of Cyber-Physical Systems
Course TA: Brandon Bohrer (bbohrer@cs)

Betabot Due Date: 2/17/16 **3:00 PM**, worth 20 points
Veribot Due Date: 2/23/16 **11:59PM**, worth 80 points

1. **Event-triggered Highway Driving**

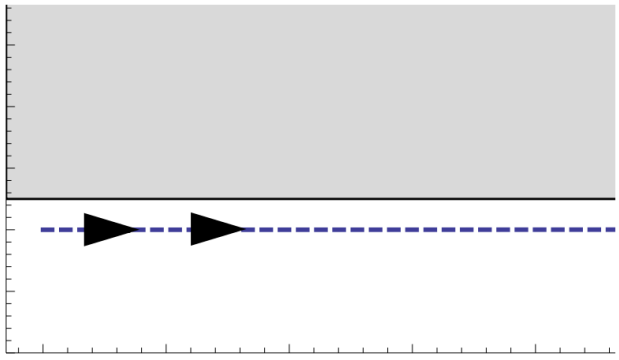

Figure 1: Lead and control car

In this problem, you will design a hybrid program (HP) to model a controlled car (`ctrl`) following a lead car (`lead`) along a straight road.

You can get templates for the problems from `http://symbolaris.com/course/fcps17/lab2.zip`.

- The lead car should keep a constant and nonnegative velocity (i.e. $vel_{lead} \geq 0$).
- The driver of the controlled car can only choose to accelerate at rate ($A > 0$), or brake at rate $-B$, where ($B > 0$). The choice of acceleration A should only be available to the driver when it is safe – a condition that you will have to define.
- The controlled car has continuous access to the lead car's position and velocity (i.e. the controller you design should be *event-triggered*).
- Assume the cars are infinitesimal points. In other words, a crash occurs only if the position of the controlled car exceeds the position of the lead car (i.e. a crash occurs only if $pos_{ctrl} > pos_{lead}$).
- Your controller must never blow up, otherwise the safety property would be vacuously true in some or all cases. For example the safety theorem $[?\mathit{false}]\mathit{safe}$ holds vacously for any formula *safe* because the test $?\mathit{false}$ always fails. To avoid this, make sure your tests work like if-then-else statements. If you have $(?(\phi_1);\alpha_1) \cup \ldots \cup (?(\phi_n);\alpha_n)$ you must make sure the ϕ_i's are exhaustive, i.e. at least one of them is true in every state.

1.1 [Betabot] What is a good safety condition for this system? A good efficiency condition? Add this to `L2Q1_discussion.txt`.

1.2 [Betabot] Fill in the missing parts of the HP below to model this system. Also fill in your safety condition from 1.1. Save this file as `L2Q1.kyx`. 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.

1.3 [Veribot] Use KeYmaera X to prove that the HP you designed in 1.2 satisfies your safety condition and download the resulting `L2Q1.kya` file.

1.4 [Veribot] **Bonus:** Drivers get uncomfortable when their car gets too close to the car ahead. Update your safety condition to require that the cars never come within constant distance `c` of each other. Then update your model to satisfy this requirement and prove it in KeYmaera. Only attempt the bonus problem *after* proving safety without the buffer – you are required to submit both versions to get credit.

2. **Time-triggered Highway Driving**

In this problem, you should allow the lead car to either accelerate at rate A or brake at rate $-B$ and also change the controller from being event-driven to being time-driven. This means that when your car chooses an acceleration, it may be stuck with that choice for some time. Your model will now have a "stop watch" which must be set to 0 before each continuous evolution.

- The lead car may accelerate or brake arbitrarily at rate A or $-B$. The controlled car never has access to the lead car's acceleration.
- In Part 1, your car could only accelerate at rate A or brake at rate $-B$. This means that once it comes to a stop, it has no option but to accelerate. If acceleration is not safe, then the controller has no control options and the safety property would become vacuously true. To address this issue and avoid vacuously true theorems, in this problem you are allowed to set the acceleration to anything between $-B$ and A (inclusive).
- The controlled car has intermittent access to the lead car's position and velocity. The time between updates is variable, but is guaranteed to be less than time `T` (i.e. your controller must be *time-triggered*).
- The controller should never blow up (i.e. the transition semantics of the hybrid program should not be empty), so be careful that your tests are exhaustive.

2.1 [Betabot] Using the template, design a time-triggered controller and model the system as a hybrid program. Then, write a `dL` formula that shows your safety condition from Question 1.1 is still satisfied by this controller. Submit this file as `L2Q2.kyx`.

2.2 [Veribot] Using KeYmaera X, prove that your `dL` formula is true. Downlad the resulting `L2Q2.kya` file.

2.3 [Veribot] **Question:** Compare and contrast the Event-triggered and Time-triggered highway driving. Describe their relationship. Which was easier to prove safe? Which would be easier to implement? Why and what caused these differences? Submit your answer to this question in `L2Q2_discussion.txt`.

3. **Submission Checklist**

Labs may be submitted in groups of two. If that is the case, then you must submit a file `andrewids.txt` containing both of your andrewids, **otherwise one of you will not get credit because I will think you did not submit.** To make my grading infrastructure happy, please put them on the same line separated by a space like this: `aplatzer bbohrer`. Only one of you needs to submit on Autolab.

(a) **BetaBots:** submit a zip file on autolab containing your preliminary .kyx files for each of the tasks as well as the BetaBot dicussion file. If you have GNU Make installed, running `make` in the directory where you unpacked the `lab2` zip will create your handin zip. The Makefile does not check whether you have all the correct files. Instead, check your "autograding" results on Autolab: this will not tell you whether your submission is correct, but it will ensure that you have all the correct files and your models are syntactically well-formed. 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 Fri 02/17.*

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 me to provide more useful feedback.

(b) **Final submission:** The final submission works the same way, except you submit `.kya` files (which contain both the model and the proof). To receive credit, proofs must be complete (i.e. the "Property Proved" dialog appears after copy/pasting the proof from your `.kya` file into the Proof Programming window in the Web UI. Also submit your veribot discussion file. As above, you can use the provided Makefile to build your zip file, and you should check your Autolab results when you submit. *Due Thu 02/23.*

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 BetaBot zip file should contain:

- `L2Q1.kyx`
- `L2Q2.kyx`
- `L2Q1_discussion.txt`
- `andrewids.txt` (if working in pairs)

The VeriBot zip file should contain:

- `L2Q1.kya`
- `L2Q2.kya`
- `L2Q2_discussion.txt`
- `andrewids.txt` (if working in pairs)