15-424/15-624 Lab 2
15-424/15-624 Foundations of Cyber-Physical Systems
Course TAs: João Martins (jmartins@cs), Annika Peterson (apeterso@andrew)

Betabot Due Date: 9/24/14, worth 20 points
Veribot Due Date: 10/1/14, worth 80 points

1. **Event-triggered Highway Driving**

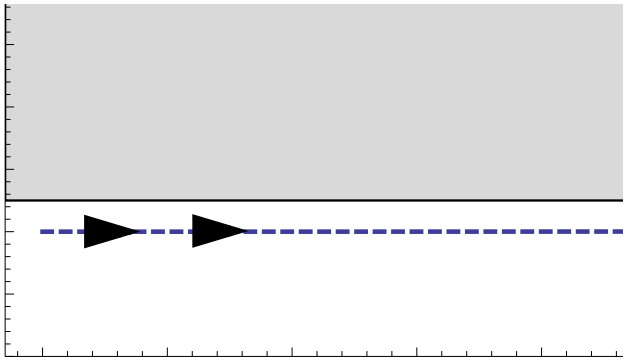

Figure 1: Lead and control car

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 `symbolaris.com/course/fcps14/lab2.zip`.

- The lead car should keep a constant and positive 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 $pos_{ctrl} > pos_{lead}$).
- The controller should always have a valid choice (i.e. the transition semantics of the controller should never be empty).

1.1 [Betabot] What is a good safety condition for this system? A good efficiency condition? Add this to `L2_name.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_name.key`.

1.3 [Veribot] Use KeYmaera to prove that the HP you designed in 1.2 satisfies your safety condition. Save the resulting proof as `L2Q1_name.proof` and the corresponding .key file as `L2Q1_name.key`. Some useful proving techniques to review are: the $\forall$ left rule and how to find and hide irrelevant formulas. See the youtube tutorial videos[1] and course notes for more information.

[1] `http://video.symbolaris.com`

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.

```
(_________) /* initial conditions */
->
\[
  (
    (   /* Safely assign accelerate or brake for ctrl */
        ---------
    );
    /* Continuous dynamics */
    ({ _________ & _________} /* Evolution domain and event-trigger */
     ++
     { _________ & _________} /* Evolution domain and inverse event-trigger */
    )
  )*@invariant(_________)  /* Loop Invariant */
\] (_________) /* Safety Condition */
```

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 or brake. 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 transition. You have more freedom in your controller design for this question to address this issue.
- 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 always have a valid choice (i.e. the transition semantics of the hybrid program should not be empty).

```
(_________)          /* Requires (initial conditions) */
->
\[
    (
        ________;        /* Assign a safe acceleration to ctrl */
        ________;        /* Assign braking or acceleration to lead */
        t := 0;          /* Start the stop watch */
        {________ &      /* Continuous dynamics (don't forget about time!) */
         ________ t <= T}          /* Evolution domain and time-trigger */
    )*@invariant(________)         /* Loop Invariant*/
\]
(_________)          /* Safety condition */
```

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_name.key`.

2.2 [Veribot] Using KeYmaera, prove that your dℒ formula is true. Save and submit the proof as `L2Q2_name.proof`, along with an updated version of your .key file `L2Q2_name.key`. In addition to the proving techniques you used for part 1, it will be useful here to use a cut to remove the time variable T.

2.3 [Veribot] **Question:** Compare and contrast the Event-triggered and Time-triggered highway driving. Which was easier to prove safe? Which would be easier to implement? Why? Submit your answer to this question in `L2_name.txt`.

3. **Submission Checklist**

Labs can be submitted in groups of two. If that is the case, then the submissions should look like `L2Q?_name1_name2.key` and `L2Q?_name1_name2.proof`. Only one of you needs to submit it on autolab.

(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 09/24.*

(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 10/1*

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:

- `L2Q1_name.key`
- `L2Q1_name.proof`
- `L2Q2_name.key`
- `L2Q2_name.proof`