**15-424/15-624 Lab 1**
**15-424/15-624 Foundations of Cyber-Physical Systems**

Test Due Date: 9/11/13, worth 20 points
Final Due Date: 9/18/13, worth 80 points
Course TA: Sarah Loos (sloos+fcps@cs.cmu.edu)

1. **Autobots, Roll Out**

   As the n00b in your robotics group, the senior members gave you the broken robot. Its steering is jammed, so it can only move in a straight line and control acceleration.

   1. [**test**] Fill in the missing continuous dynamics in the hybrid program below that will model your robot accelerating in a straight line, with position `pos`, velocity `vel`, and acceleration `acc`.

   2. [**test**] Fill in the safety condition that will ensure the velocity of the robot is never negative. Save this file as `test1_username.key` (where username is replaced with your andrew ID) and submit to autolab by the test due date. Templates are provided in lab1.zip for your convenience.

   3. [**final**] Use KeYmaera to prove the velocity of the robot is always positive. Save the resulting key and proof files as `final1_username.key` and `final1_username.proof`.

```
(vel = 0 & acc >= 0)      /* Requires */
->
\[{--------}\]           /* Continuous dynamics */
(--------)                /* Ensures (safety condition) */
```

2. **Charging Station, Single Control**

   Oh no! Your robot's battery is almost dead! Luckily the robot is already on a straight trajectory to a wall charging station and has positive velocity. With its remaining power, your robot has just *one chance* to engage the brakes properly to bring the robot to a stop at the right place.

   - If the robot brakes too hard, it will not make it to the charging station and will have to wait for a human to plug it in. Running out of power is *inefficient*.
   - If the robot doesn't brake hard enough to stop at the charging station, it will dent the wall behind the station. When building manager Jim Skees finds out, he'll have your robot banned from the building! Running into walls is *unsafe*.

   1. [**test**] Specify the missing safety and efficiency requirements that the hybrid program below should ensure. Then fill in the missing control and continuous dynamics. Save the resulting file as `test2_username.key` and submit to autolab by the test deadline.

   2. [**final**] Use KeYmaera to prove that the hybrid program is safe and efficient. Save the resulting proof and associated key file as `final2_username.proof` and `final2_username.key`.

   3. [**final**] **Question:** What is the evolution domain for the continuous dynamics in this hybrid program? Why is it necessary? Add your answer to text file named `lab1_username.txt`.

```
(pos < station & vel > 0) /* Requires */
->
\[
    acc := ---------;     /* Assign a safe acceleration. */
    {----- & vel >= 0}    /* Use continuous dynamics from part 1. */
\]
(------------             /* Ensures (safety condition) */
 &------------)           /* Ensures (efficiency condition) */
```

3. **Charging Station, Double Control**

In part 2, you are always able to coast the robot all the way to the charging station, since it is already moving. But what if the robot is stopped? In this question, the goals are the same as in part 2; however, the robot starts with zero velocity. You have two chances to control the robot, first to get it moving by accelerating, and then to bring it to a stop at the right point by braking.

The robot also has a time limit `T` on how long it can accelerate before exhausting the remaining battery life. Once the brakes are engaged, they will stay engaged until the robot comes to a complete stop.

1. [**test**] Submit the filled in formula below by the test due date as `test3_username.key`.

2. [**final**] Submit your updated key file (`final3_username.key`) and proof (`final3_username.proof`) by the final due date.

3. [**final**] **Question:** What is your efficiency condition? Is it different from part 2, why or why not? Add your answer to `lab1_username.txt`.

```
(pos < station & vel = 0 & 0 < T)        /* Requires */
->
\[
    t := 0;
    acc := _________;                    /* Assign a safe acceleration. */
    {____, t' = 1 & vel >= 0 & t <= T};
    ?(t > 0);
    acc := _________;                    /* Assign a safe deceleration. */
    {____, t' = 1 & vel >= 0}
\]
(____________                            /* Ensures (safety condition) */
 &____________)                          /* Ensures (efficiency condition)*/
```

4. **Submission Checklist**

For both submissions, submit a zip file via autolab containing the listed files. In each file name, `username` should be replaced by your Andrew ID. Autolab is used for collection, but will not automatically grade your assignment. Files with the extension `.key` are just text files containing hybrid programs. Before submitting, check that each `.key` file parses by loading into KeYmaera without producing error messages. Files with the extension `.proof` are automatically generated by KeYmaera. You can save a proof while it is in progress or when it is completed by clicking on File → Save in KeYmaera. To receive full credit, proofs must be complete (i.e. the "Property Proved" window has appeared, so all branches are closed and there are no remaining goals).

Test submission (Due 9/11):

- `test1_username.key`
- `test2_username.key`
- `test3_username.key`

Final submission (Due 9/18):

- `final1_username.key`
- `final1_username.proof`
- `final2_username.key`
- `final2_username.proof`
- `final3_username.key`
- `final3_username.proof`
- `lab1_username.txt`