Assignment 4: ODEs, Games, and Nondeterministic Assignments
15-424/15-624/15-824 Logical Foundations of Cyber-Physical Systems

TAs: Irene Li (mengzeli@andrew.cmu.edu)
Yong Kiam Tan (yongkiat@cs.cmu.edu)

Due Date: Thursday, November 1st, 11:59PM (no late days), worth 60 points

1. **Easy as π.** In class, we have started looking at some more interesting differential equations with curved motion. Use this new knowledge to create a hybrid program which has no transcendental literals or trigonometric functions (e.g., π, e, sin, cos), but at the end of execution has the exact value of π in a variable named *pi*. Does this mean that we can now use π in hybrid programs? If so, should we? Explain.

2. **Ghostly proof rules.** Recall the differential auxiliaries (dA) proof rule which can be derived from the differential ghosts axiom:

$$\text{(dA)}\quad \frac{\vdash F \leftrightarrow \exists y\, G \qquad G \vdash [\{x' = f(x), y' = a(x)\cdot y + b(x)\,\&\,Q\}]G}{F \vdash [\{x' = f(x)\,\&\,Q\}]F}$$

This proof rule says that we can add an extra ghost variable y that follows a new differential equation $y' = a(x)\cdot y + b(x)$ that is linear in y. The extra variable can be used to rewrite the invariant F in a way that makes it more amenable to proof using the other proof rules of dL.

In class, we saw how to use the following special instance of dA to prove interesting properties in the case where $F \equiv p > 0$ is a strict inequality:

$$\text{(dA}_>\text{)}\quad \frac{py^2 > 0 \vdash [\{x' = f(x), y' = a(x)\cdot y + b(x)\,\&\,Q\}]py^2 > 0}{p > 0 \vdash [\{x' = f(x)\,\&\,Q\}]p > 0}$$

Notice that we have omitted the left premise of dA in dA$_>$ because $p > 0 \leftrightarrow \exists y\, py^2 > 0$ is a provable formula of real arithmetic.

   (a) In the same style as dA$_>$, write a proof rule called dA$_\geq$ that would (soundly) allow you to prove properties of the form $F \equiv p \geq 0$. Briefly argue why your proposed proof rule is sound.

   Hint: Haven't we seen dA on a previous assignment?

   (b) To test out your proposed dA$_\geq$ proof rule, use it to prove the following property:

$$x \geq 1 \vdash [\{x' = 2 - 2x\}]x \geq 1$$

3. **Taylor series.** When an ODE cannot be solved exactly, a useful technique is to use a *Taylor series approximation* to get an upper or lower bound on the solution instead. Prove the following formula using the proof rules and axioms of dL:

$$x = 1 \land t = 0 \rightarrow [\{x' = x, t' = 1 \,\&\, x \geq 1\}]x \geq 1 + t + \frac{t^2}{2}$$

Recall that the solution of the ODE $x' = x$ (with initial value $x_0 = 1$) is $x(t) = e^t$, so the above formula expresses a lower bound for e^t (for all $t \geq 0$).

4. **Games, games, games.** Answer these 3 questions for each of the following formulas:

   - For which starting states does Angel have a winning strategy? (Recall that $\langle\alpha\rangle\phi$ means Angel has a strategy to win into ϕ for hybrid game α)
   - Briefly describe Angel's winning strategy from those starting states.
   - (Only applies to games where Angel has a winning strategy in at least one state). Say we let Demon pick **one occurrence of one** hybrid program operator and flip it between being an Angel or Demon operator, e.g. replacing **one** $\alpha \cup \beta$ with $\alpha \cap \beta$ or vice-versa. Can Demon can make it so that Angel never has a winning strategy in any state?

   (a) **A warm-up:** $\langle(x := 0 \cap x := 1)^{\times}\rangle x \geq 0$

   (b) **Ups and Downs:**

   $$\langle\big((x := x + 1 \cup \{x' = v\}^d); (y := y - 1 \cup \{y' = w\}^d)\big)^*\rangle|x - y| \leq 1$$

   (c) **A chase:** $\langle(w := w \cap w := -w); (v := v \cup v := -v); \{x' = v\}^d; \{y' = w\}\rangle x < y$

   Hint: Try to give an intuitive reading to the hybrid games before thinking of Angel's strategies.

5. **Lab 1 revisited.** In Lab 1, you wrote a hybrid program in which a robot accelerates along a straight line for a non-zero duration less than or equal to T, and then decelerates to stop at a charging station. We will now revisit this problem using nondeterministic assignments of the form $x := *$.

   (a) As a warmup, write a hybrid program using guarded nondeterministic assignment (that is, nondeterministic assignment plus a test) which assigns *acc* to be any real number in the range $[0, A]$.

   (b) Rewrite the hybrid program from Lab 1 using a *more interesting* guarded nondeterministic assignment. Your guard should allow *all* safe choices of acceleration, thus defining a safety envelope within which all control choices are safe.

For simplicity, you can use the following partial solution to Lab 1:

```
Init
->
[
   t := 0;
   acc := (station-pos) / (T * T);  // Modify this assignment!
   {pos' = vel, vel' = acc, t' = 1 & vel >= 0 & t <= T};
   ?(t > 0);
   acc := -(vel^2 / (2 * (station - pos)));
   {pos' = vel, vel' = acc , t' = 1 & vel >= 0}
]
( Safe & Efficient )
```

Recall that the safety requirement was for the robot to never run into the charging station and the efficiency requirement was for the robot to come to a stop *at the station* (i.e., not too early or it will not get recharged).

(c) What are the pros and cons of using a guarded nondeterministic assignment in this model?

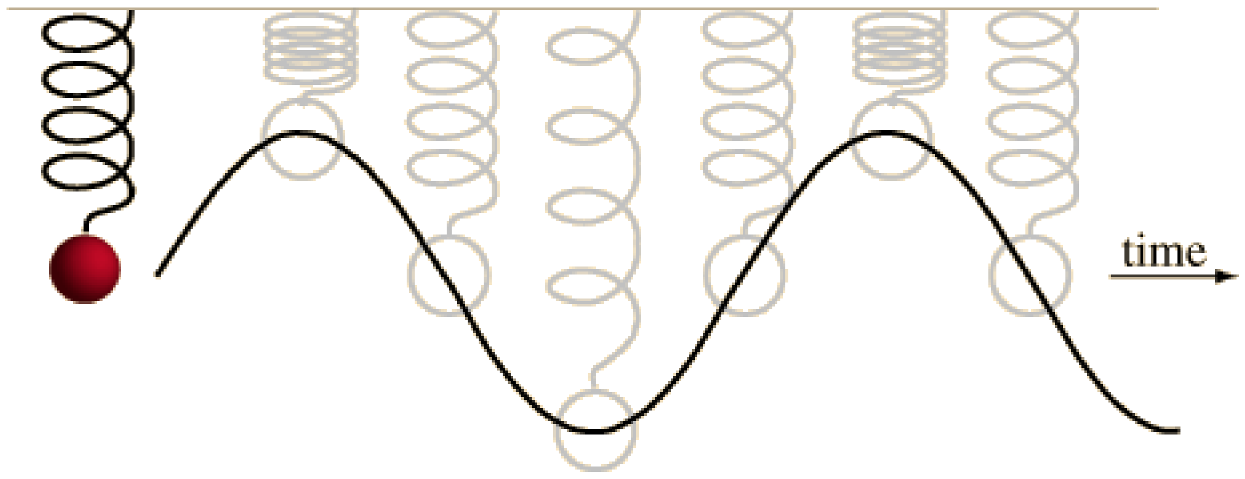
(d) *Suppose* you were to prove the safety and efficiency properties from Lab 1 for the new hybrid program. Is this new theorem stronger than the one given above (in other words, would this theorem imply the original)? Or is the old theorem stronger? Or neither? Explain. You do not have to actually do the proof.

(e) *Suppose* that the second assignment to *acc* was also changed to a guarded nondeterministic assignment. What are the control choices that could be safely allowed by its guard? Explain your answer.

6. **Quantum's Adventures Part 2: Finding Harmony.** Modeling is a crucial part of CPS design. If we write down unsuitable models, we get unsuitable CPSs which will lead us to ultimately meaningless proofs. This question will give you the opportunity to exercise and sharpen your modeling skills. Consider the following scenario:

   Quantum loves to bounce, but sometimes it gets lonely. Luckily, it is friends with a spring named Harmony that hangs all day from the ceiling. When they hang out, Quantum likes to hold on to the end of the spring and bounce. However, being a careful little ball, it is worried about running into the ceiling or the floor. Luckily hybrid programs can come to the rescue!

   Let us model Quantum and Harmony's movement. This set-up can be visualized as follows:



The main force that describes the motion of a spring is called the spring force, and it is described with Hooke's Law as:

$$F = -kx$$

where k is the spring constant ($k > 0$) and x is the displacement from resting position. Applying Newton's second law of motion, we get the following differential equation for this situation:

$$\frac{d^2x}{dt^2} + \frac{k}{m}x = 0$$

(a) First, define the safety conditions that you would use. (Safety first!)

(b) Next, define the ODEs of motion.

(c) Overall, this is a fairly simple model – it does not even have any controls! Now, come up with 2-3 ways this model can be expanded (feel free to be creative and incorporate new forces or sources for control).

(d) Pick one of these ideas and design a hybrid program to model it. If you can only come up with a partial model, that is okay, but explain what you tried and the difficulties you ran into.

Note: The goal of this question is to give you practice with open-ended models which you will need to build in the course project. So, do try to experiment!