Assignment 4: Differential Invariants and Nondeterministic Assignment
15-424/15-624 Foundations of Cyber-Physical Systems
TAs: Brandon Bohrer (bbohrer@cs)

Due: 11:59pm, Thursday, 3/23/17
Total Points: 80 Revision 4: Typos in Q5, Q6, Lab 1 Warmup clarification, Syntactic Derivatives clarification

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 functions (example 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. **Syntactic derivatives.** In Lecture 11, the *syntactic derivative* of a formula is defined (Definition 12) as follows:

$$(\theta \leq \eta)' \equiv ((\theta)' \leq (\eta)') \tag{1}$$
$$(\theta < \eta)' \equiv ((\theta)' < (\eta)') \tag{2}$$
$$(\theta \neq \eta)' \equiv ((\theta)' = (\eta)') \tag{3}$$
$$(\theta = \eta)' \equiv ((\theta)' = (\eta)') \tag{4}$$
$$(P \wedge Q)' \equiv ((P)' \wedge (Q)') \tag{5}$$
$$(P \vee Q)' \equiv ((P)' \wedge (Q)') \tag{6}$$

   The following slightly relaxed definition for the syntactic derivative of a strict inequality (2) would also give a sound proof rule for differential invariants:

$$(\theta < \eta)' \equiv ((\theta)' \leq (\eta)')$$

   (a) Revision 4: Expand list of rules/axioms that might be useful. Brandon wants to prove that this is a sound definition as a *derived axiom*, meaning that he would prove it just by using existing rules other than 2 like (1), (3), (4), (5) ,(6) (plus other existing syntactic proof rules/axioms for d$\mathcal{L}$ that you know and love), but without looking at the semantics of hybrid programs (and without splitting casing/inducting on θ, η). Briefly explain why this approach cannot work.[1] You can assume the relaxed version of this axiom in the rest of the problem.

   (b) Suppose you remove definition (3) so that you can no longer use the differential invariant proof rule for formulas involving $\neq$. Can you derive a proof rule to prove

[1] The correct approach to prove this axiom sound is a direct *semantic* proof, which then reduces to the Mean Value Theorem. But we're not asking you to do the semantic proof, just show what goes wrong in the syntactic proof.

such differential invariants regardless (in the same style as the previous question)? If so, describe how you would derive the rule? If not, explain why not. You do not have to do a proof for this question.

3. **Valid, satisfiable, or unsatisfiable.** For each of the following, determine whether the statement is valid, satisfiable, or unsatisfiable.

   (a) $\exists x \langle a := * \rangle a = x$

   (b) $\forall x [a := *] a = x$

   (c) $\forall x \langle a := * \rangle a = x$

   (d) $[a := *] a = x$

   (e) $\langle a := * \rangle (a = x \wedge a = y)$

   (f) $\langle a := * \rangle (a = x) \wedge \langle a := * \rangle (a = y)$

4. **Lab1 revisited.** In Lab 1, Question 3, 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 assignment.

   (a) **Warm-up:** Revision 3: Clarify that guarded nondeterminstic assignments are also cool for the 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 $[0, A]$ for the first choice of acceleration.

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

```
Problem.
(pos < station & vel = 0 & T > 0)
->
[
   t := 0;
   acc :=  (station-pos)/ (T*T);
   {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}
]
( pos<=station & (vel^2 + 2 * acc * (station - pos))=0)
End.
```

   (b) Rewrite this hybrid program using a *more interesting* guard! Your guards should allow *all* safe choices of acceleration, thus defining a safety envelope within which all control choices are safe.

(c) What are the pros and cons of changing the hybrid program from Lab 1 to use guarded nondeterministic assignment?

(d) *Suppose* you were to prove the safety and efficiency properties from Lab 1 for this new hybrid program. Is this new theorem stronger than the one you proved in Lab 1 (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.

5. **Spooky Ghosts**

Recall the general-purpose proof rule for DA (*Differential Auxiliaries*), which is the most common case of differential ghosts:

$$\text{dA}\ \frac{P \leftrightarrow \exists y.J \qquad \Gamma, J \vdash [x' = f(x), y' = g(y,x)\&Q]J, \Delta}{\Gamma, P \vdash [x' = f(x)\&Q]P, \Delta}$$

This proof rule says that we can extra variables to an ODE and use those extra variables to rewrite our differential invariant in a way that makes it provable when it wasn't provable before. This rule is necessary because a differential invariant argument always says some property gets "more true" or at worst stays "equally true" over time, e.g. if we have $(\theta_1 < \theta_2)$ then $\theta_2 - \theta_1$ should be increasing. Some properties which do stay true over time actually get closer and closer to false, e.g. if your phone battery is decreasing exponentially it will never reach exactly 0 but the formula battery > 0 is getting "more false" over time. Adding extra variables allows us to take such properties and rewrite them in ways that **do** get "more true" over time.

Recall in recitation we presented a few different idealized, simplified versions of the DA rule which handled commonly-occuring cases, such as:

Revision 1: There were typos in this rule

$$DA_{<}\ \frac{(e < 0) \leftrightarrow \exists y.y^2 \cdot e = -1 \qquad \Gamma, (y^2 \cdot e = -1) \vdash [x' = f(x), y' = g(y,x)\&Q](y^2 \cdot e = -1), \Delta}{\Gamma, e < 0 \vdash [x' = f(x)\&Q](e < 0), \Delta}$$

Here the branch $(e < 0) \leftrightarrow \exists y.y^2 \cdot e = -1$ is actually a tautology (always valid), so we do not have to prove it ourselves, but it is an essential part of understanding how the proof rule works so we write it out here anyway.

(a) In the same style as $DA_{<}$, write a DA proof rule called $DA_{>}$ that would (soundly) allow you to prove properties of the form $e > 0$. You do not need to prove that this rule is sound, but it should be sound. In designing $DA_{<}$, one of the key insights was that $(e < 0) \leftrightarrow \exists y.y^2 \cdot e = -1$ is always true, allowing us to rewrite e in terms of a new variable y, improving the differential structure of the formula $e < 0$ in the process. Similarly, we can rewrite $(e > 0) \leftrightarrow \exists y.P(y,e)$ for some predicate P. Write out such a predicate P as part of your solution.

(b) In the same style, write a DA proof rule called $DA_{\leq}$ that allows you to prove formulas of the form $e \leq 0$. As above, write out a P such that $(e \leq 0) \leftrightarrow \exists y.P(y, e)$

(c) Here is an example property that requires a differential ghost/DA:

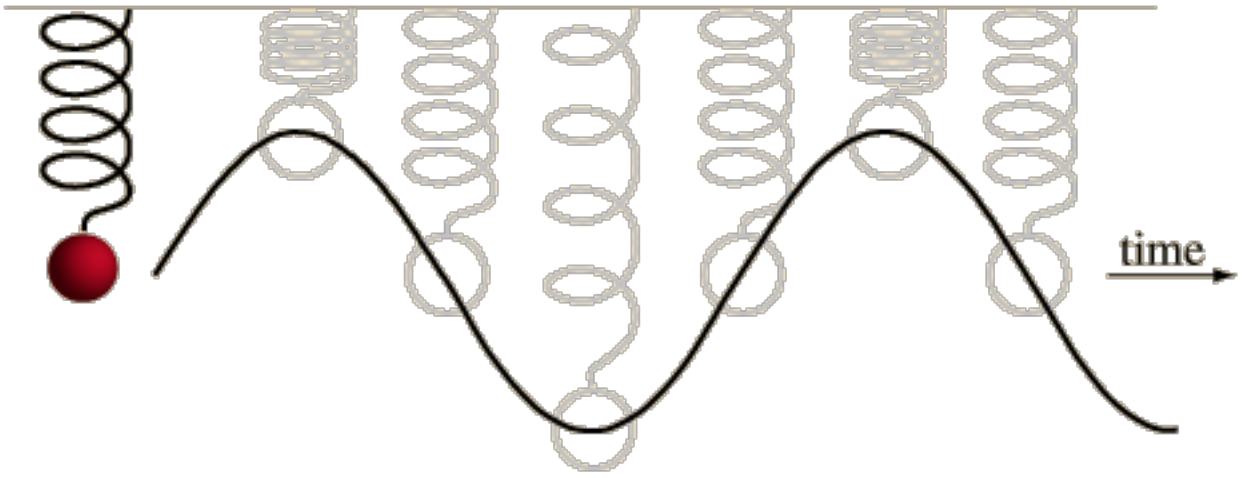
$$x > 1 \vdash [\{x' = 2 - 2x\}]x > 1$$

If you were to prove this property, the next step will be applying your rule $DA_{>}$. When applying $DA_{>}$ it is up to us to plug in a definition of $g(y, x)$ that results in a provable differential invariant. Write out a choice of $g(y, x)$ that makes this property provable. You do not have to show all your work if the answer is correct, but wrong answers without work shown will not receive partial credit.

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 ultimiately 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 he gets lonely. Luckily, he 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, he is worried about running into the ceiling or the floor. Luckily hybrid programs can come to the rescue!

Let's 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: Revision 2: x was misnamed Δ

$$F = -kx$$

where k is the spring force (some constant) and x is the initial 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 doesn't 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) Now, 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.
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!