Report
LEARNING FROM DEAD ENDS

15-816 Modal Logic

David Renshaw

May 10, 2010

Abstract

This project examines one way to attack a particlarly challenging part of proving properties about hybrid systems: invariant generation. The idea is to use information from failed proof branches to guide the search and to avoid duplicated work. This idea is apparently helpful in some, but not all, situations.

1 Introduction

A hybrid program is a description of a system governed both by discrete and by continuous dynamics [1]. Hybrid programs are useful for describing modern technological artifacts like drive-by-wire cars and robotic medical devices which today increasingly involve both computation and physics.

Differential Dynamic Logic ($\mathbf{d}\mathcal{L}$) [2] is a multi-modal logic designed for proving properties of hybrid programs. Proof search in this logic is (unsurprisingly) undecidable and presents many difficulties in practice. Chief among them is how to deal with the continuous dynamics, which may introduce (among other things) uncountable nondeterminism. Indeed, it has been shown [2] that in a certain precise sense *all* of the difficulty of proof search in $\mathbf{d}\mathcal{L}$ is contained in the search for facts about differential equations.

One approach is to use differential invariants [4], an extension of the notion of loop invariants. Differential invariants have been shown to be useful, but as of yet there is no principled way to search the space of possible invariants.

The goal of the present project is to develop a method to guide the search for invariants. The idea is that if a proof obligation does not close because we chose the wrong invariant, then we should be able to learn something about *why* the obligation did not close. If that information gets propagated back to invariant-generatation step, we might stand a better chance of succeeding on our next try.

2 Background

I now describe the syntax and Kripke semantics for a simplified version of $\mathbf{d}\mathcal{L}$. There are three kinds of ground symbols: *logical variables* such as $X_1, X_2, ...$, *state variables* such as $x_1, x_2, y_1, y_2, ...$, and *rigid functions* such as $0, 1, 2, +, \cdot$.

A $\mathbf{d}\mathcal{L}$ formula is a formula in first-order logic over the real numbers, where there are modalities $[\alpha]$ and $\langle\alpha\rangle$ for every hybrid program α. Possible worlds w_i are functions from the set of state variables to

real numbers. Each hybrid program α defines an accesibility relation ρ_α on the possible worlds. The box modality $[\alpha]\phi$ means that ϕ holds after every *every* ρ_α-successor; The diamond modality $\langle\alpha\rangle\phi$ means that ϕ holds after *some* ρ_α-successor. The accessibility relations are defined inductively as follows.

α	*name*	$\rho_\alpha(w_1, w_2)$ when...
$x := \theta$	assignment	$w_2(x) = w_1(\theta)$
$x := *$	nondeterministic assignment	$w_1(y) = w_2(y)$ for all $y \neq x$
$?H$	condition check	$w_1 = w_2$ and H is true there
$\alpha_1 \cup \alpha_2$	nondeterministic choice	$\rho_{\alpha_1}(w_1, w_2)$ or $\rho_{\alpha_2}(w_1, w_2)$
$\alpha_1; \alpha_2$	sequence	$\exists w'$ s.t. $\rho_{\alpha_1}(w_1, w'), \rho_{\alpha_2}(w', w_2)$
α_1^*	loop	$\rho_{\alpha_1}(w_1, w_1'), \rho_{\alpha_1}(w_1', w_2')...\rho_{\alpha_1}(w_k', w_2)$ for some $k \geq 0$
$\mathcal{D}$	evolution	defined below

An evolution is of the form $\mathcal{D} = \{x_1' = \theta_1, x_2' = \theta_2', ..., x_k' = \theta_k; H\}$ where the x_i's are state variables and the θ_i's are terms in real arithmetic. We have $\rho_\mathcal{D}(w_1, w_2)$ if and only if w_1 and w_2 are the respective start and end points for a solution φ to the differential equation defined by $\mathcal{D}$, where the solution curve remains within H and state variables not mentioned by $\mathcal{D}$ remain constant.

3 Proof Calculus

The proof rules we will use are adapted from [4].

$$\frac{\Gamma, \hat{x} = \theta \vdash [\hat{x}/x]\phi}{\Gamma \vdash [x := \theta]\phi}(\text{asgn}) \qquad \frac{v \text{ fresh function symbol} \qquad \Gamma \vdash [v/x]\phi}{\Gamma \vdash [x := *]\phi}(\text{asgn-any}) \qquad \frac{\Gamma, H \vdash \phi}{\Gamma \vdash [?H]\phi}(\text{test})$$

$$\frac{\Gamma \vdash [\alpha]\phi \qquad \Gamma \vdash [\beta]\phi}{\Gamma \vdash [\alpha \cup \beta]\phi}(\text{choose}) \qquad \frac{\Gamma \vdash [\alpha][\beta]\phi}{\Gamma \vdash [\alpha; \beta]\phi}(\text{seq})$$

$$\frac{\Gamma, \phi \text{ first order} \qquad (\Gamma \to \phi) \text{ true in real arithmetic}}{\Gamma \vdash \phi}(\text{close})$$

$$\frac{H \vdash \phi}{\Gamma \vdash [\mathcal{D} \wedge H]\phi}(\text{diff-done}) \qquad \frac{\Gamma, H \vdash F \qquad H \vdash \nabla_\mathcal{D} F \qquad \Gamma \vdash [\mathcal{D} \wedge (H \wedge F)]\phi}{\Gamma \vdash [\mathcal{D} \wedge H]\phi}(\text{diff-strengthen})$$

$$\frac{\Gamma \vdash F \qquad F \vdash [\alpha]F \qquad F \vdash \phi}{\Gamma \vdash [\alpha^*]\phi}(\text{loop-ind})$$

Here $\nabla_\mathcal{D} F$ is a total derivative which substitutes from $\mathcal{D}$. See [4] for a formal definition.

4 Investigation of Examples

I now examine four sample problems and see how much useful information can be gathered at dead ends.

4.1 Water Tank

Define

$$\begin{aligned}
S_0 &\equiv ?(s=0);(?(y=10);x:=0;s:=1)\cup(?(y<10\vee y>10);\{x'=1,y'=1,y\leq 10\})\\
S_1 &\equiv ?(s=1);(?(x=2);s:=2)\cup(?(x<2\vee x>2);\{x'=1,y'=1,x\leq 2\})\\
S_2 &\equiv ?(s=2);(?(y=5);x:=0;s:=3)\cup(?(y>5\vee y<5);\{x'=1,y'=-2,y\geq 5\})\\
S_3 &\equiv ?(s=3);(?(x=2);s:=0)\cup(?(x>2\vee x<2);\{x'=1,y'=-2,x\leq 2\})\\
\phi &\equiv (y\geq 1\wedge y\leq 12)
\end{aligned}$$

We want to prove the formula

$$x=0,y=1,s=0\vdash[(S_0\cup S_1\cup S_2\cup S_3)^*]\phi$$

Let's try to prove it using the postcondition as a loop invariant. That breaks the proof into four branches (The asterisk * indicates a goal that closes; some steps are elided):

$$\dfrac{\dfrac{*}{x=0,y=1,s=0\vdash\phi}\qquad\dfrac{\phi\vdash[S_0]\phi\qquad\phi\vdash[S_1]\phi\qquad\phi\vdash[S_2]\phi\qquad\phi\vdash[S_3]\phi}{\phi\vdash[(S_0\cup S_1\cup S_2\cup S_3)^*]\phi}\qquad\dfrac{*}{\phi\vdash\phi}}{x=0,y=1,s=0\vdash[(S_0\cup S_1\cup S_2\cup S_3)^*]\phi}$$

The first branch of the proof will require us to prove $\phi\vdash[S_0]\phi$. It closes:

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{\phi,s=0,y=10,x=0,s=1,\vdash\phi}}{\phi,s=0,y=10\vdash[x:=0;s:=1]\phi}\qquad\dfrac{\text{diff-strengthen with } y\geq 1\wedge s=0 \text{ as invariant}}{\phi,s=0,y<10\vee y>10\vdash[\{x'=1,y'=1,y\leq 10\}]\phi}}{\phi,s=0\vdash[?(y=10);x:=0;s:=1]\phi}\qquad\dfrac{}{\phi,s=0\vdash[?(y<10\vee y>10);\{x'=1,y'=1,y\leq 10\}]\phi}}{\phi,s=0\vdash[(?(y=10);x:=0;s:=1)\cup(?(y<10\vee y>10);\{x'=1,y'=1,y\leq 10\})]\phi}}{\phi\vdash[S_0]\phi}$$

The second branch of the main proof will require us to prove $\phi\vdash[S_1]\phi$. Let's see whether this succeeds.

$$\dfrac{\phi,s=1\vdash[(?(x=2);s:=2)\cup(?(x<2\vee x>2);\{x'=1,y'=1,x\leq 2\})]\phi}{\phi\vdash[S_1]\phi}$$

After a few steps (including an or-left split) we are left with the open goal

$$\phi, s = 1, x < 2 \vdash [\{x' = 1, y' = 1, x \leq 2\}]\phi$$

This goal cannot be closed! If the initial value of y is 12 and the initial value of x is 0, then after $\{x' = 1, y' = 1, x \leq 2\}$ we could have $y = 14$, which takes us out of ϕ. So we need to find a way to make ϕ stronger. To find out how, we need to ask: just what goes wrong? The problem is that the constraint $x \leq 2$ does not mention y. Note that the evolution can run for time at most $2 - x$. Therefore it can cause y to grow by at most $2 - x$. Therefore, we are safe if $(2 - x) + y \leq 12$. So our stronger loop invariant should augment ϕ with $s = 1 \rightarrow y \leq 10 + x$.

The S_2 branch (third branch of the main proof) closes in a way similar to the S_0 branch. However, the S_3 branch fails in a way similar to the S_1 one does. We end up needing to prove:

$$\phi, s = 3, x < 2 \vdash [\{x' = 1, y' = -2, x \leq 2\}]\phi$$

Using the above reasoning, we see that we are safe if $(2 - x)(-2) + y \geq 1$. So let's augment ϕ with $s = 3 \rightarrow y \geq 5 + 2x$.

So our final loop invariant is

$$J \equiv y \geq 1 \wedge y \leq 12 \wedge (s = 1 \rightarrow (y \leq 10 + x)) \wedge (s = 3 \rightarrow (y \geq 5 - 2x))$$

The proof then in fact does close using J and the following differential invariants:

$$\begin{aligned} F_0 &\equiv y \geq 1 \wedge s = 0 \\ F_1 &\equiv y \geq 1 \wedge y \leq 10 + x \wedge s = 1 \\ F_2 &\equiv y \leq 12 \wedge s = 2 \\ F_3 &\equiv y \leq 12 \wedge y \geq 5 - 2x \wedge s = 3 \end{aligned}$$

Therefore learning from dead ends is fruitful for this problem.

4.2 Aircraft Roundabout Maneuver

(This example taken from [3].) Define

$$\begin{aligned} \phi &\equiv (x_1 - y_1)^2 + (x_2 - y_2)^2 = p^2 \\ \alpha &\equiv c_1 := *; c_2 := *; \omega := *; \\ &\quad d_1 := -\omega(x_2 - c_2); d_2 := \omega(x_1 - c_1); \\ &\quad e_1 := -\omega(y_2 - c_2); e_2 := \omega(y_1 - c_1) \\ \mathcal{D} &\equiv \{x_1' = d_1, x_2' = d_2, d_1' = -\omega d_2, d_2' = \omega d_1, \\ &\quad y_1' = e_1, y_2' = e_2, e_1' = -\omega e_2, e_2' = \omega e_1\} \end{aligned}$$

Here $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$ are the positions of two airplanes. The hybrid program α describes a two step process in which the planes 1) agree upon a common center c and angular velocity ω and 2) adjust their velocities (instantaneously changing airspeed and direction) to prepare for the roundabout.

We want to prove

$$\phi \vdash [\alpha; \mathcal{D}]\phi$$

which says that during the roundabout the planes remain the same distance from each other. This can be reduced to proving

$$\phi, d_1 = -\omega(x_2 - c_2),\, d_2 = \omega(x_1 - c_1), e_1 = -\omega(y_2 - c_2),\, e_2 = \omega(y_1 - c_1) \vdash [\mathcal{D}]\phi$$

And here we need to pick an invariant for the evolution $\mathcal{D}$. We might try the postcondition ϕ. Then we would need to prove $\nabla_{\mathcal{D}}\phi$, that is

$$2(x_1 - y_1)(d_1 - e_1) + 2(x_2 - y_2)(d_2 - e_2) = 0$$

or, written in vector notation,

$$2(\mathbf{x} - \mathbf{y}) \cdot (\mathbf{d} - \mathbf{e}) = 0.$$

This is not true in general. Our next attempt might be to use

$$J \equiv (x_1 - y_1)(d_1 - e_1) + (x_2 - y_2)(d_2 - e_2) = 0$$

a differential invariant, or maybe $J \wedge \phi$. We can compute

$$\nabla_{\mathcal{D}} J \equiv \|\mathbf{d} - \mathbf{e}\|^2 - \omega(\mathbf{x} - \mathbf{y})^{\perp} \cdot (\mathbf{d} - \mathbf{e}) = 0$$

where the $(\cdot)^{\perp}$ operator means "rotate 90 degrees counterclockwise." As before, this formula is not true in general. We need a stronger invariant.

One way to ensure that

$$\|\mathbf{d} - \mathbf{e}\|^2 - \omega(\mathbf{x} - \mathbf{y})^{\perp} \cdot (\mathbf{d} - \mathbf{e}) = 0$$

is to require that

$$\omega(\mathbf{x} - \mathbf{y})^{\perp} = (\mathbf{d} - \mathbf{e}).$$

As a scalar differential invariant, we would write this is

$$K \equiv -\omega(x_2 - y_2) = d_1 - e_1 \wedge \omega(x_1 - y_1) = d_2 - e_2.$$

It then is easy to see that $\nabla_{\mathcal{D}} K$ is valid and that strengthening by K allows to complete our proof. So again we find that learning from dead ends is useful.

4.3 Bouncing Ball

I now describe a proof where it is not obvious how we would benefit from dead-end knowledge. Let H, G, C be positive constants (i.e. nullary rigid functions). Assume that $C < 1$. Define

$$\mathcal{D} \equiv \{y' = v, v' = -G; y \geq 0\}$$

$$\alpha \equiv \Big(\mathcal{D}; \big((?(y = 0); v := -Cv) \cup (?\top)\big)\Big).$$

Then α^* is a hybrid program describing a bouncing ball. We are going to prove that the ball never bounces higher than the height it was dropped from. That is, we will prove

$$\cdot \vdash [v := 0; y := H; \alpha^*]y \leq H.$$

We will make use of the invariants

$$I \equiv \frac{1}{2}v^2 + Gy \leq GH$$
$$J \equiv y \geq 0 \land I.$$

(It is unclear how we would derive either of these from a proof dead-end.)
Let $\mathcal{D}_1$ be $\mathcal{D}$ augmented with I. Then we can construct a proof.

$$\dfrac{\dfrac{\dfrac{*}{J, y = 0, v_0 = -Cv \vdash y \geq 0 \land \frac{1}{2}v_0^2 + Gy \leq GH}}{J \vdash [(?y = 0; v := -Cv)]J} \quad \dfrac{\dfrac{*}{J, \top \vdash J}}{J \vdash [?\top]J}}{J \vdash [(?y = 0; v := -Cv) \cup (?\top)]J}$$

Note that we have

$$\begin{aligned} \nabla_{\mathcal{D}} I &\equiv -Gv + Gv \leq 0 \\ &\equiv \top \end{aligned}$$

Plugging in the above proof gives us

$$\dfrac{\dfrac{\dfrac{*}{J, y \geq 0 \vdash I} \quad \dfrac{*}{y \geq 0 \vdash \nabla_{\mathcal{D}} I} \quad \dfrac{\dfrac{...}{J, y \geq 0 \land I \vdash [(?y = 0; v := -Cv) \cup (?\top)]J}}{J \vdash [\mathcal{D}_1][(?y = 0; v := -Cv) \cup (?\top)]J}}{J \vdash [\mathcal{D}][(?y = 0; v := -Cv) \cup (?\top)]J}}{J \vdash [\alpha]J}$$

And plugging this in give us

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{v = 0, y = H \vdash J} \quad \dfrac{...}{J \vdash [\alpha]J} \quad \dfrac{\dfrac{*}{v = 0, y = H, J \vdash y \leq H}}{v = 0, y = H \vdash [\alpha^* \land J]y \leq H}}{v = 0, y = H \vdash [\alpha^*]y \leq H}}{v = 0 \vdash [y := H; \alpha^*]y \leq H}}{\cdot \vdash [v := 0; y := H; \alpha^*]y \leq H}$$

4.4 Train Control

Here is another example where dead-end knowledge might not be useful. Define

$$\begin{aligned}\beta_1 &\equiv s := \frac{v^2}{2b} + (\frac{A}{b} + 1)(\frac{A}{2}\epsilon^2 + \epsilon v)\\ \beta_2 &\equiv (?(m - z \leq s);\ a := -b) \cup (?(m - z \geq s);\ a := A)\\ \beta_2 &\equiv t := 0;\ \{z' = v, v' = a, t' = 1, (v \geq 0 \wedge t \leq \epsilon)\}\end{aligned}$$

We want to prove

$$(v^2 \leq 2b(m - z) \wedge b > 0 \wedge A \geq 0) \rightarrow [(\beta_1; \beta_2; \beta_3)^*](z \leq m).$$

5 Future Work

We have seen above that sometimes dead ends contain useful information. We have not addressed the question of how exactly to detect and to use such information in an automatic theorem prover. In the future I hope to investigate that issue.

References

[1] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.

[2] André Platzer. Differential dynamic logic for hybrid systems. *J Autom Reas*, 41(2):143–189, 2008.

[3] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics.* Springer, Heidelberg, 2010. To appear.

[4] André Platzer and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. *Form. Methods Syst. Des.*, 35(1):98–120, 2009.

[5] André Platzer and Jan-David Quesel. KeYmaera: A hybrid theorem prover for hybrid systems. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *IJCAR*, volume 5195 of *LNCS*, pages 171–178. Springer, 2008.