15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Ghosts & Differential Ghosts

André Platzer

Carnegie Mellon University
Lecture 12

1. Introduction

Lecture 10 on Differential Equations & Differential Invariants and Lecture 11 on Differential Equations & Proofs equipped us with powerful tools for proving properties of differential equations without having to solve them. *Differential invariants* (DI) [Pla10a, Pla12] prove properties of differential equations by induction based on the right-hand side of the differential equation, rather than its much more complicated global solution. *Differential cuts* (DC) [Pla10a, Pla12] made it possible to simply prove another property C of a differential equation and then change the dynamics of the system around so that it is restricted to never leave region C. It can be shown that differential cuts are a fundamental proof principle for differential equations [Pla12], because some properties can only be proved with differential cuts. That is the *No Differential Cut Elimination* theorem, because, unlike cuts in first-order logic, differential cuts cannot generally be eliminated but are sometimes necessary [Pla12].

Yet, it can also be shown that there are properties for which even differential cuts are not enough, but differential ghosts become necessary [Pla12]. Differential ghosts [Pla12], spooky as they may sound, turn out to be a useful proof technique for differential equations. Differential ghosts or differential auxiliaries are extra variables that are introduced into the system solely for the purpose of the proof. Differential ghosts are the differential analogue of ghost variables or auxiliary variables, which sometimes have to be added into systems for the purpose of the proof. Both ghosts and differential ghosts serve a similar intuitive purpose: remember intermediate state values so that the relation of the values at intermediate states to values at final states can be analyzed. And that is also where the somewhat surprising name comes from. Auxiliary variables are often called ghosts, because they are not really present in the actual system, but just invented to make the story more interesting or, rather, the proof more

conclusive. Ghosts give the proof a way of referring to how the state used to be that is no more. There are many reasons for introducing ghost state into a system, which will be investigated subsequently.

This lecture is based on [Pla12, Pla10b].

The most important learning goals of this lecture are:

Modeling and Control: This lecture does not have much impact on modeling and control of CPS, because, after all, the whole point of ghosts and differential ghosts is that they are only added for the purposes of the proof. However, it can still sometimes be more efficient to add such ghost and differential ghost variables into the original model right away. It is good style to mark such additional variables in the model and controller as ghost variables in order to retain the information that they do not need to be included in the final system executable.

Computational Thinking: This lecture leverages computational thinking principles for the purposes of rigorous reasoning about CPS models by analyzing how extra dimensions can simplify or enable reasoning about lower-dimensional systems. From a state space perspective, extra dimensions are a bad idea, because, e.g., the number of points on a gridded space grows exponentially in the number of dimensions. From a reasoning perspective, the important insight of this lecture is that extra state variables sometimes help and may even make reasoning possible that is otherwise impossible. One intuition why extra ghost state may help reasoning is that it can be used to consume the energy that a given dissipative system is leaking (a similar purpose that dark matter had been speculated to exist) or produce the energy that a given system consumes. The addition of such extra ghost state then enables invariants of generalized energy constants involving both original and ghost state that was not possible using only the original state. That is, ghost state may new cause energy invariants. This lecture continues the trend of generalizing important logical phenomena from discrete systems to continuous systems. The verification techniques developed in this lecture are critical for verifying some CPS models of appropriate scale and technical complexity but are not necessary for all CPS models. A secondary goal of today's lecture is to develop more intuition and deeper understandings of differential invariants and differential cuts.

CPS Skills: The focus in this lecture is on reasoning about CPS models, but there is an indirect impact on developing better intuitions for operational effects in CPS by introducing the concept of relations of state to extra ghost state. A good grasp on such relations can help with the understanding of CPS dynamics quite substantially. The reason is that ghosts and differential ghosts enable extra invariants, which enable stronger statements about what we can rely on as a CPS evolves.

2. Recap

Recall the following proof rules of differential invariants (DI), differential weakening (DW) and differential cuts (DC) for differential equations from Lecture 11 on Differential Equations & Proofs:

Note 1 (Proof rules for differential equations).

$$\text{(DI)}\ \frac{H \vdash F'^{\theta}_{x'}}{F \vdash [x' = \theta \,\&\, H]F} \qquad \text{(DW)}\ \frac{H \vdash F}{\Gamma \vdash [x' = \theta \,\&\, H]F, \Delta}$$

$$\text{(DC)}\ \frac{\Gamma \vdash [x' = \theta \,\&\, H]C, \Delta \qquad \Gamma \vdash [x' = \theta \,\&\, (H \wedge C)]F, \Delta}{\Gamma \vdash [x' = \theta \,\&\, H]F, \Delta}$$

With cuts and generalizations, earlier lectures have also shown that the following can be proved:

$$\frac{A \vdash F \quad F \vdash [x' = \theta \,\&\, H]F \quad F \vdash B}{A \vdash [x' = \theta \,\&\, H]B} \tag{1}$$

This is useful for replacing a precondition A and postcondition B by another invariant F that implies postcondition B and is implied by precondition A.

3. Arithmetic Ghosts

The easiest way to see why it sometimes makes sense to add variables into a system model is to take a look at divisions. Divisions are not officially part of real arithmetic, because divisions can be defined. For example, when a division b/c is ever mentioned in a term such as $q = b/c$, then we can characterize q to remember the value of b/c by indirectly characterizing q in terms of b and c without $/$ and then subsequently use q wherever b/c first occurred:

$$q := \frac{b}{c} \quad \rightsquigarrow \quad q := *;\ ?qc = b \quad \rightsquigarrow \quad q := *;\ ?qc = b \wedge c \neq 0$$

where $q := *$ is the nondeterministic assignment that assigns an arbitrary real number to q. The first transformation (simply written $\rightsquigarrow$) characterizes $q = b/s$ indirectly by multiplying up as $qc = b$. The second transformation then conscientiously remembers that divisions only make all the sense in the world when we avoid dividing by zero. Because divisions by zero only cause a lot of trouble. This transformation can be used when b/c occurs in the middle of a term too:

$$x := 2+\frac{b}{c}+e \quad \rightsquigarrow \quad q := *;\ ?qc = b;\ x := 2+q+e \quad \rightsquigarrow \quad q := *;\ ?qc = b \wedge c \neq 0;\ x := 2+q+e$$

Here q is called an *arithmetic ghost*, because q is an auxiliary variable that is only added to the hybrid program for the sake of defining the arithmetic quotient $\frac{b}{c}$.

4. Nondeterministic Assignments & Ghosts of Choice

The HP statement $x := *$ that has been used in Sect. 3 is a nondeterministic assignment that assigns an arbitrary real number to x. Comparing with the syntax of hybrid programs from Lecture 3 on Choice & Control, however, it turns out that such a statement is not in the official language of hybrid programs.

$$\alpha, \beta \ ::= \ x := \theta \mid ?H \mid x' = \theta \,\&\, H \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^* \tag{2}$$

What now?

One possible solution, which is the one taken in the implementation of KeYmaera [PQ08], is to simply add the nondeterministic assignment $x := *$ as a statement to the syntax of hybrid programs.

$$\alpha, \beta \ ::= \ x := \theta \mid ?H \mid x' = \theta \,\&\, H \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^* \mid \boldsymbol{x := *}$$

Consequently, nondeterministic assignments need a semantics to become meaningful:

7. $\rho(x := *) = \{(\nu, \omega) \ : \ \omega = \nu$ except for the value of x, which can be any real number$\}$

And nondeterministic assignments need proof rules so that they can be handled in proofs:

Note 2. $(\langle:*\rangle)$ $\dfrac{\exists x\, \phi}{\langle x := *\rangle \phi}$ $\qquad$ $([:*])$ $\dfrac{\forall x\, \phi}{[x := *]\phi}$

Proof rule $\langle:*\rangle$ says that there is one way of assigning an arbitrary value to x so that ϕ holds afterwards (i.e. $\langle x := *\rangle \phi$ holds) if (and only if) ϕ holds for some value of x (i.e. $\exists x\, \phi$ holds). And proof rule $[:*]$ says that ϕ holds for all ways of assigning an arbitrary value to x (i.e. $[x := *]\phi$ holds) if (and only if) ϕ holds for all values of x (i.e. $\forall x\, \phi$ holds) because x might have any such value after running $x := *$.

An alternative approach for adding nondeterministic assignments $x := *$ to hybrid programs is to reconsider whether we even have to do add a new construct for $x := *$ or whether it can already be expressed in other ways. That is, to understand whether $x := *$ is truly a new program construct or whether it can be defined in terms of the other hybrid program statements from (2). Is $x := *$ definable by a hybrid program?

Before you read on, see if you can find the answer for yourself.

According to the proof rules [:∗] and ⟨:∗⟩, nondeterministic assignments $x := *$ can be expressed equivalently by suitable quantifiers. But that does not help at all in the middle of a program, where we can hardly write down a quantifier to express that the value of x now changes.

There is another way, though. Nondeterministic assignment $x := *$ assigns any real number to x. One hybrid program that has the same effect of giving x any arbitrary real value [Pla10b, Chapter 3] is:

$$x := * \quad \overset{\text{def}}{\equiv} \quad x' = 1 \cup x' = -1 \tag{3}$$

That is not the only definition of $x := *$, though. An equivalent definition is [Pla14]:

$$x := * \quad \overset{\text{def}}{\equiv} \quad x' = 1; x' = -1$$

When working through the intended semantics of the left-hand side $x := *$ shown in case 7 above and the actual semantics of the right-hand side of (3) according to Lecture 3 on Choice & Control, it becomes clear that both sides of (3) mean the same, because they have the same reachability relation. Hence, the above definition (3) captures the intended concept of giving x any arbitrary real value, nondeterministically. And, in particular, just like if-then-else, nondeterministic assignments do not really have to be added to the language of hybrid programs, because they can already be defined. Likewise, no proof rules would have to be added for nondeterministic assignments, because there are already proof rules for the constructs used in the right-hand side of the definition of $x := *$ in (3). Since the above proof rules ⟨:∗⟩,[:∗] for $x := *$ are particularly easy, though, it is usually more efficient to include them directly, which is what KeYmaera does.

What may, at first sight, appear slightly spooky about (3), however, is that the left-hand side $x := *$ is clearly an instant change in time where x changes its value instantaneously to some arbitrary new real number. That is not quite the case for the right-hand side of (3), which involves two differential equations, which take time to follow.

The clue is that this passage of time is not observable in the state of the system. Consequently, the left-hand side of (3) really means the same as the right-hand side of (3). Remember from earlier lectures that time is not special. If a CPS wants to refer to time, it would have a clock variable t with the differential equation $t' = 1$. With such an addition, however, the passage of time t becomes observable in the value of variable t and, hence, a corresponding variation of the right-hand side of (3) would not be equivalent to $x := *$ at all (indicated by $\not\equiv$):

$$x := * \quad \not\equiv \quad x' = 1, t' = 1 \cup x' = -1, t' = 1$$

Both sides differ, because the right side exposes the amount of time t it took to get the value of x to where it should be, which, secretly, records information about the absolute value of the change that x underwent from its old to its new value. That change is something that the left-hand side $x := *$ knows nothing about.

5. Differential-algebraic Ghosts

The transformation in Sect. 3 can eliminate all divisions, not just in assignments, but also in tests and all other hybrid programs, with the notable exception of differential equations. Eliminating divisions in differential equations turns out to be a little more involved.

The following elimination using a (discrete) arithmetic ghost q is correct:

$$x' = \frac{2x}{c} \,\&\, c \neq 0 \land \frac{x+1}{c} > 0 \quad \rightsquigarrow \quad q := *; ?qc = 1; x' = 2xq \,\&\, c \neq 0 \land (x+1)q > 0$$

where the extra ghost variable q is supposed to remember the value of $\frac{1}{c}$.

The following attempt with a (discrete) arithmetic ghost q, however, would change the semantics rather radically:

$$x' = \frac{c}{2x} \,\&\, 2x \neq 0 \land \frac{c}{2x} > 0 \quad \not\rightsquigarrow \quad q := *; ?q2x = 1; x' = cq \,\&\, 2x \neq 0 \land cq > 0$$

because q then only remembers the inverse of the initial value of $2x$, not the inverse of the value of $2x$ as x evolves along the differential equation $x' = \frac{c}{2x}$. That is q has a constant value during the differential equation but, of course, the quotient $\frac{c}{2x}$ changes over time since x does.

One way to proceed is to figure out how the value of the quotient $q = \frac{1}{2x}$ changes over time as x changes by $x' = \frac{c}{2x}$. By deriving what q stands for, that results in

$$q' = \left(\frac{1}{2x}\right)' = \frac{-2x'}{4x^2} = \frac{-2\frac{c}{2x}}{4x^2} = -\frac{c}{4x^3}$$

Alas, we go unlucky here, because that gives yet another division to keep track of.

The other and entirely systematic way to proceed is to lift nondeterministic assignments q to differential equations $q' = *$ with the intended semantics that q changes arbitrarily over time while following that nondeterministic differential equation:[1]

$$q' = \frac{b}{c} \quad \rightsquigarrow \quad q' = * \,\&\, qc = b \quad \rightsquigarrow \quad q' = * \,\&\, qc = b \land c \neq 0$$

While it is slightly more complicated to give a semantics to $q' = *$, the idea behind the transformation is completely analogous to the case of discrete arithmetic ghosts:

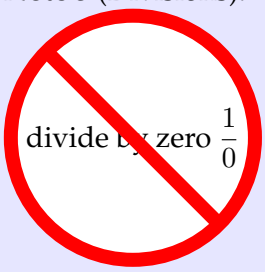
$$x' = 2 + \frac{b}{c} + e \quad \rightsquigarrow \quad x' = 2 + q + e, q' = * \,\&\, qc = b \quad \rightsquigarrow \quad x' = 2 + q + e, q' = * \,\&\, qc = b \land c \neq 0$$

Variable q is a *differential-algebraic ghost* in the sense of being an auxiliary variable in the differential-algebraic equation for the sake of defining the quotient $\frac{b}{c}$.

[1] See [Pla10b, Chapter 3] for the precise meaning of the nondeterministic differential equation $q' = *$. It is the same as the differential-algebraic constraint $\exists d\, q' = d$, but differential-algebraic constraints have not been introduced in this course so far, either. The intuition of allowing arbitrary changes of the value of q over time is fine, though, for our purposes.

Together with the reduction of divisions in discrete assignments from Sect. 3, plus the insight that divisions in tests and evolution domain constraints can always be rewritten to division-free form, this gives a (rather sketchy) proof showing that hybrid programs and differential dynamic logic do not need divisions [Pla10b]. The advantage of eliminating divisions this way is that differential dynamic logic does not need special precautions for divisions and that the handling of zero divisors is made explicit in the way the divisions are eliminated from the formulas. In practice, however, it is still useful to use divisions, yet great care has to be exercised to make sure that no inadvertent divisions by zero could ever cause troublesome singularities.

Note 3 (Divisions).



Whenever dividing, exercise great care not to accidentally divide by zero, for that will cause quite some trouble. More often than not, this trouble corresponds to missing requirements in the system. For example $\frac{v^2}{2b}$ may be a good stopping distance when braking to a stop from initial velocity v, except when $b = 0$, which corresponds to having no brakes at all.

6. Discrete Ghosts

All the ghost variables so far were introduced to define operators such as divisions or nondeterministic assignments $x := *$. There are other reasons for using auxiliary alias ghost variables, though.

The discrete way of adding ghost variables is to introduce a new ghost variable y into a proof that remembers the value of a term θ. This can be useful in a proof in order to have a name, y, that recalls the value of θ later on in the proof, especially when the value of θ changes subsequently during the execution of hybrid programs α in the remaining modalities, which makes it possible to relate the value of θ before and after the run of that hybrid program α.

Lemma 1 (Discrete ghosts). *The following is a sound proof rule for introducing an auxiliary variable or (discrete)* ghost *y:*

$$\text{(IA)}\ \frac{\Gamma \vdash [y := \theta]\phi, \Delta}{\Gamma \vdash \phi, \Delta}$$

where y is a new program variable.

The fact that proof rule IA is sound can be explained easily based on the soundness of the substitution axiom [:=] from Lecture 5 on Dynamical Systems & Dynamic Axioms. The assignment axiom [:=] proves validity of

$$\phi \leftrightarrow [y := \theta]\phi$$

because the fresh variable y does not occur in ϕ. Hence, discrete ghost rule IA just applies the substitution axiom backwards to introduce a ghost variable y that was not there before.

Discrete ghosts can be interesting when ϕ contains modalities that change variables in θ for y can then remember the value that θ had before that change. For example:

$$\dfrac{\dfrac{xy-1=0 \vdash [c:=xy][x'=x,y'=-y]xy=1}{xy-1=0 \vdash [x'=x,y'=-y]xy=1}\,\text{IA}}{\vdash xy-1=0 \to [x'=x,y'=-y]xy=1}\,\to\text{r}$$

This sequent derivation memorizes the value that the interesting term xy had before the differential equation started in the ghost variable c. It is a bit hard to complete the proof, because substituting c away using the assignment rule [:=]r would undo the pleasant effect that the IA rule had, because the whole point of the fresh variable c is that it does not occur elsewhere.[2] So the only way the proof can make progress is by applying a proof rule to the differential equation. Unfortunately, the sequent calculus from Lecture 6 on Truth & Proof focuses the application of proof rules to the top-level of sequents. That is usually an advantage but now a disadvantage. For that reason, KeYmaera uses a concept called updates that postpone the application of the substitution rule [:=]r until all modalities are gone.

[2] This potentially surprising phenomenon happens in some form or other for other ghosts as well, because, the whole point of ghosts is to compute something that the original model and property do not depend on. So, sufficiently sophisticated forms of dead-code elimination would get rid of ghosts, which would be counterproductive for the proof.

Note 5 (Excursion: Updates). *KeYmaera postpones the substitution resulting from an assignment according to rule [:=]r,[:=]l,⟨:=⟩r,⟨:=⟩l if the postcondition is not a first-order formula but involves modalities with HPs. What this corresponds to is, essentially to leave the assignment as is and apply proof rules to the postcondition, but only in this particular case of a prefix of assignments! Because that would be a bit confusing without further notice, KeYmaera changes the notation slightly and turns an assignment into what it calls an* update.

$$(R7)\ \frac{\{x:=\theta\}\phi}{[x:=\theta]\phi} \qquad (R8)\ \frac{\phi_x^\theta}{\{x:=\theta\}\phi}$$

The meaning of the formula $\{x:=\theta\}\phi$ in the premise of R7 is exactly the same as the formula $[x:=\theta]\phi$ in the conclusion of R7. The notation $\{x:=\theta\}\phi$ is only meant as a reminder for the user that KeYmaera decided to put the handling of the assignment by substitution on hold until the postcondition ϕ looks more civilized (meaning: first-order). KeYmaera collects all the state changes in such an update (or a list of updates). KeYmaera will then, essentially, just carry the update prefix $\{x:=\theta\}$ around with it and apply the sequent proof rules directly to the respective postcondition ϕ after the update $\{x:=\theta\}$ until the substitution that $\{x:=\theta\}$ is waiting for can ultimately be applied (R8) which will make the update disappear again. Thus, KeYmaera splits the assignment rule [:=] into two parts: the conversion of assignments to updates R7 followed by the application of updates as substitutions R8:

$$\cfrac{\cfrac{\Gamma\vdash \phi_x^\theta,\Delta}{\Gamma\vdash \{x:=\theta\}\phi,\Delta}\,R8}{\Gamma\vdash [x:=\theta]\phi,\Delta}\,R7$$

Similarly for $\langle x:=\theta\rangle\phi$.

Observe that postponing the substitution of assignments in $[x:=\theta]\phi$ may be necessary when the postcondition ϕ contains further modalities with loops assigning to x or differential equations binding x'. We also need to be careful to not leave updates lurking around for premises that need the sequent context Γ,Δ removed for soundness reasons, because both Γ,Δ and an update $\{x:=\theta\}$ represent knowledge about the symbolic current state.

More information on updates can be found in [Pla08, Pla10b, Chapter 2.2,2.3,2.5].

Continuing the sequent proof above with updates to postpone the handling of the first assignment leads to the following proof:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{\vdash 0=xy+x(-y)}\,*}{\vdash (0=x'y+xy')^{x\ -y}_{x'\ y'}}\,\mathbb{R}}{xy-1=0\vdash\{c:=xy\}[x'=x,y'=-y]c=xy}\,DI}{xy-1=0\vdash\{c:=xy\}[x'=x,y'=-y]xy=1}\,[]gen'}{xy-1=0\vdash[c:=xy][x'=x,y'=-y]xy=1}\,R7}{xy-1=0\vdash[x'=x,y'=-y]xy=1}\,IA}{\vdash xy-1=0\to[x'=x,y'=-y]xy=1}\,\to r$$

▷

The generalization step $[]gen'$ leads to a second premise that has been elided (marked by $\triangleright$) and proves, because $c = 1$ is an easily provable additional differential invariant, because the discrete ghost c starts out as 1 initially by the antecedent and never changes its value. This particular property also proves directly quite easily, but the proof technique of discrete ghosts is of more general interest.

See ≪Proof using discrete ghosts≫

7. Proving Bouncing Balls with Sneaky Solutions

Recall a dℒ formula for the falling ball part of the bouncing ball proof from Lecture 7 on Control Loops & Invariants, which was based on an argument in Lecture 4:

$$2gx = 2gH - v^2 \wedge x \geq 0 \rightarrow [x' = v, v' = -g \,\&\, x \geq 0](2gx = 2gH - v^2 \wedge x \geq 0) \tag{4}$$

Recall the abbreviation

$$(x'' = -g \,\&\, x \geq 0) \equiv (x' = v, v' = -g \,\&\, x \geq 0)$$

Lecture 7 proved dℒ formula (4) using the solutions of the differential equation with the solution proof rule $[']$. Yet, dℒ formula (4) can also be proved with a mix of differential invariants, differential cuts and differential weakening, instead:

$$\text{DC}\,\dfrac{\text{DI}\,\dfrac{\mathbb{R}\,\dfrac{*}{x \geq 0 \vdash 2gv = -2v(-g)}}{x \geq 0 \vdash (2gx' = -2vv')^{v,\,-g}_{x',\,v'}} \quad \text{DW}\,\dfrac{\text{ax}\,\dfrac{*}{x \geq 0 \wedge 2gx = 2gH - v^2 \vdash 2gx = 2gH - v^2 \wedge x \geq 0}}{2gx = 2gH - v^2 \vdash [x'' = -g \,\&\, x \geq 0 \wedge 2gx = 2gH - v^2](2gx = 2gH - v^2 \wedge x \geq 0)}}{2gx = 2gH - v^2 \vdash [x'' = -g \,\&\, x \geq 0](2gx = 2gH - v^2 \wedge x \geq 0)}$$

Note that differential weakening (DW) works for proving the postcondition $x \geq 0$, but DI would not work for proving $x \geq 0$, because its derivative is $(x \geq 0)' \equiv v \geq 0$, which is not an invariant of the bouncing ball since its velocity ultimately becomes negative when it is falling again according to gravity.

The above proof is very elegant and has notably easier arithmetic than the arithmetic requires when working with solutions of the bouncing ball in earlier lectures.

> **Note 6** (Differential invariants lower degrees). *Differential invariants DI work by differentiation, which lowers polynomial degrees. The solution proof rule $[']$ works with solutions, which ultimately integrate the differential equation and, thus, increase the degrees. The computational complexity of the resulting arithmetic is, thus, often in favor of differential invariants even in cases where the differential equations can be solved so that the solution rule $[']$ would be applicable.*

Besides the favorably simple arithmetic coming from differential invariants, the other reason why the proof worked so elegantly is that the invariant $2gx = 2gH - v^2 \wedge x \geq 0$ was a clever choice that we came up with in a creative way in Lecture 4. There is nothing wrong with being creative. On the contrary!

But it also pays off to be systematic and develop a rich toolbox of techniques for proving properties of differential equations. Is there a way to prove (4) without such a distinctively clever invariant that works as a differential invariant right away? Yes, of course, because (4) can even be proved using solutions [′]. But it turns out that interesting things happen when we systematically try to understand how to make a proof happen that does not use the solution rule [′] and, yet, still uses solution-based arguments. Can you conceive a way to use solutions for differential equations without invoking the actual solution rule [′]?

Before you read on, see if you can find the answer for yourself.

8. Exploiting Differential Ghosts for Falling Balls

Note 7 (Ghost solutions). *Whenever there is a solution of a differential equation that we would like to make available to a proof without using the solution rule $[']r$, a differential cut and subsequent differential invariant can be used to cut the solution as an invariant into the system. The tricky part is that solutions depend on time, and time may not be part of the differential equation system. If there is no time variable, however, a differential ghost first needs to be added that pretends to be time.*

Consider dℒ formula (4) again, which turns into

$$A_{x,v} \vdash [x'' = -g \,\&\, x \geq 0]B_{x,v} \tag{4}$$

using the abbreviations:

$$\begin{aligned} A_{x,v} &\overset{\text{def}}{\equiv} 2gx = 2gH - v^2 \wedge x \geq 0 \\ B_{x,v} &\overset{\text{def}}{\equiv} 2gx = 2gH - v^2 \wedge x \geq 0 \\ (x'' = -g) &\overset{\text{def}}{\equiv} (x' = v, v' = -g) \end{aligned}$$

The proof begins by introducing a discrete ghost v_0 remembering the initial velocity of the bouncing ball and proceeds by adding a differential ghost t for the time variable with derivative $t' = 1$ so that the solution $v = v_0 - tg$ can be differentially cut into the system and proved to be differentially invariant:

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{x \geq 0 \vdash -g = -1g}\,\mathbb{R}}{x \geq 0 \vdash (v' = -t'g)^{-g}_{v'}{}^{1}_{t'}}}{A_{x,v} \vdash \{v_0 := v\}[x'' = -g, t' = 1 \,\&\, x \geq 0]v = v_0 - tg}\,\text{DI} \qquad A_{x,v} \vdash \{v_0 := v\}[x'' = -g, t' = 1 \,\&\, x \geq 0 \wedge v = v_0 - tg]B_{x,v}}{\dfrac{\dfrac{\dfrac{A_{x,v} \vdash \{v_0 := v\}[x'' = -g, t' = 1 \,\&\, x \geq 0]B_{x,v}}{A_{x,v} \vdash \{v_0 := v\}[x'' = -g \,\&\, x \geq 0]B_{x,v}}\,\text{DA}}{A_{x,v} \vdash [v_0 := v][x'' = -g \,\&\, x \geq 0]B_{x,v}}\,\text{R7}}{A_{x,v} \vdash [x'' = -g \,\&\, x \geq 0]B_{x,v}}\,\text{IA}}\,\text{DC}$$

where the proof step marked DA (for differential auxiliaries or differential ghosts) introduces new variable t with derivative 1 as a differential ghost into the system.[3] Observe how the differential invariant step DI made the sequent context as well as the update $\{v_0 := v\}$ disappear, which is generally important for soundness.

The left premise in the above proof has been closed by arithmetic. The right premise in the above proof proves as follows by first introducing yet another discrete ghost x_0

[3] When discussing the differential ghost proof rule DA in a more general form later on, we will see that DA introduces an extra left premise, which is omitted in this proof (marked by ◁). That additional premise, however, proves easily because $B_{x,v} \leftrightarrow \exists t\, B_{x,v}$ is rather trivially valid in first-order logic, as the fresh variable t does not even occur in $B_{x,v}$ at all here (vacuous quantification).

with DA that remembers the initial position so that it can be referred to in the solution. The solution $x = x_0 + v_0 t - \frac{g}{2}t^2$ can then be differentially cut into the system by DC and proved to be differentially invariant by DI:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x \geq 0 \land v = v_0 - tg \vdash v = v_0 - 2\frac{g}{2}t}\,ax}{x \geq 0 \land v = v_0 - tg \vdash (x' = v_0 t' - 2\frac{g}{2}tt')^{v}_{x'\ t'}\ 1}\,\text{DI}}{A_{x,v} \vdash \{x_0 := x; v_0 := v\}[x'' = -g, t' = 1 \,\&\, x \geq 0 \land v = v_0 - tg]x = x_0 + v_0 t - \frac{g}{2}t^2 \triangleright}\,\text{DC}}{A_{x,v} \vdash \{x_0 := x; v_0 := v\}[x'' = -g, t' = 1 \,\&\, x \geq 0 \land v = v_0 - tg]B_{x,v}}\,\text{IA}}{A_{x,v} \vdash \{v_0 := v\}[x'' = -g, t' = 1 \,\&\, x \geq 0 \land v = v_0 - tg]B_{x,v}}$$

The differential cut proof step marked DC has a second premise using the cut which is elided above (marked by $\triangleright$) and proves as follows:

$$\text{DW}\dfrac{x \geq 0 \land v = v_0 - tg \land x = x_0 + v_0 t - \frac{g}{2}t^2 \vdash B_{x,v}}{A_{x,v} \vdash \{x_0 := x; v_0 := v\}[x'' = -g, t' = 1 \,\&\, x \geq 0 \land v = v_0 - tg \land x = x_0 + v_0 t - \frac{g}{2}t^2]B_{x,v}}$$

The resulting arithmetic can be proved by real arithmetic with enough care, but it has a twist! First of all, the arithmetic can be simplified substantially using the equality substitution rule =r from Lecture 6 to replace v by $v_0 - tg$ and replace x by $x_0 + v_0 t - \frac{g}{2}t^2$ and use subsequent weakening (Wl) to get rid of both equations after use. This simplification reduces the computational complexity of real arithmetic a lot:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash 2g(x_0 + v_0 t - \frac{g}{2}t^2) = 2gH - (v_0 - tg)^2}}{x \geq 0 \vdash 2g(x_0 + v_0 t - \frac{g}{2}t^2) = 2gH - (v_0 - tg)^2}\,\text{Wl} \quad \dfrac{}{x \geq 0 \vdash x \geq 0}\,ax}{x \geq 0 \vdash 2g(x_0 + v_0 t - \frac{g}{2}t^2) = 2gH - (v_0 - tg)^2 \land x \geq 0}\,\land\text{r}}{x \geq 0, v = v_0 - tg, x = x_0 + v_0 t - \frac{g}{2}t^2 \vdash 2g(x_0 + v_0 t - \frac{g}{2}t^2) = 2gH - (v_0 - tg)^2 \land x \geq 0}\,\text{Wl}}{x \geq 0, v = v_0 - tg, x = x_0 + v_0 t - \frac{g}{2}t^2 \vdash 2gx = 2gH - (v_0 - tg)^2 \land x \geq 0}\,\text{=r}}{x \geq 0, v = v_0 - tg, x = x_0 + v_0 t - \frac{g}{2}t^2 \vdash 2gx = 2gH - v^2 \land x \geq 0}\,\text{=r}}{x \geq 0 \land v = v_0 - tg \land x = x_0 + v_0 t - \frac{g}{2}t^2 \vdash 2gx = 2gH - v^2 \land x \geq 0}\,\land\text{l}$$

Observe how this use of equality substitution and weakening helped simplify the arithmetic complexity of the formula substantially and even helped to eliminate a variable (v) right away. This can be useful to simplify arithmetic in many other cases as well. Both eliminating variables as well as applying and hiding equations right away can often simplify the complexity of handling real arithmetic. The arithmetic in the remaining left branch

$$2g\left(x_0 + v_0 t - \frac{g}{2}t^2\right) = 2gH - (v_0 - tg)^2$$

expands by polynomial arithmetic and cancels as follows:

$$2g\left(x_0 + \cancel{v_0 t} - \cancel{\frac{g}{2}t^2}\right) = 2gH - v_0^2 + \cancel{2v_0 tg} + \cancel{t^2 g^2}$$

Those cancellations simplify the arithmetic, leaving the remaining condition:

$$2gx_0 = 2gH - v_0^2 \tag{5}$$

Indeed, this relation characterizes exactly how H, which turns out to have been the maximal height, relates to the initial height x_0 and initial velocity v_0. In the case of initial velocity $v_0 = 0$, for example, the equation (5) collapses to $x_0 = H$, i.e. that H is the initial height in that case. Consequently, the computationally easiest way of proving the resulting arithmetic is to first prove by a differential cut DC that (5) is a trivial differential invariant, resulting in a proof of (4); see Exercise 2.

Yet, as we go through all proof branches again to check that we really have a proof, however, we notice a subtle but blatant oversight. Can you spot it, too?

The very first left-most branch with the initial condition for the differential invariant $v = v_0 = tg$ does not, actually, prove. The catch is that we silently assumed $t = 0$ to be the initial value for the new clock t, but that our proof did not actually say so. Oh my, what could be done about that now?

Before you read on, see if you can find the answer for yourself.

Discrete ghosts to the rescue! Even though we do not know the initial value of the differential ghost t, we can simply use a discrete ghost again to call it t_0 and get on with it. Will that work? Can you find it out? Or should we start a revision of the proof to find out?

$$\text{IA}\,\dfrac{\text{R7}\,\dfrac{\text{DC}\,\dfrac{A_{x,v} \vdash \{v_0 := v\}\{t_0 := t\}[x'' = -g, t' = 1 \,\&\, x \geq 0]v = v_0 - (t - t_0)g \qquad A_{x,v} \vdash \{v_0 := v\}\{t_0 := t\}[x'' = -g, t' = 1 \,\&\, x \geq 0 \land v = v_0 - (t - t_0)g]B_{x,v}}{A_{x,v} \vdash \{v_0 := v\}\{t_0 := t\}[x'' = -g, t' = 1 \,\&\, x \geq 0]B_{x,v}}}{A_{x,v} \vdash \{v_0 := v\}[t_0 := t][x'' = -g, t' = 1 \,\&\, x \geq 0]B_{x,v}}}{A_{x,v} \vdash \{v_0 := v\}[x'' = -g, t' = 1 \,\&\, x \geq 0]B_{x,v}}$$

As this proof shows, everything works as expected as long as we realize that this requires a change of the invariants used for the differential cuts. The solution of the velocity to differentially cut in will be $v = v_0 - (t - t_0)g$ and the solution of the position to differentially cut in subsequently will be $x = x_0 + v_0(t - t_0) - \frac{g}{2}(t - t_0)^2$.

See ≪Proof of falling balls≫

For the case of the bouncing ball, this proof was unnecessarily complicated, because the solution rule ['] could have been used instead right away, instead. Yet, even if this particular proof was more involved, the arithmetic ended up being nearly trivial in the end (which Note 6 already observed to hold in general for differential invariant proofs). But the same proof technique of adding differential ghosts and discrete ghosts as needed can be pretty useful in more complicated systems.

Note 8 (On the utility of ghosts). *Adding differential ghosts and discrete ghosts as needed can be useful in more complicated systems that do not have computable solutions, but in which other relations between initial (or intermediate) and final state can be proved. The same technique can also be useful for cutting in solutions when only part of a differential equation system admits a polynomial solution.*

For example, the differential equation system $d' = \omega e, e' = -\omega d, v' = a$ is difficult, because it has non-polynomial solutions. Still, one part of this differential equation, the velocity $v' = a$, is easily solved. Yet, the solution rule ['][r] is not applicable, because no real arithmetic solution of the whole differential equation system exists (except when $\omega = 0$). Regardless, after suitable discrete ghosts and differential ghosts for adding a clock $t' = 1$, a differential cut with the solution $v = v_0 + at$ of $v' = a$ adds this precise knowledge about the time-dependent change of the variable v to the evolution domain for subsequent use in the proof.

9. Differential Ghosts

The proof technique of differential ghosts is not limited to adding the differential equation $t' = 1$ for time, but can add other differential equations $y' = \eta$ into the differential equation system as well. Besides, the invariant to prove can very well be modified to make use of the additional ghost variable y by referring to it, which did not happen in the above proof, in which the postcondition $B_{x,v}$ remained unchanged (see 3).

Lemma 2 (Differential ghosts). *The following is a sound proof rule* differential auxiliaries *(DA) for introducing auxiliary differential variables or* differential ghosts *[Pla12]:*

$$\text{(DA)}\ \frac{\Gamma \vdash F \leftrightarrow \exists y\, G, \Delta \quad \Gamma, G \vdash [x' = \theta, y' = \eta \,\&\, H]G, \Delta}{\Gamma, F \vdash [x' = \theta \,\&\, H]F, \Delta}$$

where y new and $y' = \eta, y(0) = y_0$ has a global solution y on H for each y_0.

Rule DA is applicable if y is a new variable and the new differential equation $y' = \eta$ has global solutions on H (e.g., because term η satisfies a Lipschitz condition [Wal98, Proposition 10.VII], which is definable in first-order real arithmetic and thus decidable). Without that condition, adding $y' = \eta$ could limit the duration of system evolutions incorrectly. In fact, it would be sufficient for the domains of definition of the solutions of $y' = \eta$ to be no shorter than those of x. Soundness is easy to see, because precondition F implies G for some choice of y (left premise). Yet, for any y, G is an invariant of the extended dynamics (right premise). Thus, G always holds after the evolution for some y (its value can be different than in the initial state), which still implies F (left premise). Since y is fresh and its differential equation does not limit the duration of solutions of x on H, this implies the conclusion. Since y is fresh, y does not occur in H, and, thus, its solution does not leave H, which would incorrectly restrict the duration of the evolution as well.

Intuitively, rule DA can help proving properties, because it may be easier to characterize how x changes in relation to an auxiliary differential ghost variable y with a suitable differential equation ($y' = \eta$) compared to understanding the change of x in isolation.

10. Spooky Ghosts

In fact, differential ghosts even give us a, shockingly spooky, way of generating differential equations for differential ghosts on the fly as needed for proofs to work out. That might sound scary but is amazingly useful. To see how it works, invent your own differential ghost $y' = \square$ with a still-unspecified right-hand side $\square$, which is nothing but a common spooky cloud, and just keep "proving" as if nothing had happened:

$$\dfrac{\dfrac{*}{\mathbb{R}\vdash x>0 \leftrightarrow \exists y\, xy^2 = 1} \quad \dfrac{\dfrac{\dfrac{\text{could prove if } \square = \frac{y}{2}}{\vdash -xy^2 + 2xy\square = 0}}{\vdash (x'y^2 + x2yy' = 0)^{-x}_{x'\ y'}\ \square}}{\text{DI}\ xy^2 = 1 \vdash [x' = -x, y' = \square]xy^2 = 1}}{\text{DA}\quad x > 0 \vdash [x' = -x]x > 0}$$

The right premise could prove if only $\square$ were chosen to be $\frac{y}{2}$, in which case the premise $-xy^2 + 2xy\square = 0$ is quite easily proved. That, of course, was a bit too

spooky for the soundness-loving truth-connoisseur. So let's instantiate the spooky cloud with its concrete choice $\frac{y}{2}$ and start all over with a proper proof:

$$\dfrac{\dfrac{*}{\mathbb{R}\vdash x>0\leftrightarrow\exists y\, xy^2=1} \qquad \dfrac{\dfrac{*}{\mathbb{R}\vdash -xy^2+2xy\frac{y}{2}=0}}{\vdash (x'y^2+x2yy'=0)^{-x\ \frac{y}{2}}_{x'\ y'}}}{\text{DI}\ xy^2=1\vdash[x'=-x,y'=\frac{y}{2}]xy^2=1}}{\text{DA}\ \ x>0\vdash[x'=-x]x>0}$$

Fortunately, this proper sequent proof confirms the suspicion of a proof that we developed above. In that sense, all is fair in how we come up with a proof, even if we use spooky ghost arguments involving .[4] But in the end, it is crucial to conduct a proper proof with sound proof rules to ensure the conclusion is valid.

It can be shown [Pla12] that there are properties such as this one that crucially need differential ghosts (or differential auxiliaries) to prove.

11. Summary

The major lesson from today's lecture is that it can sometimes be easier to relate a variable to its initial value or to other quantities than to understand its value in isolation. Ghosts, in their various forms, let us achieve that by adding auxiliary variables into the system dynamics, so that the values of the original variables of interest can be related to the values of the ghosts. Sometimes such ghosts are even necessary to prove properties. As a workaround, it might help to rewrite the original model so that it already includes the ghost variables, preferably marked as ghosts in the model. The phenomenon that relations between state and ghost variables are sometimes easier to prove than just standalone properties of state variables applies in either case. This lecture shines a light on the power of relativity theory in the sense of relating variables to one another.

This lecture also showed how properties of differential equations can be proved using solution-like arguments if only part of the differential equation system can be solved.

A. Axiomatic Ghosts

This section is devoted to yet another kind of ghosts: axiomatic ghosts. While less important for simple systems, axiomatic ghosts are the way to go for systems that involve special functions such as $\sin, \cos$ etc.

When neglecting wind, gravitation, and so on, which is appropriate for analysing cooperation in air traffic control [TPS98], the in-flight dynamics of an aircraft at x can

[4] Of course, is not quite as spooky as one might suspect. It can be made rigorous with term variables.

be described by the following differential equation system; see [TPS98] for details:

$$x_1' = v\cos\vartheta \qquad x_2' = v\sin\vartheta \qquad \vartheta' = \omega. \tag{6}$$

That is, the linear velocity v of the aircraft changes both positions x_1 and x_2 in the (planar) direction corresponding to the orientation ϑ the aircraft is currently heading toward. Further, the angular velocity ω of the aircraft changes the orientation ϑ of the aircraft.

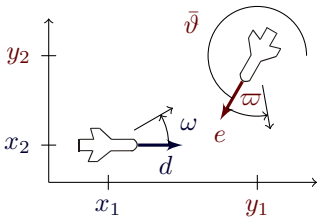


Figure 1: Aircraft dynamics

Unlike for straight-line flight ($\omega = 0$), the nonlinear dynamics in (6) is difficult to analyse [TPS98] for curved flight ($\omega \neq 0$), especially due to the trigonometric expressions which are generally undecidable. Solving (6) requires the Floquet theory of differential equations with periodic coefficients [Wal98, Theorem 18.X] and yields mixed polynomial expressions with multiple trigonometric functions. A true challenge, however, is the need to verify properties of the states that the aircraft reach by following these solutions, which requires proving that complicated formulas with mixed polynomial arithmetic and trigonometric functions hold true for all values of state variables and all possible evolution durations. However, quantified arithmetic with trigonometric functions is undecidable by Gödel's incompleteness theorem [Göd31].

To obtain polynomial dynamics, we axiomatize the trigonometric functions in the dynamics differentially and reparametrize the state correspondingly. Instead of angular orientation ϑ and linear velocity v, we use the linear speed vector

$$d = (d_1, d_2) := (v\cos\vartheta, v\sin\vartheta) \in \mathbb{R}^2$$

which describes both the linear speed $\|d\| := \sqrt{d_1^2 + d_2^2} = v$ and the orientation of the aircraft in space; see Figs. 1 and 2. Substituting this coordinate change into differential equations (6), we immediately have $x_1' = d_1$ and $x_2' = d_2$. With the coordinate change, we further obtain differential equations for d_1, d_2 from differential equation system (6) by simple symbolic differentiation:

$$\begin{aligned} d_1' &= (v\cos\vartheta)' = v'\cos\vartheta + v(-\sin\vartheta)\vartheta' = -(v\sin\vartheta)\omega = -\omega d_2, \\ d_2' &= (v\sin\vartheta)' = v'\sin\vartheta + v(\cos\vartheta)\vartheta' \quad = \quad (v\cos\vartheta)\omega = \quad \omega d_1. \end{aligned}$$

The middle equality holds for constant linear velocity ($v' = 0$), which we assume, because only limited variations in linear speed are possible and cost-effective during the

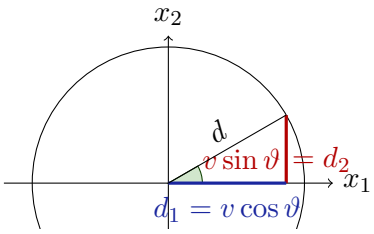


Figure 2: Reparametrize for differential axiomatization

flight [TPS98, LLL00] so that angular velocity ω is the primary control parameter in air traffic control. Hence, equations (6) can be restated as the following differential equation $\mathcal{F}(\omega)$:

$$x_1' = d_1 \,,\, x_2' = d_2 \,,\, d_1' = -\omega d_2 \,,\, d_2' = \omega d_1 \qquad (\mathcal{F}(\omega))$$
$$y_1' = e_1 \,,\, y_2' = e_2 \,,\, e_1' = -\varrho e_2 \,,\, e_2' = \varrho e_1 \qquad (\mathcal{G}(\varrho))$$

Differential equation $\mathcal{F}(\omega)$ expresses that position $x = (x_1, x_2)$ changes according to the linear speed vector $d = (d_1, d_2)$, which in turn rotates according to ω. Simultaneous movement together with a second aircraft at $y \in \mathbb{R}^2$ having linear speed $e \in \mathbb{R}^2$ (also indicated with angle $\bar{\vartheta}$ in Fig. 1) and angular velocity ϱ corresponds to the differential equation $\mathcal{F}(\omega), \mathcal{G}(\varrho)$. Differential equations capture simultaneous dynamics of multiple traffic agents succinctly using conjunction.

By this *differential axiomatization*, we thus obtain polynomial differential equations. Note, however, that their solutions still involve the same complicated nonlinear trigonometric expressions so that solutions still give undecidable arithmetic [Pla10b, Appendix B]. Note that differential invariant type arguments work with the differential equations themselves and not with their solutions, so that differential axiomatization actually helps proving properties, because the solutions are still as complicated as they have always been, but the differential equations become easier

The same technique helps when handling other special functions in other cases by differential axiomatization.

Exercises

Exercise 1. Augment the discrete ghost proofs in Sect. 6 to a full sequent proof of

$$xy - 1 = 0 \to [x' = x, y' = -y]xy = 1$$

Exercise 2. Augment the proofs in this lecture as described to obtain a full sequent proof of (4). Your advised to find a big sheet of paper, first.

References

[Göd31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Mon. hefte Math. Phys.*, 38:173–198, 1931.

[LLL00] Carolos Livadas, John Lygeros, and Nancy A. Lynch. High-level modeling and analysis of TCAS. *Proc. IEEE - Special Issue on Hybrid Systems: Theory & Applications*, 88(7):926–947, 2000.

[Pla08] André Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reas.*, 41(2):143–189, 2008. doi:10.1007/s10817-008-9103-8.

[Pla10a] André Platzer. Differential-algebraic dynamic logic for differential-algebraic programs. *J. Log. Comput.*, 20(1):309–352, 2010. doi:10.1093/logcom/exn070.

[Pla10b] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14509-4.

[Pla12] André Platzer. The structure of differential invariants and differential cut elimination. *Logical Methods in Computer Science*, 8(4):1–38, 2012. doi:10.2168/LMCS-8(4:16)2012.

[Pla14] André Platzer. Differential game logic. *CoRR*, abs/1408.1980, 2014. arXiv:1408.1980.

[PQ08] 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. doi:10.1007/978-3-540-71070-7_15.

[TPS98] Claire Tomlin, George J. Pappas, and Shankar Sastry. Conflict resolution for air traffic management: a study in multi-agent hybrid systems. *IEEE T. Automat. Contr.*, 43(4):509–521, 1998.

[Wal98] Wolfgang Walter. *Ordinary Differential Equations*. Springer, 1998.