15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Foundations of Cyber-Physical Systems

André Platzer

Carnegie Mellon University

Lecture 0

1 Overview

Cyber-physical systems (CPSs) combine cyber capabilities (computation and/or communication) with physical capabilities (motion or other physical processes). Cars, aircraft, and robots are prime examples, because they move physically in space in a way that is determined by discrete computerized control algorithms. Designing these algorithms to control CPSs is challenging due to their tight coupling with physical behavior. At the same time, it is vital that these algorithms be correct, since we rely on CPSs for safety-critical tasks like keeping aircraft from colliding. In this course we will strive to answer the fundamental question posed by Jeannette Wing:

> ”How can we provide people with cyber-physical systems they can bet their lives on?”

Students who successfully complete this course will:

- Understand the core principles behind CPSs.
- Develop models and controls.
- Identify safety specifications and critical properties of CPSs.
- Understand abstraction and system architectures.
- Learn how to design by invariant.
- Reason rigorously about CPS models.
- Verify CPS models of appropriate scale.

- Understand the semantics of a CPS model.
- Develop an intuition for operational effects.

The cornerstone of our course design are hybrid programs (HPs), which capture relevant dynamical aspects of CPSs in a simple programming language with a simple semantics. One important aspect of HPs is that they directly allow the programmer to refer to real-valued variables representing real quantities and specify their dynamics as part of the HP.

This course will give you the required skills to formally analyze the CPSs that are all around us – from power plants to pace makers and everything in between – so that when you contribute to the design of a CPS, you are able to understand important safety-critical aspects and feel confident designing and analyzing system models. It will provide an excellent foundation for students who seek industry positions and for students interested in pursuing research.

2 Course Materials

Course material will be made available on the course web page.[1] There also is an optional textbook:

- André Platzer, *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, 2010.

More information on the design of the undergraduate course *Foundations of Cyber-Physical Systems* can be found in the Course Syllabus.[2]

[1] http://symbolaris.com/course/fcps13.html

[2] http://symbolaris.com/course/15424-syllabus.pdf

3 Lectures

These course consists of the following sequence of lectures (lecture notes are hyperlinked):

1. Cyber-physical systems: introduction
2. Differential equations & domains
3. Choice & control
4. Safety & contracts
5. Dynamical systems & dynamic axioms
6. Truth & proof
7. Control loops & invariants
8. Events & delays
9. Proofs & arithmetic
10. Differential equations & differential invariants
11. Differential equations & proofs
12. Dynamic logic & dynamical systems
13. Differential invariants & proof theory
14. Ghosts & differential ghosts
15. Trains & proofs
16. Differential & temporal logic
17. Differential & temporal proofs
18. Virtual substitution & real equations
19. Virtual substitution & real arithmetic
20. Hybrid systems & games
21. Winning strategies & regions
22. Winning & proving hybrid games
23. Game proofs & separations
24. Logical theory & completeness
25. Logical foundations of CPS

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Differential Equations & Domains

André Platzer

Carnegie Mellon University

Lecture 2

1. Introduction

In the last lecture, we have learned about the characteristic features of *cyber-physical systems* (CPS): they combine cyber capabilities (computation and/or communication) with physical capabilities (motion or other physical processes). Cars, aircraft, and robots are prime examples, because they move physically in space in a way that is determined by discrete computerized control algorithms. Designing these algorithms to control CPSs is challenging due to their tight coupling with physical behavior. At the same time, it is vital that these algorithms be correct, since we rely on CPSs for safety-critical tasks like keeping aircraft from colliding.

Since CPS combine cyber and physical capabilities, we need to understand both to understand CPS. It is not enough to understand both in isolation, though, because we also need to understand how the cyber and the physics work together, i.e. what happens when they interface and interact, because this is what CPSs are all about.

You already have experience with models of computation and algorithms for the cyber part of CPS, because you have seen the use of programming languages for computer programming in previous courses. In CPS, we do not program computers, but program CPS instead. So we program computers that interact with physics to achieve their goals. In this lecture, we study models of physics and the most elementary part of how they can interact with cyber. Physics by and large is obviously a deep subject. But for CPS one of the most fundamental models of physics is sufficient, that of ordinary differential equations.

While this lecture covers the most important parts of differential equations, it is not to be understood as doing complete diligence to the area of ordinary differential equations. You are advised to refer back to your differential equations course and follow the supplementary information[1] available on the course web page as needed during this course. We refer to the book by Walter [Wal98] for details and proofs about differential equations. For further background on differential equations, we refer you to the literature [Har64, Rei71, EEHJ96].

These lecture notes are based on material on cyber-physical systems, hybrid programs, and logic [Pla12, Pla10, Pla08, Pla07]. Cyber-physical systems play an important role in numerous domains [PCA07, LS10, LSC$^+$12] with applications in cars [DGV96], aircraft [TPS98], robots [PKV09], and power plants [FKV04], chemical processes [RKR10, KGDB10], medical models [GBF$^+$11, KAS$^+$11], and even an importance for understanding biological systems [Tiw11].

More information about CPS can be found in [Pla10, Chapter 1]. Differential equations and domains are described in [Pla10, Chapter 2.2,2.3] in more detail.

2. Differential Equations as Models of Continuous Physical Processes

Differential equations model processes in which the (state) variables of a system evolve continuously in time. A differential equation concisely describes how the system evolves over time. It describes how the variables change locally, so it, basically, indicates the direction in which the variables evolve at each point in space. Fig. 1 shows the respective directions in which the system evolves by a vector at each point and illustrates one solution which follows those vectors everywhere. Of course, the figure would be rather cluttered if we would literally try to indicate the vector at each and every point, of which there are uncountably infinitely many. But this is a shortcoming only of our illustration. Differential equations actually define such a vector for the direction of evolution at every point in space.

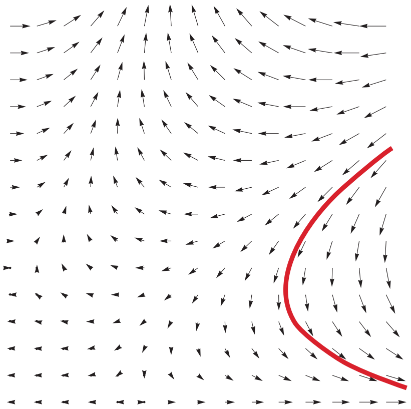


Figure 1: Vector field and one solution of a differential equation

[1] http://symbolaris.com/course/fcps13-resources.html

As an example, suppose we have a car whose position is denoted by x. How the value of variable x changes over time depends on how fast the car is driving. Let v denote the velocity of the car. Since v is the velocity of the car, its position x changes such that its derivative x' is v, which we write by the differential equation $x' = v$. This differential equation is supposed to mean that the time-derivative of the position x is the velocity v. So how x evolves depends on v. If the velocity is $v = 0$, then the position x does not change at all. If $v > 0$, then the position x keeps on increasing. How fast x increases depends on the value of v, bigger v give quicker changes in x.

Of course, the velocity v, itself, may also be subject to change over time. The car might accelerate, so let a denote its acceleration. Then the velocity v changes with time-derivative a, so $v' = a$. Overall, the car then follows the differential equation (system):[2]

$$x' = v, v' = a$$

That is, the position x of the car changes with time-derivative v, which, in turn, changes with time-derivative a.

What we mean by this differential equation, intuitively, is that the system has a vector field where all vectors point into direction a. What does this mean exactly?

3. The Meaning of Differential Equations

We relate some intuitive concept to how differential equations describe the direction of the evolution of a system as a vector field Fig. 1. But what exactly is a vector field? What does it mean to describe directions of evolutions at every point in space? Could these directions not possibly contradict each other so that the description becomes ambiguous? What is the exact meaning of a differential equation in the first place?

The only way to truly understand any system is to understand exactly what each of its pieces does. CPSs are demanding and misunderstandings about their effect often have far-reaching consequences. The physical impacts of CPSs do not leave much room for failure, so we immediately want to get into the mood of consistently studying the behavior and exact meaning of all relevant aspects of CPS.

An ordinary differential equation in explicit form is an equation $y'(t) = f(t, y)$ where $y'(t)$ is meant to be the derivative of y with respect to time t. A solution is a differentiable function Y which satisfies this equation when substituted in the differential equation, i.e., when substituting $Y(t)$ for y and the derivative $Y'(t)$ of Y at t for $y'(t)$.

Definition 1 (Ordinary differential equation)**.** Let $f : D \to \mathbb{R}^n$ be a function on a domain $D \subseteq \mathbb{R} \times \mathbb{R}^n$. The function $Y : I \to \mathbb{R}^n$ is a *solution* on the interval $I \subseteq \mathbb{R}$ of the *initial value problem*

$$\begin{bmatrix} y'(t) = & f(t, y) \\ y(t_0) = & y_0 \end{bmatrix} \tag{1}$$

[2] Note that the value of x changes over time, so it is really a function of time. Hence, the notation $x'(t) = v(t), v'(t) = a$ is sometimes used. It is customary, however, to suppress the argument t for time and just write $x' = v, v' = a$ instead.

with *ordinary differential equation* (*ODE*) $y' = f(t, y)$, if, for all $t \in I$

1. $(t, Y(t)) \in D$,
2. $Y'(t)$ exists and $Y'(t) = f(t, Y(t))$,
3. $Y(t_0) = y_0$.

If $f : D \to \mathbb{R}^n$ is continuous, then it is easy to see that $Y : I \to \mathbb{R}^n$ is continuously differentiable. Similarly if f is k-times continuously differentiable then Y is $k+1$-times continuously differentiable. The definition is accordingly for higher-order differential equations, i.e., differential equations involving higher-order derivatives $y^{(n)}(t)$ for $n > 1$.

Let us consider the intuition for this definition. A differential equation (system) can be thought of as a vector field such as the one in Fig. 1, where, at each point, the vector shows in which direction the solution evolves. At every point, the vector would correspond to the right-hand side of the differential equation. A solution of a differential equation adheres to this vector field at every point, i.e., the solution (e.g., the solid line in Fig. 1) locally follows the direction indicated by the vector of the right-hand side of the differential equation. There are many solutions of the differential equation corresponding to the vector field illustrated in Fig. 1. For the particular initial value problem, however, a solution also has to start at the position y_0 at time t_0 and then follow the differential equations or vector field from this point. In general, there could still be multiple solutions for the same initial value problem.

Example 2. Some differential equations are easy to solve. The initial value problem

$$\begin{bmatrix} x'(t) = & 5 \\ x(0) = & 2 \end{bmatrix}$$

has a solution $x(t) = 5t + 2$. This can be checked easily by inserting the solution into the differential equation and initial value equation:

$$\begin{bmatrix} (x(t))' = & (5t+2)' = 5 \\ x(0) = & 5 \cdot 0 + 2 = 2 \end{bmatrix}$$

Example 3. Consider the initial value problem

$$\begin{bmatrix} x'(t) = & -2x \\ x(1) = & 3 \end{bmatrix}$$

which has a solution $x(t) = 3e^{-2(t-1)}$. The test, again, is to insert the solution into the (differential) equations of the initial value problems and check:

$$\begin{bmatrix} (3e^{-2(t-1)})' = & -6e^{-2(t-1)} = -2x(t) \\ x(1) = & 3e^{-2(1-1)} = 3 \end{bmatrix}$$

Example 4. Consider the differential equation system $z' = v, v' = a$ and the initial value problem

$$\left[\begin{array}{rl} z'(t) = & v(t) \\ v'(t) = & a \\ z(0) = & z_0 \\ v(0) = & v_0 \end{array}\right]$$

Note that this initial value problem is a *symbolic initial value problem* with symbols z_0, v_0 as initial values (not specific numbers like 5 and 2.3). Moreover, the differential equation has a constant symbol a, and not a specific number like 0.6, in the differential equation. In vectorial notation, the initial value problem with this differential equation system corresponds to a vectorial system when we denote $y(t) := (z(t), v(t))$, i.e., with dimension $n = 2$ in Def. 1:

$$\left[\begin{array}{rl} y'(t) = \begin{pmatrix} z \\ v \end{pmatrix}'(t) = & \begin{pmatrix} v(t) \\ a \end{pmatrix} \\ y(0) = \begin{pmatrix} z \\ v \end{pmatrix}(0) = & \begin{pmatrix} z_0 \\ v_0 \end{pmatrix} \end{array}\right]$$

The solution of this initial value problem is

$$\begin{aligned} z(t) &= \frac{a}{2}t^2 + v_0 t + z_0 \\ v(t) &= at + v_0 \end{aligned}$$

We can show that this is the solution by inserting the solution into the (differential) equations of the initial value problems and checking:

$$\left[\begin{array}{rl} (\frac{a}{2}t^2 + v_0 t + z_0)' = & 2\frac{a}{2}t + v_0 = v(t) \\ (at + v_0)' = & a \\ z(0) = & \frac{a}{2}0^2 + v_0 0 + z_0 = z_0 \\ v(0) = & a0 + v_0 = v_0 \end{array}\right]$$

Example 5. Consider the differential equation system $x' = y, y' = -x$ and the initial value problem

$$\left[\begin{array}{rl} x'(t) = & y(t) \\ y'(t) = & -x(t) \\ x(0) = & 1 \\ y(0) = & 1 \end{array}\right]$$

The solution of this initial value problem is

$$\begin{aligned} x(t) &= \cos(t) + \sin(t) \\ y(t) &= \cos(t) - \sin(t) \end{aligned}$$

We can show that this is the solution by inserting the solution into the (differential) equations of the initial value problems and checking:

$$\left[\begin{array}{rl} (\cos(t)+\sin(t))' = & -\sin(t)+\cos(t) = y(t) \\ (\cos(t)-\sin(t))' = & -\sin(t)-\cos(t) = -x(t) \\ x(0) = & \cos(0)+\sin(0) = 1 \\ y(0) = & \cos(0)-\sin(0) = 1 \end{array}\right]$$

Note 1 (Descriptive power of differential equations)**.** *As a general phenomenon, observe that solutions of differential equations can be much more involved than the differential equations themselves, which is part of the representational and descriptive power of differential equations.*

4. Domains of Differential Equations

Now we understand exactly what a differential equation is and how it describes a continuous physical process. In CPS, however, physical processes interact with cyber elements such as computers. When and how do physics and cyber elements interact? The first thing we need to understand for that is how to describe when physics stops so that the cyber elements take control of what happens next. Obviously, physics does not literally stop evolving, but rather keeps on evolving all the time. Yet, the cyber parts only take effect every now and then. So, our intuition may imagine physics "pauses" for a period of duration 0 and lets the cyber take action to influence the inputs that physics is based on.

The cyber and the physics could interface in more than one way. Physics might evolve and the cyber elements interrupt to inspect measurements about the state of the system periodically to decide what to do next. Or the physics might trigger certain conditions that cause cyber elements to compute their responses. Another way to look at that is that a differential equation that a system follows forever without further intervention by anything would not describe a particularly well-controlled system. All those ways have in common that our model of physics needs to come up with information about when it stops evolving to give cyber a chance to perform its task.

This information is what is a called an *evolution domain* H of a differential equation, which describes a region that the system cannot leave. If the system were ever about to leave this region, it would stop evolving right away before it leaves the evolution domain.

Note 2. *A* differential equation $x' = f(x)$ with evolution domain H *is denoted by*

$$x' = f(x) \,\&\, H$$

This notation $x' = f(x) \,\&\, H$ signifies that the system follows the differential equation $x' = f(x)$ for any duration, but is never allowed to leave the region described by H. So the system evolution has to stop while the state is still in H.

If, e.g., t is a time variable with $t' = 1$, then $x' = v, v' = a, t' = 1 \,\&\, t \leq \varepsilon$ describes a system that follows the differential equation at most until time $t = \varepsilon$ and not any further. The evolution domain $H \overset{\text{def}}{\equiv} (v \geq 0)$, instead, restricts the system $x' = v, v' = a \,\&\, v \geq 0$ to nonnegative velocities. Should the velocity ever become negative while following the differential equation $x' = v, v' = a$, then the system stops before that happens.

In the scenario illustrated in Fig. 2, the system starts at time 0 inside the evolution domain that is depicted as a shaded green region in Fig. 2. Then the system follows the differential equation $x' = f(x)$ for any period of time, but has to stop before it leaves H. Here, it stops at time r.

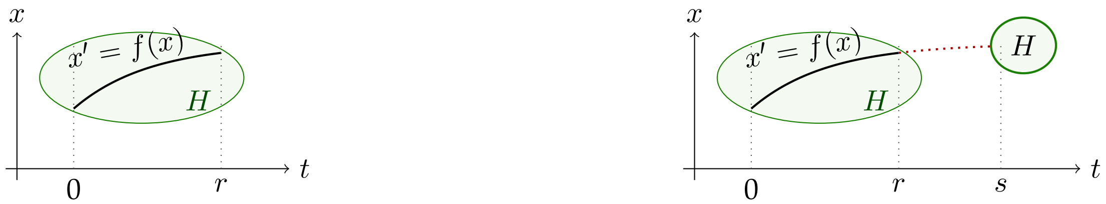


Figure 2: System $x' = f(x) \,\&\, H$ follows the differential equation $x' = f(x)$ but cannot leave the (shaded) evolution domain H.

In contrast, consider the scenario shown on the right of Fig. 2. The system is *not* allowed to evolve until time s, because—even if the system is back in the evolution domain H at that time—it has left the evolution domain H between time r and s (indicated by dotted lines), which is not allowed. Consequently, the continuous evolution on the right of Fig. 2 will also stop at time r at the latest.

How can we properly describe the evolution domain H? We will need some logic for that.

5. Continuous Programs: Syntax

After these preparations for understanding differential equations and domains, we start developing a programming language for cyber-physical systems. Ultimately, this programming language of *hybrid programs* will contain more features than just differential equations. But this most crucial feature is what we start with. This course develops this programming language and its understanding and its analysis in layers one after the other.

Continuous Programs. The first element of the syntax of hybrid programs is the following.

Note 3. *Version 1 of* hybrid programs *(HPs) are* continuous programs. *These are defined by the following grammar (α is a HP, x a variable, θ a term possibly containing x, and H a formula of first-order logic of real arithmetic):*

$$\alpha \ ::= \ x' = \theta \,\&\, H$$

This means that a hybrid program α consists of a single statement of the form $x' = \theta \,\&\, H$. In later lectures, we will add more statements to hybrid programs, but focus on differential equations for now. The formula H is called *evolution domain constraint* of the *continuous evolution* $x' = \theta \,\&\, H$. Further x is allowed to be a vector of variables and, then, θ is a vector of terms of the same dimension. This corresponds to the case of differential equation systems such as:

$$x' = v, v' = a \,\&\, (v \geq 0 \wedge v \leq 10)$$

Differential equations are allowed without an evolution domain constraint H as well, for example:

$$x' = y, y' = x + y^2$$

which corresponds to choosing *true* for H, since the formula *true* is true everywhere and imposes no condition on the state.

Terms. A rigorous definition of the syntax of hybrid programs also depends on defining what a term θ is and what a formula H of first-order logic of real arithmetic is. A *term* θ is a polynomial term defined by the grammar (where θ, ϑ are terms, x a variable, and c a rational number constant):

$$\theta, \vartheta ::= x \mid c \mid \theta + \vartheta \mid \theta \cdot \vartheta$$

This means that a term θ is either a variable x, or a rational number constant $c \in \mathbb{Q}$, or a sum of terms θ, ϑ, or a product of terms θ, ϑ. Subtraction $\theta - \vartheta$ is another useful case, but it turns out that it is already included, because subtraction can be defined by $\theta + (-1) \cdot \vartheta$.

First-order Formulas. The formulas of first-order logic of real arithmetic are defined as usual in first-order logic, yet using the language of real arithmetic. The formulas of *first-order logic of real arithmetic* are defined by the following grammar (where F, G are formulas of first-order logic of real arithmetic, θ, ϑ are (polynomial) terms, and x a variable):

$$F, G ::= \theta = \vartheta \mid \theta \geq \vartheta \mid \neg F \mid F \wedge G \mid F \vee G \mid F \rightarrow G \mid F \leftrightarrow G \mid \forall x\, F \mid \exists x\, F$$

The usual abbreviations are allowed, such as $\theta \leq \vartheta$ for $\vartheta \geq \theta$ and $\theta < \vartheta$ for $\neg(\theta \geq \vartheta)$.

6. Continuous Programs: Semantics

Note 4 (Syntax vs. Semantics)**.** *Syntax just defines a notation. Its meaning is defined by the semantics.*

Terms. The meaning of a continuous evolution $x' = \theta \,\&\, H$ depends on understanding the meaning of terms θ. A term θ is a syntactic expression. Its value depends on the interpretation of the variables contained in θ. What values those variables have changes depending on the state of the CPS. A *state* ν is a mapping from variables to real numbers. The set of states is denoted $\mathcal{S}$.

Definition 6 (Valuation of terms)**.** The *value of term* θ in state ν is denoted $[\![\theta]\!]_\nu$ and defined by induction on the structure of θ:

$$\begin{aligned}
[\![x]\!]_\nu &= \nu(x) && \text{if } x \text{ is a variable}\\
[\![c]\!]_\nu &= c && \text{if } c \text{ is a rational constant}\\
[\![\theta + \vartheta]\!]_\nu &= [\![\theta]\!]_\nu + [\![\vartheta]\!]_\nu\\
[\![\theta \cdot \vartheta]\!]_\nu &= [\![\theta]\!]_\nu \cdot [\![\vartheta]\!]_\nu
\end{aligned}$$

In particular, the value of a variable-free term like $4+5\cdot 2$ does not depend on the state ν. In this case, the value is 14. The value of a term with variables, like $4 + x \cdot 2$, depends on ν. Suppose $\nu(x) = 5$, then $[\![4 + x \cdot 2]\!]_\nu = 14$. If $\omega(x) = 2$, then $[\![4 + x \cdot 2]\!]_\omega = 8$.

First-order Formulas. Unlike for terms, the value of a logical formula is not a number but instead *true* or *false*. Whether a logical formula evaluates to *true* or *false* depends on the interpretation of its symbols. In first-order logic of real arithmetic, the meaning of all symbols except variables is fixed. The meaning of terms and of formulas of first-order logic of real arithmetic is as usual in first-order logic, except that $+$ really means addition, $\cdot$ means multiplication, $\geq$ means greater or equals, and that the quantifiers $\forall x$ and $\exists x$ quantify over the reals.

Let ν_x^d denote the state that agrees with state ν except for the interpretation of variable x, which is changed to the value $d \in \mathbb{R}$:

$$\nu_x^d(y) = \begin{cases} d & \text{if } y \text{ is the variable } x\\ \nu(y) & \text{otherwise}\end{cases}$$

We write $\nu \models F$ to indicate that F evaluates to *true* in state ν and define it as follows.

Definition 7 (First-order logic semantics)**.** The *satisfaction relation* $\nu \models F$ for a first-order formula F of real arithmetic in state ν is defined inductively:

- $\nu \models (\theta_1 = \theta_2)$ iff $[\![\theta_1]\!]_\nu = [\![\theta_2]\!]_\nu$.
- $\nu \models (\theta_1 \geq \theta_2)$ iff $[\![\theta_1]\!]_\nu \geq [\![\theta_2]\!]_\nu$.
- $\nu \models \neg F$ iff $\nu \not\models F$, i.e. if it is not the case that $\nu \models F$.
- $\nu \models F \land G$ iff $\nu \models F$ and $\nu \models G$.
- $\nu \models F \lor G$ iff $\nu \models F$ or $\nu \models G$.
- $\nu \models F \rightarrow G$ iff $\nu \not\models F$ or $\nu \models G$.
- $\nu \models F \leftrightarrow G$ iff ($\nu \models F$ and $\nu \models G$) or ($\nu \not\models F$ and $\nu \not\models G$).
- $\nu \models \forall x\, F$ iff $\nu_x^d \models F$ for all $d \in \mathbb{R}$.
- $\nu \models \exists x\, F$ iff $\nu_x^d \models F$ for some $d \in \mathbb{R}$.

If $\nu \models F$, then we say that F is true at ν or that ν is a model of F. A formula F is *valid*, written $\vDash F$, iff $\nu \models F$ for all states ν. A formula F is a *consequence* of a set of formulas Γ, written $\Gamma \vDash F$, iff, for each ν: $\nu \models G$ for all $G \in \Gamma$ implies that $\nu \models F$.

With this definition, we know how to evaluate whether a evolution domain H of a continuous evolution $x' = \theta \,\&\, H$ is true in a particular state ν or not. If $\nu \models H$, then H holds in that state. Otherwise (i.e. if $\nu \not\models H$), H does not hold in ν. Yet, in which states ν do we need to check the evolution domain?

Continuous Programs. There is more than one way to define the meaning of a program, including defining a denotational semantics, an operational semantics, a structural operational semantics, an axiomatic semantics. For our purposes, what is most relevant is how a hybrid program changes the state of the system. Consequently, the semantics of HPs is based on which final states are reachable from which initial state. It considers which (final) state ω is reachable by running a HP α from an (initial) state ν. Semantical models that expose more detail, e.g., about the internal states during the run of an HP are possible [Pla10, Chapter 4] but not needed for our usual purposes.

If a differential equation starts in a state ν, the system could reach many possible states when following this particular differential equation. Even though the solutions of initial value problems (differential equation with an initial state) are unique under mild conditions (Appendix B), they still do not lead to a single unique state. Which state one ends up at when following a differential equation depends not only on the initial state ν, but also on how long the system follows that differential equation. Consequently, the meaning of a continuous program will invariably have to allow for many possible reachable states. Recall that $\mathcal{S}$ denotes the set of states.

The meaning of an HP α is given by a reachability relation $\rho(\alpha) \subseteq \mathcal{S} \times \mathcal{S}$ on states. That is, $(\nu, \omega) \in \rho(\alpha)$ means that final state ω is reachable from initial state ν by running HP α. From any initial state ν, there might be many states ω that are reachable, so many different ω for which $(\nu, \omega) \in \rho(\alpha)$. Form other initial states ν, there might be no reachable states ω at all for which $(\nu,\omega) \in \rho(\alpha)$. So $\rho(\alpha)$ is a proper relation, not a function.

Note 5. *The reachability relation $\rho(x' = \theta \,\&\, H)$ of a continuous program holds for all pairs of states that can be connected by a solution of the differential equation that is entirely within H:*

$$\begin{aligned}\rho(x' = \theta \,\&\, H) = \{(\varphi(0), \varphi(r)) \;:\; &\varphi(t) \models H \textit{ for all } 0 \le t \le r\\ &\textit{for a solution } \varphi : [0,r] \to \mathcal{S} \textit{ of } x' = \theta \textit{ of any duration } r \in \mathbb{R}\}.\end{aligned}$$

The first line in the definition of $\rho(x' = \theta \,\&\, H)$ means that the solution satisfies H at all times. The second line means that φ solves the differential equation, which essentially means that $\varphi(t) \models x' = \theta$ for all $0 \le t \le r$, when interpreting $\varphi(t)(x') \overset{\text{def}}{=} \frac{\mathsf{d}\varphi(\zeta)(x)}{\mathsf{d}\zeta}(t)$. Let us elaborate what this means and explicitly consider differential equation systems:

Definition 8 (Semantics of continuous programs)**.** $(\nu,\omega) \in \rho(x_1' = \theta_1, \dots, x_n' = \theta_n \,\&\, H)$ iff there is a *flow* φ of some duration $r \ge 0$ along $x_1' = \theta_1, \dots, x_n' = \theta_n \,\&\, H$ from state ν to state ω, i.e. a function $\varphi : [0,r] \to \mathcal{S}$ such that:

- $\varphi(0) = \nu, \varphi(r) = \omega$;
- φ respects the differential equations: For each variable x_i, the valuation $[\![x_i]\!]_{\varphi(\zeta)} = \varphi(\zeta)(x_i)$ of x_i at state $\varphi(\zeta)$ is continuous in ζ on $[0,r]$ and has a derivative of value $[\![\theta_i]\!]_{\varphi(\zeta)}$ at each time $\zeta \in (0,r)$;
- the value of other variables $z \notin \{x_1,\dots,x_n\}$ remains constant, that is, we have $[\![z]\!]_{\varphi(\zeta)} = [\![z]\!]_\nu$ for all $\zeta \in [0,r]$;
- and φ respects the invariant: $\varphi(\zeta) \models H$ for each $\zeta \in [0,r]$.

Observe that this definition is explicit about the fact that variables without differential equations do not change during a continuous program. The semantics of HP is *explicit change*: nothing changes unless (an assignment or) a differential equation specifies how. Also observe the explicit passing from syntax to semantics by the use of the valuation function $[\![\cdot]\!]$ in Def. 8.

A. Existence Theorems

For your reference, this appendix contains a short primer on some important results about differential equations [Pla10, Appendix B].

There are several classical theorems that guarantee existence and/or uniqueness of solutions of differential equations (not necessarily closed-form solutions with elementary functions, though). The existence theorem is due to Peano [Pea90]. A proof can be found in [Wal98, Theorem 10.IX].

Theorem 9 (Existence theorem of Peano)**.** *Let $f : D \to \mathbb{R}^n$ be a continuous function on an open, connected domain $D \subseteq \mathbb{R} \times \mathbb{R}^n$. Then, the initial value problem* (1) *with $(t_0, y_0) \in D$ has a solution. Further, every solution of* (1) *can be continued arbitrarily close to the boundary of D.*

Peano's theorem only proves that a solution exists, not for what duration it exists. Still, it shows that every solution can be *continued arbitrarily close to the boundary* of the domain D. That is, the closure of the graph of the solution, when restricted to $[0, 0] \times \mathbb{R}^n$, is not a compact subset of D. In particular, there is a global solution on the interval $[0, \infty)$ if $D = \mathbb{R}^{n+1}$ then.

Peano's theorem shows the existence of solutions of continuous differential equations on open, connected domains, but there can still be multiple solutions.

Example 10. The initial value problem with the following continuous differential equation

$$\begin{bmatrix} y' & = \sqrt[3]{|y|} \\ y(0) & = 0 \end{bmatrix}$$

has multiple solutions:

$$\begin{aligned} y(t) &= 0 \\ y(t) &= \left(\frac{2}{3}t\right)^{\frac{3}{2}} \\ y(t) &= \begin{cases} 0 & \text{for } t \leq s \\ \left(\frac{2}{3}(t-s)\right)^{\frac{3}{2}} & \text{for } t > s \end{cases} \end{aligned}$$

where $s \geq 0$ is any nonnegative real number.

B. Existence and Uniqueness Theorems

As usual, $C^k(D, \mathbb{R}^n)$ denotes the space of k times continuously differentiable functions from domain D to $\mathbb{R}^n$.

If we know that the differential equation (its right-hand side) is continuously differentiable on an open, connected domain, then the Picard-Lindelöf theorem gives a stronger result than Peano's theorem. It shows that there is a unique solution (except, of course, that the restriction of any solution to a sub-interval is again a solution). For this, recall that a function $f : D \to \mathbb{R}^n$ with $D \subseteq \mathbb{R} \times \mathbb{R}^n$ is called *Lipschitz continuous* with respect to y iff there is an $L \in \mathbb{R}$ such that for all $(t, y), (t, \bar{y}) \in D$,

$$\|f(t, y) - f(t, \bar{y})\| \leq L\|y - \bar{y}\|.$$

If, for instance, $\frac{\partial f(t,y)}{\partial y}$ exists and is bounded on D, then f is Lipschitz continuous with $L = \max_{(t,y)\in D} \|\frac{\partial f(t,y)}{\partial y}\|$ by mean value theorem. Similarly, f is *locally Lipschitz continuous* iff for each $(t, y) \in D$, there is a neighbourhood in which f is Lipschitz continuous. In particular, if f is continuously differentiable, i.e., $f \in C^1(D, \mathbb{R}^n)$, then f is locally Lipschitz continuous.

Most importantly, Picard-Lindelöf's theorem [Lin94], which is also known as the Cauchy-Lipschitz theorem, guarantees existence and uniqueness of solutions. As restrictions of solutions are always solutions, we understand uniqueness up to restrictions. A proof can be found in [Wal98, Theorem 10.VI]

Theorem 11 (Uniqueness theorem of Picard-Lindelöf)**.** *In addition to the assumptions of Theorem 9, let f be locally Lipschitz continuous with respect to y (for instance, $f \in C^1(D, \mathbb{R}^n)$ is sufficient). Then, there is a unique solution of the initial value problem* (1)*.*

Picard-Lindelöf's theorem does not show the duration of the solution, but shows only that the solution is unique. Under the assumptions of Picard-Lindelöf's theorem, every solution can be extended to a solution of maximal duration arbitrarily close to the boundary of D by Peano's theorem, however. The solution is unique, except that all restrictions of the solution to a sub-interval are also solutions.

Example 12. The initial value problem

$$\begin{bmatrix} y' & = y^2 \\ y(0) & = 1 \end{bmatrix}$$

has the unique maximal solution $y(t) = \frac{1}{1-t}$ on the domain $t < 1$. This solution cannot be extended to include the singularity at $t = 1$.

The following global uniqueness theorem shows a stronger property when the domain is $[0, a] \times \mathbb{R}^n$. It is a corollary to Theorems 9 and 11, but used prominently in the proof of Theorem 11, and is of independent interest. A direct proof of the following global version of the Picard-Lindelöf theorem can be found in [Wal98, Proposition 10.VII].

Corollary 13 (Global uniqueness theorem of Picard-Lindelöf)**.** *Let $f : [0, a] \times \mathbb{R}^n \to \mathbb{R}^n$ be a continuous function that is Lipschitz continuous with respect to y. Then, there is a unique solution of the initial value problem* (1) *on* $[0, a]$*.*

Exercises

Exercise 1. Review the basic theory of ordinary differential equations and examples.

Exercise 2. Review the syntax and semantics of first-order logic.

References

[DGV96] Akash Deshpande, Aleks Göllü, and Pravin Varaiya. SHIFT: A formalism and a programming language for dynamic networks of hybrid automata. In Panos J. Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems*, volume 1273 of *LNCS*, pages 113–133. Springer, 1996.

[EEHJ96] Kenneth Eriksson, Donald Estep, Peter Hansbo, and Claes Johnson. *Computational Differential Equations*. Cambridge University Press, 1996.

[FKV04] G. K. Fourlas, K. J. Kyriakopoulos, and C. D. Vournas. Hybrid systems modeling for power systems. *Circuits and Systems Magazine, IEEE*, 4(3):16 – 23, quarter 2004.

[GBF+11] Radu Grosu, Grégory Batt, Flavio H. Fenton, James Glimm, Colas Le Guernic, Scott A. Smolka, and Ezio Bartocci. From cardiac cells to genetic regulatory networks. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *CAV*, volume 6806 of *LNCS*, pages 396–411. Springer, 2011. doi:10.1007/978-3-642-22110-1_31.

[Har64] Philip Hartman. *Ordinary Differential Equations*. John Wiley, 1964.

[KAS+11] BaekGyu Kim, Anaheed Ayoub, Oleg Sokolsky, Insup Lee, Paul L. Jones, Yi Zhang, and Raoul Praful Jetley. Safety-assured development of the gpca infusion pump software. In Samarjit Chakraborty, Ahmed Jerraya, Sanjoy K. Baruah, and Sebastian Fischmeister, editors, *EMSOFT*, pages 155–164. ACM, 2011. doi:10.1145/2038642.2038667.

[KGDB10] Branko Kerkez, Steven D. Glaser, John A. Dracup, and Roger C. Bales. A hybrid system model of seasonal snowpack water balance. In Karl Henrik Johansson and Wang Yi, editors, *HSCC*, pages 171–180. ACM, 2010. doi:10.1145/1755952.1755977.

[Lin94] M. Ernst Lindelöf. Sur l'application de la méthode des approximations successives aux équations différentielles ordinaires du premier ordre. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, 114:454–457, 1894.

[LS10] Insup Lee and Oleg Sokolsky. Medical cyber physical systems. In Sachin S. Sapatnekar, editor, *DAC*, pages 743–748. ACM, 2010.

[LSC+12] Insup Lee, Oleg Sokolsky, Sanjian Chen, John Hatcliff, Eunkyoung Jee, BaekGyu Kim, Andrew L. King, Margaret Mullen-Fortino, Soojin Park, Alex Roederer, and Krishna K. Venkatasubramanian. Challenges and research directions in medical cyber-physical systems. *Proc. IEEE*, 100(1):75–90, 2012. doi:10.1109/JPROC.2011.2165270.

[PCA07] Leadership under challenge: Information technology R&D in a competitive world. an assessment of the federal networking and information technology R&D program. President's Council of Advisors on Science and Technology, Aug 2007. http://www.ostp.gov/pdf/nitrd_review.pdf.

[Pea90] Giuseppe Peano. Demonstration de l'intégrabilité des équations différentielles ordinaires. *Mathematische Annalen*, 37(2):182–228, 1890.

[PKV09] Erion Plaku, Lydia E. Kavraki, and Moshe Y. Vardi. Hybrid systems: from verification to falsification by combining motion planning and discrete search. *Form. Methods Syst. Des.*, 34(2):157–182, 2009.

[Pla07] André Platzer. Differential dynamic logic for verifying parametric hybrid systems. In Nicola Olivetti, editor, *TABLEAUX*, volume 4548 of *LNCS*, pages 216–232. Springer, 2007. doi:10.1007/978-3-540-73099-6_17.

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

[Pla10] 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. Logics of dynamical systems. In *LICS*, pages 13–24. IEEE, 2012. doi:10.1109/LICS.2012.13.

[Rei71] William T. Reid. *Ordinary Differential Equations*. John Wiley, 1971.

[RKR10] Derek Riley, Xenofon Koutsoukos, and Kasandra Riley. Reachability analysis of stochastic hybrid systems: A biodiesel production system. *European Journal on Control*, 16(6):609–623, 2010.

[Tiw11] Ashish Tiwari. Logic in software, dynamical and biological systems. In *LICS*, pages 9–10. IEEE Computer Society, 2011. doi:10.1109/LICS.2011.20.

[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.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Choice & Control

André Platzer

Carnegie Mellon University
Lecture 3

1 Introduction

In the previous lecture, we have seen the beginning of cyber-physical systems, yet emphasized their continuous part in the form of differential equations $x' = \theta$. The sole interface between continuous physical capabilities and cyber capabilities was by way of their evolution domain. The evolution domain H in a continuous program $x' = \theta \,\&\, H$ imposes restrictions on how far or how long the system can evolve along that differential equation. Suppose a continuous evolution has succeeded and the system stops following its differential equation, e.g., because the state would otherwise leave the evolution domain. What happens now? How does the cyber take control? How do we describe what the cyber elements compute and how they interact with physics?

This lecture extends the model of continuous programs for continuous dynamics to the model of hybrid programs for hybrid dynamics.

This lecture is based on material on cyber-physical systems and hybrid programs [Pla12b, Pla10, Pla08, Pla07].

Continuous programs $x' = \theta \,\&\, H$ are very powerful for modeling continuous processes. They cannot—on their own—model discrete changes of variables, however.[1] During the evolution along a differential equation, all variables change continuously in time, because the solution of a differential equation is (sufficiently) smooth. Discontinuous change of variables, instead, needs a way for a discrete change of state. What could be a model for describing discrete changes in a system?

[1] There is a much deeper sense [Pla12a] in which continuous dynamics and discrete dynamics are quite surprisingly close together. That understanding requires a lot more logic than we have at our disposal at this stage of the course. It also leads to a full understanding of what constitutes the hybridness of hybrid systems. Yet, its understanding does rest on the foundations of hybrid systems, which we need to understand first.

There are many models for describing discrete change. You will have seen a number of them already. CPSs combine cyber and physics. In CPS, we do not program computers, but program CPSs instead. As part of that, we program the computers that control the physics. And programming computers amounts to using a programming language. Of course, for programming an actual CPS, our programming language will ultimately have to involve physics. But we have already seen continuous programs in the previous lecture for that very purpose. What's missing is a way to program the discrete and cyber aspects.

Does it matter which discrete programming language we choose as a basis? It could be argued that the discrete programming language does not matter as much as the hybrid aspects do. After all, there are many programming languages that are Turing-equivalent, i.e. that compute the same functions. Yet even among them there are numerous differences for various purposes in the discrete case, which are studied in the area of Programming Languages.

For the particular purposes of CPS, however, we will find further desiderata, i.e. things that we expect from a programming language to be adequate for CPS. We will develop what we need as we go.

More information about choice and control can be found in [Pla10, Chapter 2.2,2.3].

2 Discrete Programs and Sequential Composition

Discrete change happens in computer programs when they assign a new value to a variable. The statement $x := \theta$ assigns the value of term θ to variable x. It leads to a discrete, discontinuous change, because the value of x does not vary smoothly but radically when assigning θ to x.

This gives us a discrete model of change, $x := \theta$, in addition to the continuous model of change, $x' = \theta \,\&\, H$ from the previous lecture. Now, we can model systems that are either discrete or continuous. Yet, how can we model proper CPS that combine cyber and physics and that, thus, simultaneously combine discrete with continuous dynamics?

One way how cyber and physics can interact is if a computer provides input to physics. Physics may mention a variable like a for acceleration and a computer program sets its value depending on whether the computer program wants to accelerate or brake. That is, cyber could set the values of actuators that affect physics.

In this case, cyber and physics interact in such a way that cyber first does something and physics then follows. That corresponds to a sequential composition $(\alpha; \beta)$ in which first the HP α on the left of ; runs and, when it's done, the HP β on the right of ; runs. For example, the following HP

$$a := a + 1;\ x' = v, v' = a \tag{1}$$

will first let cyber perform a discrete change of setting a to $a + 1$ and then let physics follow the differential equation $x'' = a$. The overall effect is that cyber increases a and physics then lets x evolve with acceleration a (and increases velocity v with derivative

a). Thus, HP (1) models a situation where the desired acceleration is commanded once to increase and the robot then moves with that acceleration. Note that the sequential composition operator (;) has basically the same effect that it has in programming languages like Java or C0. It separates statements that are to be executed sequentially one after the other. If you look closely, however, you will find a subtle difference in that Java and C0 expect more ; than hybrid programs.

The HP in (1) executes control (it sets the acceleration for physics), but it has very little choice. Actually no choice at all. So only if the CPS is very lucky will an increase in acceleration be the right action to remain safe.

3 Decisions in Hybrid Programs

In general, a CPS may have to check conditions on the state to see which action to take. One way of doing that is the use of an if-then-else, as in classical discrete programs.

$$\begin{aligned}&\texttt{if}(v < 4)\, a := a + 1\, \texttt{else}\, a := -b;\\&x' = v, v' = a\end{aligned} \tag{2}$$

This HP will check the condition $v < 4$ to see if the current velocity is still less then 4. If it is, then a will be increased by 1. Otherwise, a will be set to $-b$ for some braking deceleration constant $b > 0$. Afterwards, i.e. when the if-then-else statement has run to completion, the HP will again evolve x with acceleration a along a differential equation.

The HP (2) takes only the current velocity into account to reach a decision on whether to accelerate or brake. That is usually not enough information to guarantee safety, because a robot doing that would be so fixated on achieving its desired speed that it would happily speed into any walls or other obstacles along the way. Consequently, programs that control robots also take other state information into account, for example the distance $x - o$ to an obstacle o from the robot's position x, not just its velocity v:

$$\begin{aligned}&\texttt{if}(x - o > 5)\, a := a + 1\, \texttt{else}\, a := -b;\\&x' = v, v' = a\end{aligned} \tag{3}$$

They could also take both distance and velocity into account for the decision:

$$\begin{aligned}&\texttt{if}(x - o > 5 \land v < 4)\, a := a + 1\, \texttt{else}\, a := -b;\\&x' = v, v' = a\end{aligned} \tag{4}$$

Note 1 (Iterative design). *As part of the labs of this course, you will develop increasingly more intelligent controllers for robots that face increasingly challenging environments. Designing controllers for robots or other CPS is a serious challenge. You will want to start with simple controllers for simple circumstances and only move on to more advanced challenges when you have fully understood and mastered the previous controllers, what behavior they guarantee and what functionality they are still missing.*

4 Choices in Hybrid Programs

What we learn from the above discussion is a common feature of CPS models: they often include only some but not all detail about the system. And for good reasons, because full detail about everything can be overwhelming. A (somewhat) more complete model of (4) might look as follows, with some further formula S as an extra condition for checking whether to actually accelerate:

$$\begin{aligned}&\texttt{if}(x-o>5\land v<4\land S)\,a:=a+1\,\texttt{else}\,a:=-b;\\&x'=v,v'=a\end{aligned}\tag{5}$$

The extra condition S may be very complicated and often depends on many factors. It could check to smooth the ride, optimize battery efficiency, or pursue secondary goals. Consequently, (4) is not actually a faithful model for (5), because (4) insists that the acceleration would always be increased just because $x-o>5\land v<4$, unlike (5), which checks the additional condition S. Likewise, (3) certainly is no faithful model of (5). But it looks simpler.

How can we describe a model that is simpler than (5) by ignoring the details of S yet that is still faithful? What we want this model to do is characterize that the controller may either increase acceleration by 1 or brake and that acceleration certainly only happens when $x-o>5$. But the model should make less commitment than (3) about under which circumstances braking is chosen. So we want a model that allows braking under more circumstances than (3) without having to model precisely under which circumstances that is. In order to simplify the system faithfully, we want a model that allows more behavior than (3).

Note 2 (Abstraction)**.** *Successful CPS models often include relevant aspects of the system only and simplify irrelevant detail. The benefit of doing so is that the model and its analysis becomes simpler, enabling us to focus on the critical parts without being bogged down in tangentials. This is the power of abstraction, arguably the primary secret weapon of computer science. It does take considerable skill, however, to find the best level of abstraction for a system. A skill that you will continue to sharpen through your entire career as a computer scientist.*

Let us take the development of this model this step by step. The first feature that the controller of this model has is a choice. The controller can choose to increase acceleration or to brake, instead. Such a choice between two actions is denoted by the operator $\cup$:

$$\begin{aligned}&(a:=a+1\cup a:=-b);\\&x'=v,v'=a\end{aligned}\tag{6}$$

When running this hybrid program, the first thing that happens is a choice between whether to run $a:=a+1$ or whether to run $a:=-b$. That is, the choice is whether to increase a by 1 or whether to reset a to $-b$ for braking. After this choice (i.e. after the ; operator), the system follows the usual differential equation $x''=a$.

Note 3 (Nondeterministic $\cup$)**.** *The choice* ($\cup$) *is* nondeterministic. *That is, every time a choice $\alpha \cup \beta$ runs, exactly one of the two choices, α or β, is chosen to run and the choice is* nondeterministic, *i.e. there is no prior way of telling which of the two choices is going to be chosen.*

The HP (6) is a *faithful abstraction* of (5), because every way how (5) can run can be mimicked by (6) so that the outcome of (6) corresponds to that of (5). Whenever (5) runs $a := a + 1$, which happens exactly if $x - o > 5 \land v < 4 \land S$ is *true*, (6) only needs to choose to run the left choice $a := a + 1$. Whenever (5) runs $a := -b$, which happens exactly if $x - o > 5 \land v < 4 \land S$ is *false*, (6) needs to choose to run the right choice $a := -b$. So all runs of (5) are possible runs of (6). Furthermore, (6) is much simpler than (5), because it contains less detail. It does not mention $v < 4$ nor the complicated extra condition S. Yet, (6) is a little too permissive, because it suddenly allows the controller to choose $a := a + 1$ even at close distance to the obstacle, i.e. even if $x - o > 5$ is *false*. That way, even if (5) was a safe controller, (6) is still an unsafe one.

5 Tests in Hybrid Programs

In order to make a faithful and not too permissive model of (5), we need to restrict the permitted choices in (6) so that the acceleration choice $a := a + 1$ can only be chosen at sufficient distance $x - o > 5$. The way to do that is to use tests on the current state of the system.

A *test*[2] $?H$ is a statement that checks the value of a first-order formula H of real arithmetic in the current state. If H holds in the current state, then the test passes, nothing happens, yet the HP continues to run normally. If, instead, H does not hold in the current state, then the test fails, and the system execution is aborted and discarded. That is, when ν is the current state, then $?H$ runs successfully without changing the state when $\nu \models H$. Otherwise, i.e. if $\nu \not\models H$, the run of $?H$ is aborted and not considered any further.

The test statement can be used to change (6) around so that it allows acceleration only at large distances while braking is still allowed always:

$$\begin{aligned}&\big((?x - o > 5; a := a + 1) \cup a := -b\big);\\ &x' = v, v' = a\end{aligned} \tag{7}$$

The first statement of (7) is a choice ($\cup$) between $(?x - o > 5; a := a + 1)$ and $a := -b$. All choices in hybrid programs are nondeterministic so any outcome is always possible. In (7), this means that the left choice can always be chosen, just as well as the right one. The first statement that happens in the left choice, however, is the test $?x - o > 5$, which the system run has to pass to continue successfully. In particular, if $x - o > 5$ is indeed *true* in the current state, then the system passes that test $?x - o > 5$ and the

[2] In a more general context, tests are also known as *challenges* [Pla13].

execution proceeds to after the sequential composition (;) to run $a := a + 1$. If $x - o > 5$ is *false* in the current state, however, the system fails the test $?x - o > 5$ and that run is aborted and discarded. The right option to brake is always available, because it does not involve any tests to pass.

Note 4 (Discarding failed runs)**.** *System runs that fail tests are discarded and not considered any further. It is as if those failed system execution attempts had never happened. Yet, other execution paths may be successful. You can imagine finding them by backtracking the choices in the system run and taking alternative choices instead.*

There are always two choices when running (7). Yet, which ones run successfully depends on the current state. If the current state is at a far distance from the obstacle ($x - o > 5$), then both options of accelerating and braking will be possible. Otherwise, only the braking choice runs without being discarded because of failing a test.

Comparing (7) with (5), we see that (7) is a faithful abstraction of the more complicated (5), because all runs of (5) can be mimicked by (7). Yet, unlike (6), the improved HP (7) retains the critical information that acceleration is only allowed by (5) at sufficient distance $x - o > 5$. Unlike (5), (7) does not restrict the cases where acceleration can be chosen to those that also satisfy $v < 4 \land S$. Hence, (7) is more permissive than (5). But (7) is also simpler and only contains crucial information about the controller. Hence, (7) is a more abstract faithful model of (5) that retains the relevant detail. Studying the abstract (7) instead of the more concrete (5) has the advantage that only relevant details need to be understood while irrelevant aspects can be ignored. It also has the additional advantage that a safety analysis of the more abstract (7), which allows lots of behavior, will imply safety of the special concrete case (5) but also implies safety of other implementations of (7). For example, replacing S by a different condition in (5) still gives a special case of (7). So if all behavior of (7) is safe, all behavior of that different replacement will also already be safe. With a single verification result about a more general, more abstract system, we can obtain verification for a whole class of systems. This important phenomenon will be investigated in more detail in later parts of the course.

Of course, which details are relevant and which ones can be simplified depends on the analysis question at hand, a question that we will be better equipped to answer in a later lecture. For now, suffice it to say that (7) has the relevant level of abstraction for our purposes.

6 Repetitions in Hybrid Programs

The hybrid programs above were interesting, but only allowed the controller to choose what action to take at most once. All controllers so far inspected the state in a test or in an if-then-else condition and then chose what to do once, just to let physics take control by following a differential equation. That makes for rather short-lived controllers. They have a job only once in their lives. And most decisions they reach may end up being bad ones. Say, one of those controllers, e.g. (7), inspects the state and finds it still okay to accelerate. If it chooses $a := a + 1$ and then lets physics move in the differential equation $x'' = a$, there will probably come a time at which acceleration is no longer such a great idea. But the controller of (7) has no way to change its mind, because he has no more choices and so no control anymore.

If the controller of (7) is supposed to be able to make a second control choice later after physics has followed the differential equation for a while, then (7) can be sequentially composed with itself:

$$\begin{aligned}&\big((?x - o > 5; a := a + 1) \cup a := -b\big);\\ &x' = v, v' = a;\\ &\big((?x - o > 5; a := a + 1) \cup a := -b\big);\\ &x' = v, v' = a\end{aligned} \tag{8}$$

In (8), the cyber controller can first choose to accelerate or brake (depending on whether $x - o > 5$), then physics evolves along differential equation $x'' = a$ for some while, then the controller can again choose whether to accelerate or brake (depending on whether $x - o > 5$ holds in the state reached then), and finally physics again evolves along $x'' = a$.

For a controller that is supposed to be allowed to have a third control choice, replication would again help:

$$\begin{aligned}&\big((?x - o > 5; a := a + 1) \cup a := -b\big);\\ &x' = v, v' = a;\\ &\big((?x - o > 5; a := a + 1) \cup a := -b\big);\\ &x' = v, v' = a;\\ &\big((?x - o > 5; a := a + 1) \cup a := -b\big);\\ &x' = v, v' = a\end{aligned} \tag{9}$$

But this is neither a particularly concise nor a particularly useful modeling style. What if a controller could need 10 control decisions or 100? Or what if there is no way of telling ahead of time how many control decisions the cyber part will have to take to reach its goal? Think of how many control decisions you might need when driving in a car from the East Coast to the West Coast. Do you know that ahead of time? Even if you do, do you want to model a system by explicitly replicating its controller that often?

Note 5 (Repetition)**.** *As a more concise and more general way of describing repeated control choices, hybrid programs allow for the repetition operator* **, which works like the* * *in regular expressions, except that it applies to a hybrid program* α *as in* α^**. It repeats* α *any number* $n \in \mathbb{N}$ *of times, by a nondeterministic choice.*

Thus, a way of summarizing (7), (8), (9) and the infinitely many more n-fold replications of (7) for any $n \in \mathbb{N}$, is by using a repetition operator:

$$\begin{aligned}\Big(&\big((?x-o>5; a:=a+1) \cup a:=-b\big);\\ &x'=v, v'=a\Big)^*\end{aligned} \tag{10}$$

This HP can repeat (7) any number of times.

But how often does a nondeterministic repetition like (10) repeat then? That choice is again nondeterministic.

Note 6 (Nondeterministic *)**.** *Repetition (*) is* nondeterministic*. That is, α^* can repeat α any number ($n \in \mathbb{N}$) of times and the choice how often to run α is* nondeterministic*, i.e. there is no prior way of telling how often α will be repeated.*

Yet, every time the loop in (10) is run, how long does its continuous evolution take? Or, actually, even in the loop-free (8), how long does the first $x''=a$ take before the controller has its second choice? How long did the continuous evolution take in (7) in the first place?

There is a choice in following a differential equation. Even if the solution of the differential equation is unique (cf. lecture 2), it is still a matter of choice how long to follow that solution. The choice is, as always in hybrid programs, nondeterministic.

Note 7 (Nondeterministic $x'=\theta$)**.** *The duration of evolution of a differential equation ($x'=\theta \,\&\, H$) is* nondeterministic *(except that the evolution can never be so long that the state leaves H). That is, $x'=\theta \,\&\, H$ can follow the solution of $x'=\theta$ any amount of time ($0 \leq r \in \mathbb{R}$) of times and the choice how long to follow $x'=\theta$ is* nondeterministic*, i.e. there is no prior way of telling how often $x'=\theta$ will be repeated (except that it can never leave H).*

7 Syntax of Hybrid Programs

With the motivation above, we formally define hybrid programs [Pla12a, Pla10].

Definition 1 (Hybrid program)**.** HPs are defined by the following grammar (α, β are HPs, x a variable, θ a term possibly containing x, and H a formula of first-order logic of real arithmetic):

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

The first three cases are called atomic HPs, the last three compound. The *test* action $?H$ is used to define conditions. Its effect is that of a *no-op* if the formula H is true in the current state; otherwise, like *abort*, it allows no transitions. That is, if the test succeeds because formula H holds in the current state, then the state does not change, and the system execution continues normally. If the test fails because formula H does not hold in the current state, then the system execution cannot continue, is cut off, and not considered any further.

Nondeterministic choice $\alpha \cup \beta$, sequential composition $\alpha;\beta$, and nondeterministic repetition α^* of programs are as in regular expressions but generalized to a semantics in hybrid systems. *Nondeterministic choice* $\alpha \cup \beta$ expresses behavioral alternatives between the runs of α and β. That is, the HP $\alpha \cup \beta$ can choose nondeterministically to follow the runs of HP α, or, instead, to follow the runs of HP β. The *sequential composition* $\alpha;\beta$ models that the HP β starts running after HP α has finished (β never starts if α does not terminate). In $\alpha;\beta$, the runs of α take effect first, until α terminates (if it does), and then β continues. Observe that, like repetitions, continuous evolutions within α can take more or less time, which causes uncountable nondeterminism. This nondeterminism occurs in hybrid systems, because they can operate in so many different ways, which is as such reflected in HPs. *Nondeterministic repetition* α^* is used to express that the HP α repeats any number of times, including zero times. When following α^*, the runs of HP α can be repeated over and over again, any nondeterministic number of times ($\geq$0).

Unary operators (including *) bind stronger than binary operators and let ; bind stronger than $\cup$, so $\alpha;\beta \cup \gamma \equiv (\alpha;\beta) \cup \gamma$ and $\alpha \cup \beta;\gamma \equiv \alpha \cup (\beta;\gamma)$. Further, $\alpha;\beta^* \equiv \alpha;(\beta^*)$.

8 Semantics of Hybrid Programs

HPs have a compositional semantics [Pla12b, Pla10, Pla08]. Their semantics is defined by a reachability relation. A *state* ν is a mapping from variables to $\mathbb{R}$. The set of states is denoted $\mathcal{S}$. The value of term θ in ν is denoted by $[\![\theta]\!]_\nu$. Recall that $\nu \models H$ denotes that first-order formula H is true in state ν (lecture 2).

Definition 2 (Transition semantics of HPs)**.** Each HP α is interpreted semantically as a binary reachability relation $\rho(\alpha) \subseteq \mathcal{S} \times \mathcal{S}$ over states, defined inductively by

1. $\rho(x := \theta) = \{(\nu, \omega) \ : \ \omega = \nu$ except that $[\![x]\!]_\omega = [\![\theta]\!]_\nu\}$
2. $\rho(?H) = \{(\nu, \nu) \ : \ \nu \models H\}$
3. $\rho(x' = \theta \,\&\, H) = \{(\varphi(0), \varphi(r)) \ : \ \varphi(t) \models x' = \theta$ and $\varphi(t) \models H$ for all $0 \leq t \leq r$ for a solution $\varphi : [0, r] \to \mathcal{S}$ of any duration $r\}$; i.e., with $\varphi(t)(x') \overset{\text{def}}{=} \frac{\mathrm{d}\varphi(\zeta)(x)}{\mathrm{d}\zeta}(t)$, φ solves the differential equation and satisfies H at all times, see lecture 2.
4. $\rho(\alpha \cup \beta) = \rho(\alpha) \cup \rho(\beta)$
5. $\rho(\alpha;\beta) = \rho(\beta) \circ \rho(\alpha) = \{(\nu, \omega) : (\nu, \mu) \in \rho(\alpha), (\mu, \omega) \in \rho(\beta)\}$
6. $\rho(\alpha^*) = \bigcup_{n \in \mathbb{N}} \rho(\alpha^n)$ with $\alpha^{n+1} \equiv \alpha^n;\alpha$ and $\alpha^0 \equiv ?\mathit{true}$.

For graphical illustrations of the transition semantics of hybrid programs and example dynamics, see Fig. 1. On the left of Fig. 1, we illustrate the generic shape of the transition structure $\rho(\alpha)$ for transitions along various cases of hybrid programs α from state ν to state ω. On the right of Fig. 1, we show examples of how the value of a variable x may evolve over time t when following the dynamics of the respective hybrid program α.

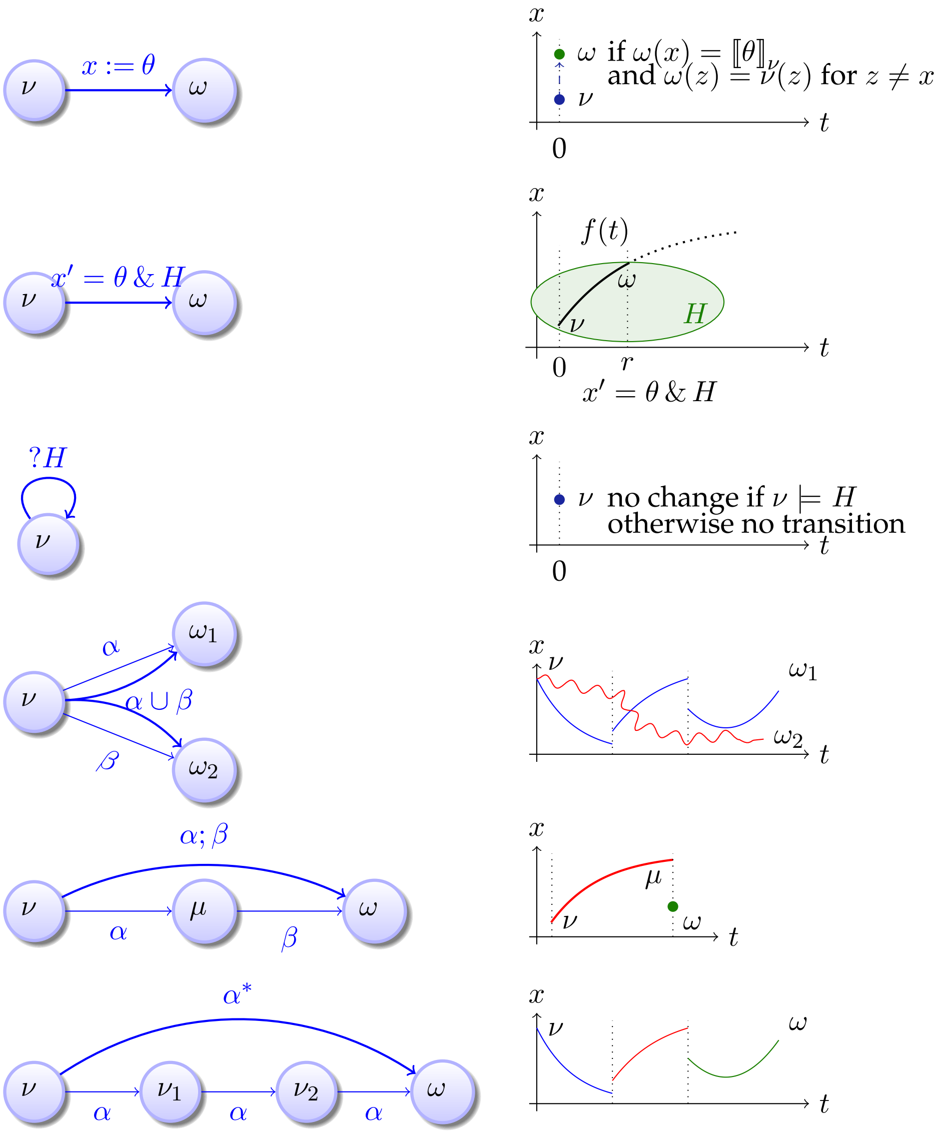


Figure 1: Transition semantics (left) and example dynamics (right) of hybrid programs

Exercises

Exercise 1. Consider your favorite programming language and discuss in what ways it introduces discrete change and discrete dynamics. Can it model all behavior that hybrid programs can describe? Can your programming language model all behavior that hybrid programs without differential equations can describe? How about the other way around?

Exercise 2. Consider the grammar of hybrid programs. The ; in hybrid programs is similar to the ; in Java and C0. If you look closely you will find a subtle difference. Identify the difference and explain why there is such a difference.

Exercise 3. Sect. 3 considered if-then-else statements for hybrid programs. But they no longer showed up in the grammar of hybrid programs. Is this a mistake?

References

[DBL12] *Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25–28, 2012*. IEEE, 2012.

[Pla07] André Platzer. Differential dynamic logic for verifying parametric hybrid systems. In Nicola Olivetti, editor, *TABLEAUX*, volume 4548 of *LNCS*, pages 216–232. Springer, 2007. doi:10.1007/978-3-540-73099-6_17.

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

[Pla10] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14509-4.

[Pla12a] André Platzer. The complete proof theory of hybrid systems. In *LICS* [DBL12], pages 541–550. doi:10.1109/LICS.2012.64.

[Pla12b] André Platzer. Logics of dynamical systems. In *LICS* [DBL12], pages 13–24. doi:10.1109/LICS.2012.13.

[Pla13] André Platzer. A complete axiomatization of differential game logic for hybrid games. Technical Report CMU-CS-13-100R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January, Revised and extended in July 2013.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Safety & Contracts

André Platzer

Carnegie Mellon University

Lecture 4

1 Introduction

In the previous lectures, we have studied models of cyber-physical systems. Hybrid programs provide a programming language for cyber-physical systems with the most prominent features being differential equations and nondeterminism alongside the usual classical control structures and discrete assignments. This gives powerful and flexible ways of modeling even very challenging systems and very complex control principles. This lecture will start studying ways of making sure that the resulting behavior meets the required correctness standards.

In 15-122 Principles of Imperative Computation, you have experienced how contracts can be used to make properties of programs explicit. You have seen how contracts can be checked dynamically at runtime, which, if they fail, alert you right away to flaws in the design of the programs. You have experienced first hand that it is much easier to find and fix problems in programs starting from the first contract that failed in the middle of the program, rather than from the mere observation that the final output is not as expected (which you may not notice either unless the output is checked dynamically).

Another aspect of contracts that you have had the opportunity to observe in Principles of Imperative Computation is that they can be used in proofs that show that every program run will satisfy the contracts. Unlike in dynamic checking, the scope of correctness arguments with proofs extends beyond the (clever) test cases that have been tried. Both uses of contracts, dynamic checking and rigorous proofs, are very helpful to check whether a system does what we intend it to, as has been argued on numerous occasions in various contexts in the literature, e.g., [Flo67, Hoa69, Pra76, Mey92, XJC09, PCL11, Log11].

The principles of contracts help cyber-physical systems [Pla08, Pla10, Pla13, DLTT13] as well. Yet, their use in proving may, arguably, be more important than their use in dynamic checking. The reason has to do with the physical impact of CPS and the (relative) non-negotiability of the laws of physics. The reader is advised to imagine a situation where a self-driving car is propelling him or her down the street. Suppose the car's control software is covered with contracts all over, but all of them are exclusively for dynamic checking, none have been proved. If that self-driving car speeds up to 100mph on a 55mph highway and drives up very close to a car in front of it, then dynamically checking the contract "distance to car in front should be more than 1 meter" does not help. If that contract fails, the car's software would know that it made a mistake, but it has become too late to do anything about it, because the brakes of the car will never work out in time. So the car would be "trapped in its own physics", in the sense that it has run out of all safe control options. There are still effective ways of making use of dynamic contract checking in CPS, but the design of those contracts then requires proof to ensure that safety is always maintained.

For those reasons, this course will focus on the role of proofs as correctness arguments much more than on dynamical checking of contracts. Because of the physical consequences of malfunctions, correctness requirements on CPS are also more stringent. And their proofs involve significantly more challenging arguments than in Principles of Imperative Computation. For those reasons, we will approach CPS proofs with much more rigor than what you have seen in Principles of Imperative Computation. But that is a story for a later lecture. The focus of today's lecture will be to understand CPS contracts and the first basics of reasoning about CPS.

This material is based on correctness specifications and proofs for CPS [Pla12c, Pla07, Pla08, Pla10]. We will come back to more details in later lectures, where we will also use the KeYmaera prover for verifying CPS [PQ08]. More information about safety and contracts can be found in [Pla10, Chapter 2.2,2.3].

2 The Adventures of a Bouncing Ball

Lecture 3 considered hybrid programs that model a choice of increasing acceleration or braking.

$$\begin{aligned}\Big(&\big((?x - o > 5; a := a + 1) \cup a := -b\big);\\ &x' = v, v' = a\Big)^*\end{aligned} \tag{1}$$

That model did perform interesting control choices and we could continue to study it in this lecture.

In order to sharpen our intuition about CPS, we will, however, study a very simple but also very intuitive system instead. Once upon a time, there was a little bouncing ball that had nothing else to do but bounce up and down the street until it was tired of doing that (Fig. 1). The bouncing ball was not much of a CPS, because the poor bouncing ball does not actually have any interesting decisions to make. But it nevertheless formed a perfectly reasonable hybrid system, because, after a closer look, it turns out to involve discrete and continuous dynamics. The continuous dynamics is caused by

Figure 1: Sample trajectory of a bouncing ball (plotted as position over time)

gravity, which is pulling the ball down and makes it fall from the sky in the first place. The discrete dynamics comes from the singular discrete event of what happens when the ball hits the ground and bounces back up. There are a number of ways of modeling the ball and its impact on the ground with physics. They include a whole range of different more or less realistic physical effects including gravity, aerodynamic resistance, the elastic deformation on the ground, and so on and so on. But the little bouncing ball didn't study enough physics to know anything about those effects. And so it had to go about understanding the world in easier terms. It was a clever bouncing ball, though, so it had experienced the phenomenon of sudden change and was trying to use that to its advantage.

If we are looking for a very simple model of what the bouncing ball does, it is easier to describe as a hybrid system. The ball at height h is falling subject to gravity:

$$h'' = -g$$

When it hits the ground, which is assumed at height $h = 0$, the ball bounces back and jumps back up in the air. Yet, as every child knows, the ball tends to come back up a little less high than before. Given enough time to bounce around, it will ultimately lie flat on the ground forever. Until it is picked up again and thrown high up in the air.

Let us model the impact on the ground as a discrete phenomenon and describe what happens so that the ball jumps back up then. One attempt of understanding this could be to make the ball jump back up rather suddenly by increasing its height by, say, 10 when it hit the ground $h = 0$:

$$\begin{aligned} &h'' = -g;\\ &\texttt{if}(h = 0)\, h := h + 10 \end{aligned} \tag{2}$$

Such a model may be useful for other systems, but would be rather at odds with our physical experience with bouncing balls, because the ball is indeed slowly climbing back up rather than suddenly being way up in the air again.

The bouncing ball ponders about what happens when it hits the ground. It does not suddenly get teleported to a new position above ground like (2) would suggest. Instead, the ball suddenly changes its direction. A moment ago, it used to fall down with a negative velocity (i.e. one that is pointing down into the ground) and suddenly climbs back up with a positive velocity (pointing up into the sky). In order to be able to write such a model, the velocity v will be made explicit in the bouncing ball's differential equation:

$$\begin{aligned} &h' = v, v' = -g;\\ &\texttt{if}(h = 0)\, v := -v \end{aligned} \tag{3}$$

Of course, something happens after the bouncing ball reversed its direction because it hit the ground. Physics continues until it hits the ground again.

$$\begin{aligned} &h' = v, v' = -g;\\ &\texttt{if}(h = 0)\, v := -v\\ &h' = v, v' = -g;\\ &\texttt{if}(h = 0)\, v := -v \end{aligned} \tag{4}$$

Then, of course, physics moves on again, so the model actually involves a repetition:

$$\begin{aligned} &\big(h' = v, v' = -g;\\ &\texttt{if}(h = 0)\, v := -v\big)^* \end{aligned} \tag{5}$$

Yet, the bouncing ball is now rather surprised. For if it follows that HP (5), it seems as if it should always be able to come back up to its initial height again. Excited about that possibility, it tries and tries again but never succeeds to bounce back up as high as it was before. So there must be something wrong with the model in (5), the ball concludes and sets out to fix (5).

Having observed itself rather carefully, the bouncing ball concludes that it feels slower when bouncing back up than it used to be when falling on down. Indeed, it feels less energetic on its way up. So its velocity must not only flip direction from down to up, at a bounce, but also seems to shrink in magnitude. The bouncing ball swiftly calls the corresponding damping factor c and quickly comes up with a better model of itself:

$$\begin{aligned} &\big(h' = v, v' = -g;\\ &\texttt{if}(h = 0)\, v := -cv\big)^* \end{aligned} \tag{6}$$

Yet, running that model in clever ways, the bouncing ball observes that model (6) could make it fall through the cracks in the ground. Terrified at that thought, the bouncing ball quickly tries to set the physics right, lest it falls through the cracks in space before it had a chance to fix its physics. The issue with (6) is that its differential equation isn't told when to stop. Yet, the bouncing ball luckily remembers that this is quite exactly what evolution domains were meant for. Above ground is what it wants to remain, and so $h \geq 0$ is what the ball asks dear physics to obey, since the table is of rather sturdy built:

$$\begin{aligned} &\big(h' = v, v' = -g \,\&\, h \geq 0;\\ &\texttt{if}(h = 0)\, v := -cv\big)^* \end{aligned} \tag{7}$$

Now, indeed, physics will have to stop evolving before gravity has made our little bouncing ball fall through the ground. Yet, physics could still choose to stop evolving while the ball is still high up in the sky. In that case, the ball will not yet be on the ground and line 2 of (7) would have no effect because $h \neq 0$ still. This is not a catastrophe, however, because the loop in (7) could simply repeat, which would allow physics to continue to evolve the differential equation further.

Quite happy with model (7) for itself, the bouncing ball goes on to explore whether the model does what the ball expects it to do.

3 Postcondition Contracts for CPS

Hybrid programs are interesting models for CPS. They describe the behavior of a CPS, ultimately captured by their semantics $\rho(\alpha)$, which is a reachability relation on states (Lecture 3). Yet, reliable development of CPS also needs a way of ensuring that the behavior will be as expected. So, for example, we may want the behavior of a CPS to always satisfy certain crucial safety properties. A robot, for example, should never do something unsafe like running over a human being.[1]

The little bouncing ball may consider itself less safety-critical, except that it may be interested in its own safety. It still wants to make sure that it couldn't ever fall through the cracks in the ground. And even though it would love to jump all the way up to the moon, the ball is also terrified of big heights and would never want to jump any higher than it was in the very beginning. So, when H denotes the initial height, the bouncing ball would love to know whether its height will always stay within $0 \leq h \leq H$ when following HP (7).

Scared of what otherwise might happen to it if $0 \leq h \leq H$ should ever be violated, the bouncing ball decides to make its goals for the HP (7) explicit. Fortunately, the bouncing ball excelled in the course Principles of Imperative Computation and recalls that contracts such as `@requires` and `@ensures` have been used in that course to make behavioral expectations for C0 programs explicit. Even though the bouncing ball clearly does not deal with a C0 program, but rather a hybrid program, it still puts `@ensures`(F) contracts in front of HP (7) to express that all runs of that HP are expected to lead only to states in which logical formula F is true. The bouncing ball even uses `@ensures` twice, once for each of its expectations.

$$\begin{aligned}
&\texttt{@ensures}(0 \leq h)\\
&\texttt{@ensures}(h \leq H)\\
&\big(h' = v, v' = -g \,\&\, h \geq 0;\\
&\ \texttt{if}(h = 0)\, v := -cv\big)^*
\end{aligned} \tag{8}$$

[1] Safety of robots has, of course, been aptly defined by Asimov [Asi42] with his Three Laws of Robotics:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

But their exact rendition in logic still remains a challenge.

4 Precondition Contracts for CPS

Having learned from the Principles of Imperative Computation experience, the little bouncing ball immediately starts thinking about whether the `@ensures` contracts in (8) would, in fact, always be true after running that HP. After all, the bouncing ball would really love to know that it can rely on that contract never failing.

Wondering about whether the `@ensures` contract in (8) would always succeed, the bouncing ball notices that this would have to depend on what values the bouncing ball starts with. It called H its initial height, but the HP (8) cannot know that. For one thing, the contracts in(8) would be hard to fulfill if $H = -5$, because $0 \leq h$ and $h \leq H$ can impossibly both be true then.

So the bouncing ball figures it should demand a `@requires` contract with the precondition $h = H$ to say that the height, h, of the bouncing ball is initially H. Because that still does not (obviously) ensure that $0 \leq h$ has a chance of holding, it requires $0 \leq H$ to hold initially:

$$\begin{aligned}
&\texttt{@requires}(h = H)\\
&\texttt{@requires}(0 \leq H)\\
&\texttt{@ensures}(0 \leq h)\\
&\texttt{@ensures}(h \leq H)\\
&\big(h' = v, v' = -g \,\&\, h \geq 0;\\
&\;\texttt{if}(h = 0)\, v := -cv\big)^*
\end{aligned} \tag{9}$$

5 Invariant Contracts for CPS

The little bouncing ball remembers the prominent role that invariants have played in the course Principles of Imperative Computation. So, the ball ventures including an invariant with its HP. In C0, invariants were associated with loops, e.g.

```
i = 0;
while (i < 10)
  //@loop_invariant 0 <= i && i <= 10;
  {
    i++;
  }
```

The bouncing ball, thus, figures that invariants for loops in HPs should also be associated with a loop, which is written α^* for nondeterministic repetition. After a moment's thought, the bouncing ball decides that falling through the cracks in the ground is still it's biggest worry, so the invariant it'd like to maintain is $h \geq 0$:

$$\begin{aligned}
&\texttt{@requires}(h = H)\\
&\texttt{@requires}(0 \leq H)\\
&\texttt{@ensures}(0 \leq h)\\
&\texttt{@ensures}(h \leq H)\\
&\big(h' = v, v' = -g \,\&\, h \geq 0;\\
&\ \texttt{if}(h = 0)\, v := -cv\big)^{*}\texttt{@invariant}(h \geq 0)
\end{aligned} \tag{10}$$

On second thought, the little bouncing ball is less sure what exactly the $\texttt{@invariant}(F)$ contract would mean for a CPS. So it decides to first give more thought to the proper way of phrasing CPS contracts and what they mean.

We will get back to the $\texttt{@invariant}(F)$ construct in a later lecture.

6 Logical Formulas for Hybrid Programs

CPS contracts play a very useful role in the development of CPS models and CPS programs. Using them as part of their design right from the very beginning is a good idea, probably even more crucial than it was in 15-122 Principles of Imperative Computation for the development of C0 programs, because CPS have more stringent requirements on safety.

Yet, we do not only want to program CPS, we also want to and have to understand thoroughly what they mean, what their contracts mean, and how we convince ourselves that the CPS contracts are respected by the CPS program. It turns out that this is where mere contracts are at a disadvantage compared to full logic. Logic allows not only the specification of a whole CPS program, but also an analytic inspection of its parts as well as argumentative relations between contracts and program parts.

Differential dynamic logic (dℒ) [Pla12c, Pla08, Pla12a, Pla07, Pla10] is the logic of hybrid systems that this courses uses for specification and verification of cyber-physical systems. There are more aspects of logic for cyber-physical systems [Pla12c, Pla12b], which will be studied (to some extent) in later parts of this course.

The most unique feature of differential dynamic logic for our purposes is that it allows us to refer to hybrid systems. Lecture 2 introduced first-order logic of real arithmetic.

Note 1 (Limits of first-order logic for CPS)**.** *First-order logic of real arithmetic is a crucial basis for describing what is true and false about CPS, because it allows us to refer to real-valued quantities like positions and velocities and their arithmetic relations. Yet, that is not enough, because first-order logic describes what is true in a single state of a system. It has no way of referring to what will be true in future states of a CPS, nor of describing the relationship of the initial state of the CPS to the final state of the CPS.*

Recall that this relationship, $\rho(\alpha)$, is what ultimately constitutes the semantics of HP α.

Note 2 (Differential dynamic logic principle)**.** *Differential dynamic logic (d$\mathcal{L}$) extends first-order logic of real arithmetic with operators that refer to the future states of a CPS in the sense of referring to the states that are reachable by running a given HP. The logic* d$\mathcal{L}$ *provides a modal operator $[\alpha]$, parametrized by α, that refers to all states reachable by HP α according to the reachability relation $\rho(\alpha)$ of its semantics. This modal operator can be placed in front of any* d$\mathcal{L}$ *formula ϕ. The* d$\mathcal{L}$ *formula*

$$[\alpha]\phi$$

expresses that all *states reachable by HP α satisfy formula ϕ.*

The logic d$\mathcal{L}$ *also provides a modal operator $\langle\alpha\rangle$, parametrized by α, can be placed in front of any* d$\mathcal{L}$ *formula ϕ. The* d$\mathcal{L}$ *formula*

$$\langle\alpha\rangle\phi$$

expresses that there is at least one *state reachable by HP α for which ϕ holds. The modalities $[\alpha]$ and $\langle\alpha\rangle$can be used to express necessary or possible properties of the transition behavior of α.*

An `@ensures`(E) postcondition for a HP α can be expressed directly as a logical formula in d$\mathcal{L}$:

$$[\alpha]E$$

So, the first CPS postcondition `@ensures`$(0 \leq h)$ for the bouncing ball HP in (8) can be stated as a d$\mathcal{L}$ formula:

$$[\big(h' = v, v' = -g \,\&\, h \geq 0;\ \texttt{if}(h = 0)\, v := -cv\big)^*]\, 0 \leq h \tag{11}$$

The second CPS postcondition `@ensures`$(h \leq H)$ for the bouncing ball HP in (8) can be stated as a d$\mathcal{L}$ formula as well:

$$[\big(h' = v, v' = -g \,\&\, h \geq 0;\ \texttt{if}(h = 0)\, v := -cv\big)^*]\, h \leq H \tag{12}$$

The logic d$\mathcal{L}$ allows all other logical operators from first-order logic, including conjunction ($\land$). So, the two d$\mathcal{L}$ formulas (11) and (12) can be stated together as a single d$\mathcal{L}$ formula:

$$\begin{aligned} &[\big(h' = v, v' = -g \,\&\, h \geq 0;\ \texttt{if}(h = 0)\, v := -cv\big)^*]\, 0 \leq h \\ \land\ &[\big(h' = v, v' = -g \,\&\, h \geq 0;\ \texttt{if}(h = 0)\, v := -cv\big)^*]\, h \leq H \end{aligned} \tag{13}$$

Stepping back, we could also have combined the two postconditions `@ensures`$(0 \leq h)$ and `@ensures`$(h \leq H)$ into a single postcondition `@ensures`$(0 \leq h \land h \leq H)$. The translation of that into d$\mathcal{L}$ would have gotten us an alternative way of combining both statements about the lower and upper bound on the height of the bouncing ball into a single d$\mathcal{L}$ formula:

$$[\big(h' = v, v' = -g \,\&\, h \geq 0;\ \texttt{if}(h = 0)\, v := -cv\big)^*]\, (0 \leq h \land h \leq H) \tag{14}$$

Which way of representing what we expect bouncing balls to do is better? Like (13) or like (14)? Are they equivalent? Or do they express different things?

It turns out that there is a very simple argument within the logic d$\mathcal{L}$ that shows that (13) and (14) are equivalent. And not just that those two particular logical formulas are equivalent but that the same equivalence holds for any d$\mathcal{L}$ formulas of this form. This will be investigated formally in a later lecture, but it is useful to observe now already to sharpen our intuition.

Having said that, do we believe d$\mathcal{L}$ formula (13) should be valid? Should (14) be valid? Before we study this question in any further detail, the first question should be what it means for a modal formula $[\alpha]\phi$ to be true. What is its semantics? Better yet, what exactly is its syntax in the first place?

7 Syntax of Differential Dynamic Logic

The formulas of differential dynamic logic are defined like the formulas of first-order logic of real arithmetic with the additional capability of using modal operators for any hybrid program α.

Definition 1 (d$\mathcal{L}$ formula)**.** The *formulas of differential dynamic logic* (d$\mathcal{L}$) are defined by the grammar (where ϕ, ψ are d$\mathcal{L}$ formulas, θ_1, θ_2 (polynomial) terms, x a variable, α a HP):

$$\phi, \psi \ ::= \ \theta_1 = \theta_2 \mid \theta_1 \geq \theta_2 \mid \neg\phi \mid \phi \land \psi \mid \phi \lor \psi \mid \phi \to \psi \mid \forall x\, \phi \mid \exists x\, \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi$$

Operators $>, \leq, <, \leftrightarrow$ *can be defined as usual, e.g.,* $\phi \leftrightarrow \psi \equiv (\phi \to \psi) \land (\psi \to \phi)$.

We use the notational convention that unary operators (including $\neg$ and quantifiers $\forall x, \exists x$ and modalities $[\alpha], \langle\alpha\rangle$)[2] bind stronger than binary operators. In particular, quantifiers and modal operators bind strong, i.e. their scope only extends to the formula immediately after. Thus, $[\alpha]\phi \land \psi \equiv ([\alpha]\phi) \land \psi$ and $\forall x\, \phi \land \psi \equiv (\forall x\, \phi) \land \psi$. In our notation, we also let $\land$ bind stronger than $\lor$, which binds stronger than $\to, \leftrightarrow$. We also associate $\to$ to the right so that $\phi \to \psi \to \varphi \equiv \phi \to (\psi \to \varphi)$. To avoid confusion, we do not adopt precedence conventions between $\to, \leftrightarrow$ but expect explicit parentheses. So $\phi \to \psi \leftrightarrow \varphi$ would be considered illegal and explicit parentheses are required to distinguish $\phi \to (\psi \leftrightarrow \varphi)$ from $(\phi \to \psi) \leftrightarrow \varphi$. Likewise $\phi \leftrightarrow \psi \to \varphi$ would be considered illegal and explicit parentheses are required to distinguish $\phi \leftrightarrow (\psi \to \varphi)$ from $(\phi \leftrightarrow \psi) \to \varphi$.

[2] Quantifiers are only quite arguably understood as unary operators. Yet, $\forall x$ is a unary operator on formulas while $\forall$ would be an operator with arguments of mixed syntactic categories. In a higher-order context, it can also be understood more formally by understanding $\forall x\, \phi$ as an operator on functions: $\forall(\lambda x.\phi)$. Similar cautionary remarks apply to the understanding of modalities as unary operators. The primary reason for adopting this understanding is that it simplifies the precedence rules.

8 Semantics of Differential Dynamic Logic

For d$\mathcal{L}$ formulas that are also formulas of first-order real arithmetic (i.e. formulas without modalities), the semantics of d$\mathcal{L}$ formulas is the same as that of first-order real arithmetic. The semantics of modalities $[\alpha]$ and $\langle\alpha\rangle$ quantifies over all ($[\alpha]$) or some ($\langle\alpha\rangle$) of the states reachable by following HP α, respectively.

Definition 2 (d$\mathcal{L}$ semantics)**.** The *satisfaction relation* $\nu \models \phi$ for a d$\mathcal{L}$ formula ϕ in state ν is defined inductively:

- $\nu \models (\theta_1 = \theta_2)$ iff $[\![\theta_1]\!]_\nu = [\![\theta_2]\!]_\nu$.
- $\nu \models (\theta_1 \geq \theta_2)$ iff $[\![\theta_1]\!]_\nu \geq [\![\theta_2]\!]_\nu$.
- $\nu \models \neg\phi$ iff $\nu \not\models \phi$, i.e. if it is not the case that $\nu \models \phi$.
- $\nu \models \phi \wedge \psi$ iff $\nu \models \phi$ and $\nu \models \psi$.
- $\nu \models \phi \vee \psi$ iff $\nu \models \phi$ or $\nu \models \psi$.
- $\nu \models \phi \rightarrow \psi$ iff $\nu \not\models \phi$ or $\nu \models \psi$.
- $\nu \models \phi \leftrightarrow \psi$ iff ($\nu \models \phi$ and $\nu \models \psi$) or ($\nu \not\models \phi$ and $\nu \not\models \psi$).
- $\nu \models \forall x\, \phi$ iff $\nu_x^d \models \phi$ for all $d \in \mathbb{R}$.
- $\nu \models \exists x\, \phi$ iff $\nu_x^d \models \phi$ for some $d \in \mathbb{R}$.
- $\nu \models [\alpha]\phi$ iff $\omega \models \phi$ for all ω with $(\nu, \omega) \in \rho(\alpha)$.
- $\nu \models \langle\alpha\rangle\phi$ iff $\omega \models \phi$ for some ω with $(\nu, \omega) \in \rho(\alpha)$.

If $\nu \models \phi$, then we say that ϕ is true at ν or that ν is a model of ϕ. A formula ϕ is *valid*, written $\vDash \phi$, iff $\nu \models \phi$ for all states ν. A formula ϕ is a *consequence* of a set of formulas Γ, written $\Gamma \vDash \phi$, iff, for each ν: ($\nu \models \psi$ for all $\psi \in \Gamma$) implies that $\nu \models \phi$.

9 CPS Contracts in Logic

Now that we know what truth and validity are, let's go back to the previous question. Is d$\mathcal{L}$ formula (13) valid? Is (14) valid? Indeed, they are equivalent, i.e. the d$\mathcal{L}$ formula

$$(13) \leftrightarrow (14)$$

is valid. Expanding the abbreviations that is the following d$\mathcal{L}$ formula is valid:

$$\begin{aligned}&\Big([\big(h' = v, v' = -g \,\&\, h \geq 0;\ \texttt{if}(h = 0)\, v := -cv\big)^*]\, 0 \leq h\\ &\land [\big(h' = v, v' = -g \,\&\, h \geq 0;\ \texttt{if}(h = 0)\, v := -cv\big)^*]\, h \leq H\Big)\\ \leftrightarrow\ &[\big(h' = v, v' = -g \,\&\, h \geq 0;\ \texttt{if}(h = 0)\, v := -cv\big)^*]\,(0 \leq h \land h \leq H)\end{aligned} \tag{15}$$

So if (13) is valid, then so should (14) be (Exercise 1). But is (13) valid?

Certainly, (13) is not true in a state ν where $\nu(h) < 0$, because from that initial state, no repetitions of the loop (which is allowed by nondeterministic repetition, Exercise 3), will lead to a state $\omega \overset{\text{def}}{=} \nu$ in which $\omega \not\models 0 \leq h$. Thus, (13) only has a chance of being valid in initial states that satisfy further assumptions, including $0 \leq h$ and $h \leq H$. In fact, that is what the preconditions were meant for in Sect. 4. How can we express a precondition contract in a d$\mathcal{L}$ formula?

Preconditions serve a very different role than postconditions do. Postconditions of HP α are what we want to hold true after every run of α. The meaning of a postcondition is what is rather difficult to express in first-order logic (to say the least). That is what d$\mathcal{L}$ has modalities for. Do we also need any extra logical operator to express preconditions?

The meaning of a precondition `@requires`(A) of a HP α is that it is assumed to hold before the HP starts. *If* A holds when the HP starts, then its postcondition `@ensures`(B) holds after all runs of HP α. What if A does not hold when the HP starts?

If precondition A does not hold initially, then all bets are off, because the person who started the HP did not obey its requirements, which says that it should only be run if its preconditions are met. The CPS contract `@requires`(A) `@ensures`(B) for a HP α promises that B will always hold after running α if A was true initially when α started. Thus, the meaning of a precondition can be expressed easily using an implication

$$A \rightarrow [\alpha]B \tag{16}$$

because an implication is valid if, in every state, its left-hand side is false or its right-hand side true. The implication (16) is valid ($\models A \rightarrow [\alpha]B$), if, indeed, for every state ν in which precondition A holds ($\nu \models A$), it is the case that all runs of HP α lead to states ω (with $(\nu, \omega) \in \rho(\alpha)$) in which postcondition B holds ($\omega \models B$). The d$\mathcal{L}$ formula (16) does not say what happens in states ν in which the precondition A does not hold ($\nu \not\models A$).

How does formula (16) talk about the runs of a HP and postcondition B again? Recall that the d$\mathcal{L}$ formula $[\alpha]B$ is true in exactly those states in which all runs of HP α lead only to states in which postcondition B is true. The implication in (16), thus, ensures that this holds in all (initial) states that satisfy precondition A.

Note 5 (Contracts to d$\mathcal{L}$ Formulas)**.** *Consider a HP α with a CPS contract using a single* `@requires`(A) *precondition and a single* `@ensures`(B) *postcondition:*

$$\begin{aligned}&\texttt{@requires}(A)\\&\texttt{@ensures}(B)\\&\alpha\end{aligned}$$

This CPS contract can be expressed directly as a logical formula in d$\mathcal{L}$*:*

$$A \rightarrow [\alpha]B$$

CPS contracts with multiple preconditions and multiple postconditions can directly be expressed as a d$\mathcal{L}$ formula as well (Exercise 4).

Recall HP (10), which is shown here in a slightly simplified form:

$$\begin{aligned}
&\texttt{@requires}(0 \leq h \land h = H)\\
&\texttt{@ensures}(0 \leq h \land h \leq H)\\
&\big(h' = v, v' = -g \,\&\, h \geq 0;\\
&\ \texttt{if}(h = 0)\, v := -cv\big)^*
\end{aligned} \tag{17}$$

The d$\mathcal{L}$ formula expressing that the CPS contract for HP (17) holds is:

$$0 \leq h \land h = H \rightarrow \big[\big(h' = v, v' = -g \,\&\, h \geq 0;\ \texttt{if}(h = 0)\, v := -cv\big)^*\big]\,(0 \leq h \land h \leq H) \tag{18}$$

So to find out whether (17) satisfies its CPS contract, we ask whether the d$\mathcal{L}$ formula (18) is valid.

In order to find out whether such a formula is valid, i.e. true in all states, we need some operational way that allows us to tell whether it is valid, because mere inspection of the semantics alone is not a particularly scalable way of approaching validity question.

10 Identifying Requirements of a CPS

Before trying to prove any formulas to be valid, it is a good idea to check whether all required assumptions have been found that are necessary for the formula to hold. So let us scrutinize d$\mathcal{L}$ formula (18) and ponder whether there are any circumstances under which it is not true. Even though the bouncing ball is a rather impoverished CPS (it suffers from a disparate lack of control), its immediate physical intuition still makes the ball an insightful example for illustrating how critical it is to identify the right requirements.

Maybe the first thing to notice is that the HP mentions g, which is meant to represent the standard gravity constant, but the formula (18) does not say. Certainly, if gravity were negative ($g < 0$), bouncing balls would function rather differently. They would suddenly be floating balls disappearing into the sky. So let's modify (18) to assume $g = 9.81$:

$$0 \leq h \land h = H \land g = 9.81 \rightarrow \big[\big(h' = v, v' = -g \,\&\, h \geq 0;\ \texttt{if}(h = 0)\, v := -cv\big)^*\big]\,(0 \leq h \land h \leq H) \tag{19}$$

Let's undo unnecessarily strict requirements right away, though. What would the bouncing ball do if it were set loose on the moon instead of on Earth? Would it still fall? Things are much lighter on the moon. Yet they still fall down ultimately, which is again the phenomenon known as gravity, just with a different constant (1.6 on the moon and 25.9 on Jupiter). Besides, none of those constants was particularly precise. Earth's gravity is more like 9.8067. The behavior of the bouncing ball depends on the value of that parameter g.

Note 6 (Parameters)**.** *A common feature of CPS is that their behavior is subject to parameters, which can have quite a non-negligible impact. Yet, it is very hard to determine precise values for parameters by measurements. When a particular concrete value for a parameter has been assumed to prove a property of a CPS, it is not clear whether that property holds for the true system, which may in reality have a slightly different parameter value.*

Instead of a numerical value for a parameter, our analysis can proceed by treating the parameter as a symbolic parameter*, i.e. a variable such as g, which is not assumed to hold a specific numerical value like 9.81. Instead, we would only assume certain constraints about the parameter, say $g > 1$ without choosing a specific value. If we then analyze the CPS with this symbolic parameter g, all analysis results will continue to hold for any concrete choice of g respecting its constraints (here $g > 1$). That results in a stronger statement about the system, which is less fragile as it does not break down just because the true g is ≈ 9.8067 rather than the previously assumed $g = 9.81$. Often times, those more general statements with symbolic parameters can even be easier to prove than statements about systems with specific magic numbers chosen for their parameters.*

In light of these thoughts, we could assume $9 < g < 10$ to be the gravity constant for Earth. Yet, we can also just consider all bouncing balls on all planets in the solar system or elsewhere at once by assuming only $g > 0$ instead of $g = 9.81$ as in (19), since this is the only aspect of gravity that the usual behavior of a bouncing ball depends on:

$$0 \leq h \land h = H \land g > 0 \rightarrow \left[\left(h' = v, v' = -g \,\&\, h \geq 0;\ \mathtt{if}(h = 0)\, v := -cv\right)^*\right] (0 \leq h \land h \leq H) \tag{20}$$

Do we expect dℒ formula (20) to be valid, i.e. true in all states? What could go wrong? The insight from modifying (18) to (19) and finally to (20) started with the observation that (18) did not include any assumptions about g. It is worth noting that (20) also does not assume anything about c. Bouncing balls clearly would not work as expected if $c > 1$, because such anti-damping would cause the bouncing ball to jump back up higher and higher and higher and ultimately as high up as the moon, clearly falsifying (20). Consequently, (20) only has a chance of being true when assuming that c is not too big:

$$\begin{aligned} &0 \leq h \land h = H \land g > 0 \land 1 > c \geq 0 \rightarrow \\ &\qquad \left[\left(h' = v, v' = -g \,\&\, h \geq 0;\ \mathtt{if}(h = 0)\, v := -cv\right)^*\right] (0 \leq h \land h \leq H) \end{aligned} \tag{21}$$

Is (21) valid now? Or does its truth depend on more assumptions that have not been identified yet? Now, all parameters (H, g, c) have some assumptions in (21). Is there some requirement we forgot about? Or did we find them all?

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

What about variable v? Why is there no assumption about it yet? Should there be one? Velocity v changes over time. What is its initial value allowed to be? What could go wrong?

Indeed, the initial velocity v of the bouncing ball could be positive ($v > 0$), which would make the bouncing ball climb initially, clearly exceeding its initial height H. This would correspond to the bouncing ball being thrown high up in the air in the beginning, so that its initial velocity v is upwards from its initial height $h = H$. Consequently, (21) has to be modified to assume $v \leq 0$ holds initially:

$$0 \leq h \land h = H \land v \leq 0 \land g > 0 \land 1 > c \geq 0 \rightarrow \\ \left[\left(h' = v, v' = -g \,\&\, h \geq 0;\ \mathtt{if}(h = 0)\, v := -cv\right)^*\right] (0 \leq h \land h \leq H) \quad (22)$$

Now there's finally assumptions about all parameters and variables of (22). That does not mean that we found the right assumptions, yet, obviously, but is still a good sanity check. Before wasting cycles on trying to prove or otherwise justify (22), let's try once more whether we can find an initial state ν that satisfies all assumptions $v \leq 0 \land 0 \leq h \land h = H \land g > 0 \land 1 > c \geq 0$ in the antecedent (i.e. left-hand side of the implication) of (22) so that ν does not satisfy the succedent (i.e. right-hand side of implication) of (22). Such an initial state ν falsifies (22) and would, thus, represent a *counterexample*.

Is there still a counterexample to (22)? Or have we successfully identified all assumptions so that it is now valid?

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

Formula (22) still has a problem. Even if the initial state satisfies all requirements in the antecedent of (22), the bouncing ball might still jump higher than it ought to, i.e. higher than its initial height H. That happens if the bouncing ball has a very big downwards velocity, so if v is a lot smaller than 0 (sometimes written $v \ll 0$). If v is a little smaller than 0, then the damping c will eat up enough the ball's kinetic energy so that it cannot jump back up higher than it was initially (H). But if v is a lot smaller than 0, then it starts falling down with so much kinetic energy that the damping on the ground does not slow it down enough, so the ball will come bouncing back higher than it was originally. Under which circumstance this happens depends on the relationship of the initial velocity and height to the damping coefficient.

We could explore this relationship in more detail. But it is actually easier to infer this relationship by conducting a proof. So we modify (22) to simply assume $v = 0$ initially:

$$0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 > c \geq 0 \rightarrow \\ \big[\big(h' = v, v' = -g \,\&\, h \geq 0;\ \mathtt{if}(h = 0)\, v := -cv\big)^*\big]\, (0 \leq h \land h \leq H) \quad (23)$$

Is d$\mathcal{L}$ formula (23) valid now? Or does it still have a counterexample?

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

It seems like all required assumptions have been identified to make the dℒ formula (23) valid so that the bouncing ball described in (23) satisfies the postcondition $0 \leq h \leq H$. But after so many failed starts and missing assumptions and requirements for the bouncing ball, it is a good idea to prove (23) once and for all beyond any doubt.

In order to be able to prove dℒ formula (23), however, we need to investigate how proving works. How can dℒ formulas be proved? And, since first-order formulas are dℒ formulas as well, one part of the question will be: how can first-order formulas be proved? How can real arithmetic be proved? How can requirements for the safety of CPS be identified systematically? All these questions will be answered in this course, but not all of them in this lecture.

In order to make sure we only need to worry about a minimal set of operators of dℒ for proving purposes, let's simply (23) by getting rid of `if-then-else` (Exercise 7):

$$0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 > c \geq 0 \to \\ \big[\big(h' = v, v' = -g \,\&\, h \geq 0;\ (?h = 0; v := -cv \cup ?h \neq 0)\big)^*\big]\,(0 \leq h \land h \leq H) \quad (24)$$

Observing the non-negligible difference between the original conjecture (19) and the revised and improved conjecture (24), leads us to often adopt the following principle.

Note 7 (Principle of Cartesian Doubt)**.** *In 1641, René Descartes suggested an attitude of systematic doubt where he would be skeptical about the truth of all believes until he found reason that they were justified. This principle is now known as Cartesian Doubt or skepticism.*

We will have perfect justifications: proofs. But until we have found proof, it is often helpful to adopt the principle of Cartesian Doubt in a very weak and pragmatic form. Before setting out on the journey to prove a conjecture, we first scrutinize it to see if we can find a counterexample that would make it false. For such a counterexample will not only save us a lot of misguided effort in trying to prove a false conjecture, but also helps us identify missing assumptions in conjectures and justifies the assumptions to be necessary. Surely, if, without assumption A, a counterexample to a conjecture exists, then A must be necessary.

11 Intermediate Conditions for CPS

Before proceeding any further with ways of proving dℒ formulas, let's simplify (24) grotesquely by removing the loop:

$$0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 > c \geq 0 \to \\ \big[h' = v, v' = -g \,\&\, h \geq 0; (?h = 0; v := -cv \cup ?h \neq 0)\big]\,(0 \leq h \land h \leq H) \quad (25)$$

Removing the loop clearly changes the behavior of the bouncing ball. It no longer bounces particularly well. All it can do now is fall and, if it reaches the floor, have its velocity reverted without actually climbing back up. So if we manage to prove (25), we certainly have not shown the actual dL formula (24). But it's a start, because the behavior modeled in (25) is a part of the behavior of (24). So it is useful (and easier) to understand (25) first.

The dL formula (25) has a number of assumptions $0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 > c \geq 0$ that can be used during the proof. It claims that the postcondition $0 \leq h \land h \leq H$ holds after all runs of the HP in the $[\cdot]$ modality. The top-level operator in the modality of (25) is a sequential composition (;), for which we need to find a proof argument.[3]

The HP in (25) follows a differential equation first and then, after the sequential composition (;), proceeds to run a discrete program $(?h = 0; v := -cv \cup ?h \neq 0)$. Depending on how long the HP follows its differential equation, the intermediate state after the differential equation and before the discrete program will be rather different.

Note 8 (Intermediate states of sequential compositions)**.** *This phenomenon happens in general for sequential compositions $\alpha; \beta$. The first HP α may reach a whole range of states, which represent intermediate states for the sequential composition $\alpha; \beta$, i.e. states that are final states for α and initial states for β. The intermediate states of $\alpha; \beta$ are the states μ in the semantics $\rho(\alpha; \beta)$ from Lecture 3:*

$$\rho(\alpha; \beta) = \rho(\beta) \circ \rho(\alpha) = \{(\nu, \omega) : (\nu, \mu) \in \rho(\alpha), (\mu, \omega) \in \rho(\beta)\}$$

Can we find a way of summarizing what all intermediate states between the differential equation and the discrete program of (25) have in common? They differ by how long the CPS has followed the differential equation.

If the system has followed the differential equation of (25) for time t, then the resulting velocity $v(t)$ at time t and height $h(t)$ at time t will be

$$v(t) = -gt, h(t) = H - \frac{g}{2}t^2 \tag{26}$$

This answer can be found by integrating or solving the differential equations. This knowledge (26) is useful but it is not (directly) clear how to use it to describe what all intermediate states have in common, because the time t in (26) is not available as a variable in the HP (25).[4] Can the intermediate states be described by a relation of the variables that (unlike t) are actually in the system? That is, an (arithmetic) formula relating h, v, g, H?

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

[3] The way we proceed here to prove (25) is actually not the recommended way. Later on, we will see a much easier way. But it is instructive to understand the more verbose approach we take first. This also prepares us for the challenges that lie ahead when proving properties of loops.

[4] Following these thoughts a bit further reveals how (26) can actually be used perfectly well to describe intermediate states when changing the HP (25) a little bit. But working with solutions is still not the way that gets us to the goal the quickest, usually.

One way of producing a relation from (26) is to get the units aligned and get rid of time t. Time drops out of the "equation" when squaring the identity for velocity:

$$v(t)^2 = g^2t^2, \quad h(t) = H - \frac{g}{2}t^2$$

and multiplying the identity for position by $2g$:

$$v(t)^2 = g^2t^2, \quad 2gh(t) = 2gH - 2\frac{g^2}{2}t^2$$

Then substituting the first equation into the second yields

$$2gh(t) = 2gH - v(t)^2$$

This equation does not depend on time t, so we expect it to hold after all runs of the differential equation irrespective of t:

$$2gh = 2gH - v^2 \tag{27}$$

We conjecture the intermediate condition (27) to hold in the intermediate state of the sequential composition in (25). In order to prove (25) we can decompose our reasoning into two parts. The first part will prove that the intermediate condition (27) holds after all runs of the first differential equation. The second part will assume (27) to hold and prove that all runs of the discrete program in (25) from any state satisfying (27) satisfy the postcondition $0 \leq h \wedge h \leq H$.

Note 9 (Intermediate conditions as contracts for sequential composition)**.** *For a HP that is a sequential composition $\alpha; \beta$ an* intermediate condition *is a formula that characterizes the intermediate states in between HP α and β. That is, for a* d$\mathcal{L}$ *formula*

$$A \to [\alpha; \beta]B$$

an intermediate condition is a formula E such that the following d$\mathcal{L}$ *formulas are valid:*

$$A \to [\alpha]E \qquad \textit{and} \qquad E \to [\beta]B$$

The first d$\mathcal{L}$ *formula expresses that intermediate condition E characterizes the intermediate states accurately, i.e. E actually holds after all runs of HP α from states satisfying A. The second* d$\mathcal{L}$ *formula says that the intermediate condition E characterizes intermediate states well enough, i.e. E is all we need to know about a state to conclude that all runs of β end up in B. That is, from all states satisfying E (in particular from those that result by running α from a state satisfying A), B holds after all runs of β.*

For proving (25), we conjecture that (27) is an intermediate condition, which requires us to prove the following two d$\mathcal{L}$ formulas:

$$\begin{aligned} 0 \leq h \wedge h = H \wedge v = 0 \wedge g > 0 \wedge 1 > c \geq 0 \to [h' = v, v' = -g \,\&\, h \geq 0] 2gh = 2gH - v^2 \\ 2gh = 2gH - v^2 \to [?h = 0; v := -cv \cup ?h \neq 0] \, (0 \leq h \wedge h \leq H) \end{aligned} \tag{28}$$

Let's focus on the latter formula. Do we expect to be able to prove it? Do we expect it to be valid?

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

The second formula of (28) claims that $0 \leq h$ holds after all runs of $?h = 0; v := -cv \cup ?h \neq 0$ from all states that satisfy $2gh = 2gH - v^2$. That is a bit much to hope for, however, because $0 \leq h$ is not even ensured in the precondition of this second formula. So the second formula of (28) is not valid. How can this problem be resolved? By adding $0 \leq h$ into the intermediate condition, thus, requiring us to prove:

$$\begin{aligned} 0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 > c \geq 0 \rightarrow [h' = v, v' = -g \,\&\, h \geq 0](2gh = 2gH - v^2 \land h \geq 0) \\ 2gh = 2gH - v^2 \land h \geq 0 \rightarrow [?h = 0; v := -cv \cup ?h \neq 0]\,(0 \leq h \land h \leq H) \end{aligned} \tag{29}$$

Proving the first formula in (29) requires us to handle differential equations, which we will get to later. The second formula in (29) is the one whose proof is discussed first.

12 A Proof of Choice

The second formula in (29) has a nondeterministic choice ($\cup$) as the top-level operator in its $[\cdot]$ modality. How can we prove a formula of the form

$$A \rightarrow [\alpha \cup \beta]B \tag{30}$$

Recalling its semantics from Lecture 3,

$$\rho(\alpha \cup \beta) = \rho(\alpha) \cup \rho(\beta)$$

HP $\alpha \cup \beta$ has two possible behaviors. It could run as HP α does or as β does. And it is chosen nondeterministically which of the two behaviors happens. Since the behavior of $\alpha \cup \beta$ could be either α or β, proving (30) requires proving B to hold after α and after β. More precisely, (30) assumes A to hold initially, otherwise (30) is vacuously true. Thus, proving (30) allows us to assume A and requires us to prove that B holds after all runs of α (which is permitted behavior for $\alpha \cup \beta$) and to prove that, assuming A holds initially, that B holds after all runs of β (which is also permitted behavior of $\alpha \cup \beta$).

Note 10 (Proving choices). *For a HP that is a nondeterministic choice $\alpha \cup \beta$, we can prove*

$$A \rightarrow [\alpha \cup \beta]B$$

by proving the following d$\mathcal{L}$ *formulas:*

$$A \rightarrow [\alpha]B \qquad \textit{and} \qquad A \rightarrow [\beta]B$$

Using these thoughts on the second formula of (29), we could prove that formula if we would manage to prove both of the following d$\mathcal{L}$ formulas:

$$\begin{aligned} 2gh = 2gH - v^2 \land h \geq 0 \rightarrow [?h = 0; v := -cv]\,(0 \leq h \land h \leq H) \\ 2gh = 2gH - v^2 \land h \geq 0 \rightarrow [?h \neq 0]\,(0 \leq h \land h \leq H) \end{aligned} \tag{31}$$

13 Proofs of Tests

Consider the second formula of (31). Proving it requires us to understand how to handle a test $?H$ in a modality $[?H]$. The semantics of a test $?H$ from Lecture 3

$$\rho(?H) = \{(\nu,\nu) \;:\; \nu \models H\} \tag{32}$$

says that a test $?H$ completes successfully without changing the state in any state ν in which H holds (i.e. $\nu \models H$) and fails to run in all other states (i.e. where $\nu \not\models H$). How can we prove a formula with a test:

$$A \rightarrow [?H]B \tag{33}$$

This formula expresses that from all initial states satisfying A all runs of $?H$ reach states satisfying B. When is there a run of $?H$ at all? There is a run from state ν if and only if H holds in ν. So the only cases to worry about those initial states that satisfy H as, otherwise, the HP in (33) cannot execute at all by fails miserably so that the run is discarded. Hence, we get to assume H holds, as the HP $?H$ does not otherwise execute. In all states that the HP $?H$ reaches from states satisfying A, (33) conjectures that B holds. Now, by (32), the final states that $?H$ reaches are the same as the initial state (as long as they satisfy H so that HP $?H$ can be executed at all). That is, postcondition B needs to hold in all states from which $?H$ runs (i.e. that satisfy H) and that satisfy the precondition A. So (33) can be proved by proving

$$A \wedge H \rightarrow B$$

Note 11 (Proving tests)**.** *For a HP that is a test $?H$, we can prove*

$$A \rightarrow [?H]B$$

by proving the following d$\mathcal{L}$ *formula:*

$$A \wedge H \rightarrow B$$

Using this for the second formula of (31), Note 11 reduces proving the second formula of (31)

$$2gh = 2gH - v^2 \wedge h \geq 0 \rightarrow [?h \neq 0]\,(0 \leq h \wedge h \leq H)$$

to proving

$$2gh = 2gH - v^2 \wedge h \geq 0 \wedge h \neq 0 \rightarrow 0 \leq h \wedge h \leq H \tag{34}$$

Now we are left with arithmetic that we need to prove. Proofs for arithmetic and propositional logical operators such as $\wedge$ and $\rightarrow$ will be considered in a later lecture. For now, we notice that the formula $0 \leq h$ in the right-hand side of $\rightarrow$ seems justified by assumption $h \geq 0$. And that $h \leq H$ does not exactly have a justification in (34), because we lost the assumptions about H somewhere.

How could that happen? We used to know $h \leq H$ in (25). We also still knew about it in the first formula of (29). But we let it disappear from the second formula of (29), because we chose an intermediate condition that was too weak when constructing (29).

This is a common problem in trying to prove properties of CPS or of any other mathematical statements. One of our intermediate steps might have been too weak, so that our attempt of proving it fails and we need to revisit how we got there. For sequential compositions, this is actually a nonissue as soon as we move on (in the next lecture) to a proof technique that is more useful than the intermediate conditions from Note 9. But similar difficulties can arise in other parts of proof attempts.

In this case, the fact that we lost $h \leq H$ can be fixed by including it in the intermediate conditions, because it can be shown to hold after the differential equation still. Other crucial assumptions have also suddenly disappeared in our reasoning. An extra assumption $1 > c \geq 0$, for example, is crucially needed to justify the first formula of (31). It is somewhat easier to see why that particular assumption can be added to the intermediate contract without changing the argument much. The reason is that c never ever changes during the system run.

Note 12. *It is very difficult to come up with bug-free code. Just thinking about your assumptions really hard does not ensure correctness, but we can gain confidence that our system does what we want it to by proving that certain properties are satisfied.*

Changing the assumptions and arguments in a hybrid program around during the search for a proof of safety is something that happens frequently. It is easy to make subtle mistakes in informal arguments such as I need to know C here and I would know C if I had included it here or there, so now I hope the argument holds. This is one of many reasons why we are better off if our CPS proofs are rigorous, because we would rather not end up in trouble because of a subtle aw in a correctness argument. A formal proof calculus for differential dynamic logic (d$\mathcal{L}$) will help us avoid the pitfalls of informal arguments. The theorem prover KeYmaera that you will use in this course implements a proof calculus for d$\mathcal{L}$.

A related observation from our informal arguments in this lecture is that we desperately need a way to keep an argument consistent as a single argument justifying one conjecture. Quite the contrary to the informal loose threads of argumentation we have pursued in this lecture for the sake of developing an intuition. Consequently, we will investigate what constitutes an actual proof in subsequent lectures. A proof in which the relationship of premises to conclusions via proof steps is rigorous.

Moreover, there's two loose ends in our arguments. For one, the differential equation in (29) is still waiting for an argument that could help us prove it. Also, the assignment in (31) still needs to be handled and its sequential composition needs an intermediate contract.

Exercises

Exercise 1. Let A, B be $\mathsf{d}\mathcal{L}$ formulas. Suppose $A \leftrightarrow B$ is valid and A is valid. Is B valid? Prove or disprove.

Exercise 2. Let A, B be $\mathsf{d}\mathcal{L}$ formulas. Suppose $A \leftrightarrow B$ is true in state ν and A is true in state ν. That is, $\nu \models A \leftrightarrow B$ and $\nu \models A$. Is B true in state ν? Prove or disprove. Is B valid? Prove or disprove.

Exercise 3. Let α be an HP. Let ν be a state with $\nu \not\models \phi$. Does $\nu \not\models [\alpha^*]\phi$ hold? Prove or disprove.

Exercise 4. Suppose you have a HP α with a CPS contract using multiple preconditions $A_1, \dots, A_n$ and multiple postconditions $B_1, \dots, B_m$:

$$\begin{aligned}
&\texttt{@requires}(A_1)\\
&\texttt{@requires}(A_2)\\
&\vdots\\
&\texttt{@requires}(A_n)\\
&\texttt{@ensures}(B_1)\\
&\texttt{@ensures}(B_2)\\
&\vdots\\
&\texttt{@ensures}(B_m)\\
&\alpha
\end{aligned}$$

How can this CPS contract be expressed in a $\mathsf{d}\mathcal{L}$ formula?

Exercise 5. For each of the following $\mathsf{d}\mathcal{L}$ formulas, determine if they are valid, satisfiable, *and/or* unsatisfiable:

1. $[?x \geq 0]x \geq 0$.
2. $[?x \geq 0]x \leq 0$.
3. $[?x \geq 0]x < 0$.
4. $[?\mathit{true}]\mathit{true}$.
5. $[?\mathit{true}]\mathit{false}$.
6. $[?\mathit{false}]\mathit{true}$.
7. $[?\mathit{false}]\mathit{false}$.
8. $[x' = 1 \,\&\, \mathit{true}]\mathit{true}$.
9. $[x' = 1 \,\&\, \mathit{true}]\mathit{false}$.

10. $[x' = 1 \,\&\, \mathit{false}]\mathit{true}$.

11. $[x' = 1 \,\&\, \mathit{false}]\mathit{false}$.

12. $[(x' = 1 \,\&\, \mathit{true})^*]\mathit{true}$.

13. $[(x' = 1 \,\&\, \mathit{true})^*]\mathit{false}$.

14. $[(x' = 1 \,\&\, \mathit{false})^*]\mathit{true}$.

15. $[(x' = 1 \,\&\, \mathit{false})^*]\mathit{false}$.

Exercise 6. What would happen with the bouncing ball if $c < 0$? Consider a variation of the arguments in Sect. 10 where instead of the assumption in (21), you assume $c < 0$. Is the formula valid? What would happen with a bouncing ball of damping $c = 1$?

Exercise 7. We went from (23) to (24) by removing an `if-then-else`. Explain how this works and justify why it is okay to do this transformation. It is okay to focus only on this case, even though the argument is more general.

Exercise 8 (**). Sect. 11 used a mix of a systematic and ad-hoc approach for producing an intermediate condition that was based on solving and combining differential equations. Can you think of a more systematic rephrasing?

References

[Asi42] Isaac Asimov. Runaround, 1942.

[DBL12] *Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25–28, 2012*. IEEE, 2012.

[DLTT13] Patricia Derler, Edward A. Lee, Stavros Tripakis, and Martin Törngren. Cyber-physical system design contracts. In Chenyang Lu, P. R. Kumar, and Radu Stoleru, editors, *ICCPS*, pages 109–118. ACM, 2013.

[Flo67] Robert W. Floyd. Assigning meanings to programs. In J. T. Schwartz, editor, *Mathematical Aspects of Computer Science, Proceedings of Symposia in Applied Mathematics*, volume 19, pages 19–32, Providence, 1967. AMS.

[Hoa69] Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.

[Log11] Francesco Logozzo. Practical verification for the working programmer with codecontracts and abstract interpretation - (invited talk). In Ranjit Jhala and David A. Schmidt, editors, *VMCAI*, volume 6538 of *LNCS*, pages 19–22. Springer, 2011. doi:10.1007/978-3-642-18275-4_3.

[Mey92] Bertrand Meyer. Applying ”design by contract”. *Computer*, 25(10):40–51, October 1992.

[PCL11] Frank Pfenning, Thomas J. Cortina, and William Lovas. Teaching imperative programming with contracts at the freshmen level. 2011.

[Pla07] André Platzer. Differential dynamic logic for verifying parametric hybrid systems. In Nicola Olivetti, editor, *TABLEAUX*, volume 4548 of *LNCS*, pages 216–232. Springer, 2007. doi:10.1007/978-3-540-73099-6_17.

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

[Pla10] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14509-4.

[Pla12a] André Platzer. The complete proof theory of hybrid systems. In *LICS* [DBL12], pages 541–550. doi:10.1109/LICS.2012.64.

[Pla12b] André Platzer. Dynamic logics of dynamical systems. *CoRR*, abs/1205.4788, 2012. arXiv:1205.4788.

[Pla12c] André Platzer. Logics of dynamical systems. In *LICS* [DBL12], pages 13–24. doi:10.1109/LICS.2012.13.

[Pla13] André Platzer. Teaching CPS foundations with contracts. In *CPS-Ed*, pages 7–10, 2013.

[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.

[Pra76] Vaughan R. Pratt. Semantical considerations on Floyd-Hoare logic. In *FOCS*, pages 109–121. IEEE, 1976.

[XJC09] Dana N. Xu, Simon L. Peyton Jones, and Koen Claessen. Static contract checking for haskell. In Zhong Shao and Benjamin C. Pierce, editors, *POPL*, pages 41–52. ACM, 2009. doi:10.1145/1480881.1480889.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Dynamical Systems & Dynamic Axioms

André Platzer

Carnegie Mellon University

Lecture 5

1 Introduction

Lecture 4 demonstrated how useful and crucial CPS contracts are for CPS. Their role and understanding goes beyond dynamic testing, though. In CPS, proven CPS contracts are infinitely more valuable than dynamically tested contracts, because dynamical tests of contracts at runtime of a CPS generally leave open very little flexibility for reacting to them in any safe way. After all, the failure of a contract indicates that some safety condition that was expected to hold is not longer true. Unless provably sufficient safety margin and fallback plans remain, the system is already in trouble then.[1]

Consequently, CPS contracts really shine in relation to how they are proved for CPS. Understanding how to prove CPS contracts requires us to understand the dynamical effects of hybrid programs in more detail. This deeper understanding of the effects of hybrid program statements is not only useful for conducting proofs but also for developing and sharpening our intuition about hybrid programs for CPS. This phenomenon illustrates a more general point that proof and effect (and/or meaning) are intimately linked and that truly understanding effect is ultimately the same as, as well as a prerequisite to, understanding how to prove properties of that effect [Pla12c, Pla12a, Pla10]. You may have seen this point demonstrated amply already in other courses from the Principles of Programming Languages group at CMU.

The route that we choose to get to this level of understanding is one that involves a closer look at dynamical systems and Kripke models, or rather, the effect that hybrid programs have on them. This will enable us to devise authoritative proof principles for differential dynamic logic and hybrid programs [Pla12c, Pla12a, Pla10, Pla08]. While there are many more interesting things to say about dynamical systems and Kripke

[1] Although, in combination with formal verification, the Simplex architecture exploits this relationship of dynamic contracts for safety purposes [SKSC98].

structures, this lecture will limit information to the truly essential parts that are crucial right now and leave more elaboration for later lectures.

More information can be found in [Pla12b, Pla12c] as well as [Pla10, Chapter 2.3].

2 A Proof of Choice (Continued)

Recall the bouncing ball from Lecture 4, with repetition removed just to simplify the discussion for illustration purposes:

$$0 \le h \land h = H \land v = 0 \land g > 0 \land 1 > c \ge 0 \to \\ \big[h' = v, v' = -g \,\&\, h \ge 0; (?h = 0; v := -cv \cup ?h \ne 0)\big]\,(0 \le h \land h \le H) \tag{1}$$

In order to try to prove the above formula, we have convinced ourselves with a number of steps of argumentation that we should try to prove the following two formulas (and many others):

$$0 \le h \land h = H \land v = 0 \land g > 0 \land 1 > c \ge 0 \to [h' = v, v' = -g \,\&\, h \ge 0](2gh = 2gH - v^2 \land g > 0) \\ 2gh = 2gH - v^2 \land g > 0 \to [?h = 0; v := -cv \cup ?h \ne 0]\,(0 \le h \land h \le H) \tag{2}$$

In our attempt of proving the latter formula, we used the following principle:

Note 1 (Proving choices). *For a HP that is a nondeterministic choice $\alpha \cup \beta$, we can prove*

$$A \to [\alpha \cup \beta]B \tag{3}$$

by proving the following d$\mathcal{L}$ *formulas:*

$$A \to [\alpha]B \qquad \text{and} \qquad A \to [\beta]B$$

Note 2 (Proving choices: proof-rule style). *Note 1 is captured more concisely in the following proof rule:*

$$\text{(R1)}\ \frac{A \to [\alpha]B \quad A \to [\beta]B}{A \to [\alpha \cup \beta]B}$$

If we can prove all premises (above rule bar) of a proof rule, then that proof rule infers the conclusion (below rule bar).

Alas, the way we have been using proof rules so far is the other way around. We had been looking at a formula such as the second formula of (2) that has the shape of the conclusion of a rule such as R1. And then we went on trying to prove the premises of that proof rule instead. This conclusion-to-premise style of using our proof rules is perfectly acceptable and useful as well. Should we ever succeed in proving the premises of R1, that proof rule would allow us to infer its conclusion too. In this way, proof rules are even useful in directing us at which formulas we should try to prove next: the premises of the instantiation of that rule.

Using these thoughts on the second formula of (2), we could prove that formula using proof rule R1 if we would manage to prove both of its premises, which, in this instance, are the following d$\mathcal{L}$ formulas:

$$
\begin{gathered}
2gh = 2gH - v^2 \land g > 0 \to [?h = 0; v := -cv]\,(0 \leq h \land h \leq H) \\
2gh = 2gH - v^2 \land g > 0 \to [?h \neq 0]\,(0 \leq h \land h \leq H)
\end{gathered}
\tag{4}
$$

Before proceeding with proofs of (4), revisit the reasoning that led to the principle in Note 2. We said that (3) can be justified by proving that, when assuming A, all runs of α lead to states satisfying B *and* all runs of β lead to B states. Is that argument reflected directly in Note 2?

Kind of, but not quite, because there is a minor difference. Our informal argument assumed A once and concluded both $[\alpha]B$ and $[\beta]B$ from A. The principle captured in Note 2 assumes A to prove $[\alpha]B$ and then, separately, assumes A again to prove $[\beta]B$. These two arguments are clearly closely related, but still slightly different. Can we formalize and follow the original argument directly somehow? Or is Note 2 our only chance?

Following the original argument, we would argue that (3) holds by proving

$$A \to ([\alpha]B \land [\beta]B)$$

or, since the parentheses are superfluous according to the usual precedence rules:

$$A \to [\alpha]B \land [\beta]B \tag{5}$$

Is there a direct way how we can justify going from (3) to (5)? Preferably one that simultaneously justifies going from (3) to the formulas identified in Note 2 as well.

These considerations will take us to a more general and more elegant proof principle than R1, to a more refined understanding of the behavior of nondeterministic choices, and to a way of justifying proof rules as being sound.

3 Dynamic Axioms for Nondeterministic Choices

Recall the semantics of nondeterministic choices from Lecture 3:

$$\rho(\alpha \cup \beta) = \rho(\alpha) \cup \rho(\beta) \tag{6}$$

Remember that $\rho(\alpha)$ is a reachability relation on states, where $(\nu, \omega) \in \rho(\alpha)$ iff HP α can run from state ν to state ω. Let us illustrate graphically what (6) means:

According to $\rho(\alpha)$, a number of states ω_i are reachable by running HP α from some initial state ν. According to $\rho(\beta)$, a number of (possibly other) states ω_i are reachable by running HP β from the same initial state ν. By (6), running $\alpha \cup \beta$ from ν can give us any of those possible outcomes. And there was nothing special about the initial state ν. The same principle holds for all other states.

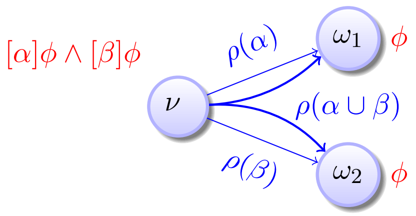


Figure 1: Illustration of transition semantics of $\alpha \cup \beta$

Note 3 ($\cup$). *The nondeterministic choice $\alpha \cup \beta$ can lead to exactly the states to which either α could take us or to which β could take us or to which both could lead. The dynamic effect of a nondeterministic choice $\alpha \cup \beta$ is that running it at any time either results in a behavior of α or of β. So both the behaviors of α and β are possible when running $\alpha \cup \beta$.*

If we want to understand whether and where d$\mathcal{L}$ formula $[\alpha \cup \beta]\phi$ is true, we need to understand which states the modality $[\alpha \cup \beta]$ refers to. In which states does ϕ have to be true so that $[\alpha \cup \beta]\phi$ is true in state ν?

By definition of the semantics, ϕ needs to be true in all states that $\alpha \cup \beta$ can reach according to $\rho(\alpha \cup \beta)$ from ν for $[\alpha \cup \beta]\phi$ to be true in ν. Referring to (6) or looking at Fig. 1, shows us that this includes exactly all states that α can reach from ν according to $\rho(\alpha)$, hence $[\alpha]\phi$ has to be true in ν. And that it also includes all states that β can reach from ν, hence $[\beta]\phi$ has to be true in ν.

Consequently,

$$\nu \models [\alpha]\phi \quad \text{and} \quad \nu \models [\beta]\phi \tag{7}$$

are necessary conditions for

$$\nu \models [\alpha \cup \beta]\phi \tag{8}$$

That is, unless (7) holds, (8) cannot possibly hold. So (7) is necessary for (8). Are there any states missing? Are there any states that (8) would require to satisfy ϕ, which (7) does not already ensure to satisfy ϕ? No, because, by (6), $\alpha \cup \beta$ does not admit any behavior that neither α nor β can exhibit. Hence (7) is also sufficient for (8), i.e. (7) implies (8).

Thus, when adopting a more logical language again, this justifies:

$$\nu \models [\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \wedge [\beta]\phi$$

This reasoning did not depend on the particular state ν but holds for all ν. Therefore,

$$\models [\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \wedge [\beta]\phi$$

Exciting! We have just proved our first axiom to be sound:

Lemma 1 ([∪] soundness)**.** *The axiom of choice is* sound*, i.e. all its instances are valid:*

$$([\cup])\ [\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \land [\beta]\phi$$

Nondeterministic choices split into their alternatives in axiom [∪]. From right to left: If all α runs lead to states satisfying ϕ (i.e., $[\alpha]\phi$ holds) and all β runs lead to states satisfying ϕ (i.e., $[\beta]\phi$ holds), then all runs of HP $\alpha \cup \beta$, which may choose between following α and following β, also lead to states satisfying ϕ (i.e., $[\alpha \cup \beta]\phi$ holds). The converse implication from left to right holds, because $\alpha \cup \beta$ can run all runs of α and all runs of β, so all runs of α (and of β) lead to states satisfying ϕ if that holds for all runs of $[\beta]\phi$.

From now on, every time we see a formula of the form $[\alpha \cup \beta]\phi$, we can remember that axiom [∪] knows a formula, namely $[\alpha]\phi \land [\beta]\phi$ that is equivalent to it. Of course, whenever we find a formula of the form $[\gamma \cup \delta]\psi$, we also remember that axiom [∪] knows a formula, namely $[\gamma]\psi \land [\delta]\psi$ that is equivalent to it, just by instantiation of axiom [∪].

Armed with this axiom [∪] at our disposal, we can now easily do a proof step from (3) to (5) just by invoking the equivalence that [∪] justifies. Let's elaborate. We want to prove:

$$A \to [\alpha \cup \beta]B \tag{3}$$

By [∪], or rather an instance of [∪] formed by using B for ϕ, we know:

$$[\alpha \cup \beta]B \leftrightarrow [\alpha]B \land [\beta]B \tag{9}$$

Since (9) is a valid equivalence, replacing the place where the left-hand side of (9) occurs in (3) by the right-hand side of (9) gives us a formula that is equivalent to (3):

$$A \to [\alpha]B \land [\beta]B \tag{5}$$

After all, according to the valid equivalence (9) justified by axiom [∪], (5) can be obtained from (3) just by replacing a formula with one that is equivalent.

Actually, stepping back, the same argument can be made to go from (5) to (3) instead of from (3) to (5). Both ways of using [∪] are perfectly fine. Although the direction that gets rid of the $\cup$ operator tends to be much more useful, because it made progress (getting rid of an HP operator). Yet axiom [∪] can also be useful in many more situations than rule R1. For example, if want to prove a d$\mathcal{L}$ formula

$$[\alpha \cup \beta]A \to B$$

where $[\alpha \cup \beta]$ is on the left-hand side of an implication, then axiom [∪] justifies that it is enough to prove the following d$\mathcal{L}$ formula instead:

$$[\alpha]A \land [\beta]A \to B$$

This inference cannot be justified with proof rule R1, but would need a separate proof rule such as

$$\text{(R3)}\ \frac{[\alpha]A \land [\beta]A \to B}{[\alpha \cup \beta]A \to B}$$

Yet, axiom $[\cup]$ justifies both R1 and R3 and many other uses of splitting a boxed choice into a conjunction. Axiom $[\cup]$ is, thus, more fundamental.

A general principle behind the d$\mathcal{L}$ axioms is most noticeable in axiom $[\cup]$. All equivalence axioms of d$\mathcal{L}$ are primarily intended to be used by reducing the formula on the left to the (structurally simpler) formula on the right. Such a reduction symbolically decomposes a property of a more complicated system into separate properties of easier fragments α and β. This decomposition makes the problem tractable and is good for scalability purposes. For these symbolic structural decompositions, it is very helpful that d$\mathcal{L}$ is a full logic that is closed under all logical operators, including disjunction and conjunction, for then both sides in $[\cup]$ are d$\mathcal{L}$ formulas again (unlike in Hoare logic [Hoa69]). This also turns out to be an advantage for computing invariants [PC08, PC09, Pla10], which will be discussed much later in this course.

The definition of soundness was not specific to axiom $[\cup]$, but applies to all d$\mathcal{L}$ axioms.

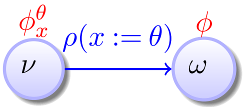
Definition 2 (Soundness)**.** An axiom is *sound* iff all its instances are valid.

4 Dynamic Axioms for Assignments

Axiom $[\cup]$ allows us to understand and handle $[\alpha \cup \beta]$ properties. If we find similar axioms for the other operators of hybrid programs, then we have a way of handling ”all” other hybrid programs, too.

Consider discrete assignments. Recall from Lecture 4 that:

$$\rho(x := \theta) = \{(\nu, \omega)\ :\ \omega = \nu \text{ except that } [\![x]\!]_\omega = [\![\theta]\!]_\nu\}$$



Lemma 3 ($[:=]$ soundness)**.** *The assignment axiom is sound:*

$$([:=])\ [x := \theta]\phi(x) \leftrightarrow \phi(\theta)$$

Axiom $[:=]$ is Hoare's assignment rule. It uses substitutions to axiomatize discrete assignments. To show that $\phi(x)$ is true after a discrete assignment, axiom $[:=]$ shows that it has been true before, when substituting the affected variable x with its new value θ.

Formula $\phi(\theta)$ is obtained from $\phi(x)$ by *substituting* θ for x at all occurrences of x, provided x does not occur in the scope of a quantifier or modality binding x or a variable of θ.

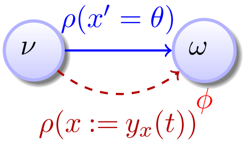
Note 7 (Bound variables)**.** *A modality containing $x :=$ or x' outside the scope of tests $?H$ or evolution domain constraints* binds x, *because it may change the value of x. A quantifier $\forall x$ or $\exists x$ also binds variable x.*

Substitutions are defined as usual [Pla10, Chapter 2.5.1].

5 Dynamic Axioms for Differential Equations

Recall from Lecture 4 that

$$\rho(x' = \theta \,\&\, H) = \{(\varphi(0), \varphi(r)) \;:\; \varphi(t) \models x' = \theta \text{ and } \varphi(t) \models H \text{ for all } 0 \leq t \leq r \\ \text{for a solution } \varphi : [0, r] \to \mathcal{S} \text{ of any duration } r\}$$



One possible approach of proving properties of differential equations is to work with a solution if one is available (and expressible in the logic).

Lemma 4 ([′] soundness)**.** *The solution axiom is sound:*

$$([']) \;\; [x' = \theta]\phi \leftrightarrow \forall t{\geq}0\, [x := y(t)]\phi \quad \textit{where } y'(t) = \theta$$

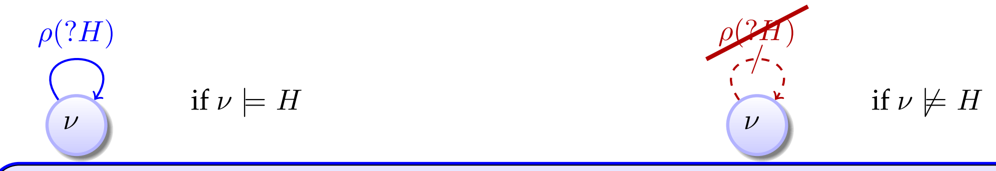
In axiom [′], $y(\cdot)$ is the solution of the symbolic initial-value problem $y'(t) = \theta, y(0) = x$. Solution $y(\cdot)$ is unique since θ is smooth (Lecture 2). Given such a solution $y(\cdot)$, continuous evolution along differential equation $x' = \theta$ can be replaced by a discrete assignment $x := y(t)$ with an additional quantifier for the evolution time t. It goes without saying that variables like t are fresh in [′] and other axioms and proof rules. Notice that conventional initial-value problems are numerical with concrete numbers $x \in \mathbb{R}^n$ as initial values, not symbols x [Wal98]. This would not be enough for our purpose, because we need to consider all states in which the system could start, which may be uncountably many. That is why axiom [′] solves one symbolic initial-value problem, instead, because we could hardly solve uncountable many numerical initial-value problems.

What we have so far about the dynamics of differential equations does not yet help us prove properties of differential equations with evolution domain constraints (a.k.a. continuous programs) $x' = \theta \,\&\, H$. It also does not yet tell us what to do if we cannot solve the differential equation or if the solution is too complicated. We will get to that matter in a much later lecture.

6 Dynamic Axioms for Tests

Recall from Lecture 4 that

$$\rho(?H) = \{(\nu, \nu) \ : \ \nu \models H\}$$



Lemma 5 ([?] soundness)**.** *The test axiom is sound:*

$$([?]) \ [?H]\phi \leftrightarrow (H \to \phi)$$

Tests in $[?H]\phi$ are proven by assuming that the test succeeds with an implication in axiom [?], because test $?H$ can only make a transition when condition H actually holds true. In states where test H fails, no transition is possible and the failed attempt to run the system is discarded. If no transition exists, there is nothing to show for $[\alpha]\phi$ formulas, because their semantics requires ϕ to hold in all states reachable by running α, which is vacuously true if no states are reachable. From left to right, axiom [?] for d$\mathcal{L}$ formula $[?H]\phi$ assumes that formula H holds true (otherwise there is no transition and thus nothing to show) and shows that ϕ holds after the resulting no-op. The converse implication from right to left is by case distinction. Either H is false, then $?H$ cannot make a transition and there is nothing to show. Or H is true, but then also ϕ is true.

7 Dynamic Axioms for Sequential Compositions

For sequential compositions $\alpha; \beta$, Lecture 4 proposed the use of an intermediate condition E characterizing the intermediate states between α and β by way of the following proof rule:

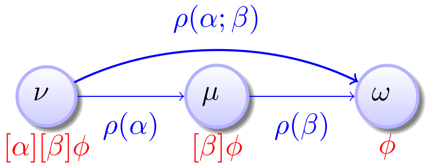
Note 10 (Intermediate conditions as contracts for sequential compositions: proof-rule style)**.** *Intermediate condition contracts for sequential compositions are captured more concisely in the following proof rule:*

$$\text{(R7)} \ \frac{A \to [\alpha]E \quad E \to [\beta]B}{A \to [\alpha; \beta]B}$$

This proof rule is useful, but it has one blatant annoyance compared to R1 or let alone the simplicity and elegance of [∪]. When using proof rule R7 from the desired conclusion to the premises, it does not say how to choose the intermediate condition E. Using R7 successfully requires us to find the right intermediate condition E, for if we don't, the proof won't succeed as we have seen in Lecture 4. That is a bit much if we have to invent a useful intermediate condition E for every single sequential composition.

Fortunately, there is a much better way that we also identify by investigating the dynamical system resulting from $\alpha;\beta$ and its induced Kripke structure. Recall from Lecture 4 that

$$\rho(\alpha;\beta) = \rho(\beta) \circ \rho(\alpha) = \{(\nu,\omega) : (\nu,\mu) \in \rho(\alpha), (\mu,\omega) \in \rho(\beta)\} \tag{10}$$



By its semantics, the dL formula $[\alpha;\beta]\phi$ is true in a state ν iff ϕ is true in all states that $\alpha;\beta$ can reach according to $\rho(\alpha;\beta)$ from ν, i.e. all those states for which $(\nu,\omega) \in \rho(\alpha;\beta)$. Which states are those? And how do they relate to the states reachable by α or by β? They do not relate to those in a way that is as direct as for axiom $[\cup]$. But they still relate, and they do so by way of (10).

Postcondition ϕ has to be true in all states reachable by $\alpha;\beta$ from ν for $[\alpha;\beta]\phi$ to be true at ν. By (10), those are exactly the states ω to which we can get by running β from an intermediate state μ to which we have gotten from ν by running α. Thus, for $[\alpha;\beta]\phi$ to be true at ν it is necessary that ϕ holds in all states ω to which we can get by running β from an intermediate state μ to which we can get by running β from ν. Consequently, $[\alpha;\beta]\phi$ is only true at ν if $[\beta]\phi$ holds in all those intermediate states μ to which we can get from ν by running α. How do we characterize those states? And how can we then express these thoughts in a single logical formula of dL?

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

If we want to express that $[\beta]\phi$ holds in all states μ to which we can get to from ν by running α, then that is exactly what truth of d$\mathcal{L}$ formula $[\alpha][\beta]\phi$ at ν means, because this is the semantics of the modality $[\beta]$.

Consequently,

$$\nu \models [\alpha][\beta]\phi \to [\alpha;\beta]\phi$$

Reexamining our argument backwards, we see that the converse implication also holds

$$\nu \models [\alpha;\beta]\phi \to [\alpha][\beta]\phi$$

The same argument works for all ν, so both implications are even valid.

Lemma 6 ([;] soundness)**.** *The composition axiom is sound:*

$$([;])\ [\alpha;\beta]\phi \leftrightarrow [\alpha][\beta]\phi$$

Proof. Since $\rho(\alpha;\beta) = \rho(\beta) \circ \rho(\alpha)$, we have that $(\nu,\omega) \in \rho(\alpha;\beta)$ iff $(\nu,\mu) \in \rho(\alpha)$ and $(\mu,\omega) \in \rho(\beta)$ for some intermediate state μ. Hence, $\nu \models [\alpha;\beta]\phi$ iff $\mu \models [\beta]\phi$ for all μ with $(\nu,\mu) \in \rho(\alpha)$. That is $\nu \models [\alpha;\beta]\phi$ iff $\nu \models [\alpha][\beta]\phi$. □

Sequential compositions are proven using nested modalities in axiom [;]. From right to left: If, after all α-runs, it is the case that all β-runs lead to states satisfying ϕ (i.e., $[\alpha][\beta]\phi$ holds), then all runs of the sequential composition $\alpha;\beta$ lead to states satisfying ϕ (i.e., $[\alpha;\beta]\phi$ holds). The converse implication uses the fact that if after all α-runs all β-runs lead to ϕ (i.e., $[\alpha][\beta]\phi$), then all runs of $\alpha;\beta$ lead to ϕ (that is, $[\alpha;\beta]\phi$), because the runs of $\alpha;\beta$ are exactly those that first do any α-run, followed by any β-run. Again, it is crucial that d$\mathcal{L}$ is a full logic that considers reachability statements as modal operators, which can be nested, for then both sides in [;] are d$\mathcal{L}$ formulas.

Axiom [;] directly explains sequential composition $\alpha;\beta$ in terms of a structurally simpler formula, one with nested modal operators but simpler hybrid programs. Again, using axiom [;] by reducing occurrences of its left-hand side to its right-hand side decomposes the formula into structurally simpler pieces, thereby making progress. One of the many ways of using axiom [;] is, therefore, captured in the following proof rule:

$$\text{(R9)}\ \frac{A \to [\alpha][\beta]B}{A \to [\alpha;\beta]B}$$

Comparing rule R9 to rule R7, the new rule R9 is much easier to apply, because it does not require us to first provide an intermediate condition E like R7 would. It also does not branch into two premises, which helps keeping the proof lean. Is there a way of reuniting R9 with R7 by using the expressive power of d$\mathcal{L}$?

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

Yes, indeed, there is a very smart choice for the intermediate condition E that makes R7 behave almost as the more efficient R9 would. The clever choice $E \overset{\text{def}}{\equiv} [\beta]B$:

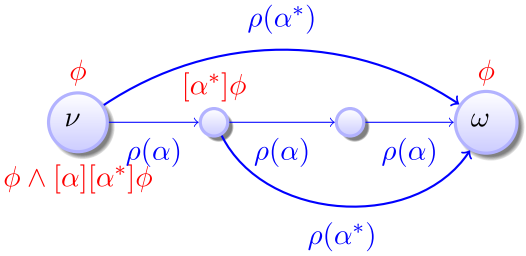
$$\frac{A \to [\alpha][\beta]B \quad [\beta]B \to [\beta]B}{A \to [\alpha;\beta]B}$$

which trivializes the right premise and makes the left premise identical to that of R9.

8 Unwinding Axioms for Loops

Recall from Lecture 4 that

$$\rho(\alpha^*) = \bigcup_{n\in\mathbb{N}} \rho(\alpha^n) \qquad \text{with} \quad \alpha^{n+1} \equiv \alpha^n;\alpha \text{ and } \alpha^0 \equiv ?true$$



Lemma 7 ([*] soundness). *The iteration axiom is sound:*

$$([*])\ [\alpha^*]\phi \leftrightarrow \phi \wedge [\alpha][\alpha^*]\phi$$

Axiom [*] is the iteration axiom, which partially unwinds loops. It uses the fact that ϕ always holds after repeating α (i.e., $[\alpha^*]\phi$), if ϕ holds at the beginning (for ϕ holds after zero repetitions then), and if, after one run of α, ϕ holds after every number of repetitions of α, including zero repetitions (i.e., $[\alpha][\alpha^*]\phi$). So axiom [*] expresses that $[\alpha^*]\phi$ holds iff ϕ holds immediately and after one or more repetitions of α. The same axiom [*] can be used to unwind loops $N \in \mathbb{N}$ times, which corresponds to Bounded Model Checking [CBRZ01]. If the formula is not valid, a bug has been found, otherwise N increases. An obvious issue with this simple approach is that we can never stop increasing N if the formula is actually valid, because we can never find a bug then. A later lecture will discuss proof techniques for repetitions based on invariants that are not subject to this issue. In particular, axiom [*] is characteristically different from the other axioms discussed in this lecture. Unlike the other axioms, [*] does not exactly get rid of the formula on the left-hand side. It just puts it in a different syntactic place, which does not sound like much progress.[2]

[2] With a much more subtle and tricky analysis, it is possible to prove that [*] still makes progress [Pla13]. But this is out of scope for our course.

9 A Proof of a Bouncing Ball

Now that we have understood so many axioms and proof rules, let us use them to prove the (single-hop) bouncing ball (1):

$$\begin{aligned}&0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 > c \geq 0 \rightarrow \\ &\qquad \big[h' = v, v' = -g \,\&\, h \geq 0; (?h = 0; v := -cv \cup ?h \neq 0)\big]\,(0 \leq h \land h \leq H)\end{aligned} \tag{1}$$

Before proceeding, let's modify the hybrid program subtly in tow ways so that there's no more evolution domains, because we have not yet understood how to prove differential equations with evolution domains:

$$\begin{aligned}&0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 > c \geq 0 \rightarrow \\ &\qquad \big[h' = v, v' = -g; (?h = 0; v := -cv \cup ?h \geq 0)\big]\,(0 \leq h \land h \leq H)\end{aligned} \tag{11}$$

To fit things on the page easily, abbreviate

$$\begin{aligned}A_{h,v} &\stackrel{\text{def}}{\equiv} 0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 > c \geq 0\\ B_{h,v} &\stackrel{\text{def}}{\equiv} 0 \leq h \land h \leq H\\ (h'' = -g) &\stackrel{\text{def}}{\equiv} (h' = v, v' = -g)\end{aligned}$$

With these abbreviations, (11) is

$$A_{h,v} \rightarrow [h'' = -g; (?h = 0; v := -cv \cup ?h \geq 0)]B_{h,v}$$

Let there be proof:

$$\begin{array}{ll}
 & A_{h,v} \rightarrow \forall t{\geq}0\,\big((H - \frac{g}{2}t^2 = 0 \rightarrow B_{H-\frac{g}{2}t^2,-c(-gt)}\big) \land (H - \frac{g}{2}t^2 \geq 0 \rightarrow B_{H-\frac{g}{2}t^2,-gt})\big) \\ \hline
[:=] & A_{h,v} \rightarrow \forall t{\geq}0\,[h := H - \frac{g}{2}t^2]\big((h = 0 \rightarrow B_{h,-c(-gt)}) \land (h \geq 0 \rightarrow B_{h,-gt})\big) \\ \hline
[:=] & A_{h,v} \rightarrow \forall t{\geq}0\,[h := H - \frac{g}{2}t^2][v := -gt]\big((h = 0 \rightarrow B_{h,-cv}) \land (h \geq 0 \rightarrow B_{h,v})\big) \\ \hline
[;] & A_{h,v} \rightarrow \forall t{\geq}0\,[h := H - \frac{g}{2}t^2; v := -gt]\big((h = 0 \rightarrow B_{h,-cv,)} \land (h \geq 0 \rightarrow B_{h,v})\big) \\ \hline
['] & A_{h,v} \rightarrow [h'' = -g]\big((h = 0 \rightarrow B_{h,-cv,)} \land (h \geq 0 \rightarrow B_{h,v})\big) \\ \hline
[:=] & A_{h,v} \rightarrow [h'' = -g]\big((h = 0 \rightarrow [v := -cv]B_{h,v}) \land (h \geq 0 \rightarrow B_{h,v})\big) \\ \hline
[?],[?] & A_{h,v} \rightarrow [h'' = -g]\big([?h = 0][v := -cv]B_{h,v} \land [?h \geq 0]B_{h,v}\big) \\ \hline
[;] & A_{h,v} \rightarrow [h'' = -g]\big([?h = 0; v := -cv]B_{h,v} \land [?h \geq 0]B_{h,v}\big) \\ \hline
[\cup] & A_{h,v} \rightarrow [h'' = -g][?h = 0; v := -cv \cup ?h \geq 0]B_{h,v} \\ \hline
[;] & A_{h,v} \rightarrow [h'' = -g; (?h = 0; v := -cv \cup ?h \geq 0)]B_{h,v}
\end{array}$$

Since each of the steps in this proof are justified by using one of the d$\mathcal{L}$ axioms, the conclusion at the very bottom of this derivation is proved if the premise at the very top can be proved. That premise

$$A_{h,v} \rightarrow \forall t{\geq}0\,\big((H - \frac{g}{2}t^2 = 0 \rightarrow B_{H-\frac{g}{2}t^2,-c(-gt)}\big) \land (H - \frac{g}{2}t^2 \geq 0 \rightarrow B_{H-\frac{g}{2}t^2,-gt})\big)$$

expands out to the following formula of first-order real arithmetic by expanding the abbreviations

$$
\begin{aligned}
&0 \le h \land h = H \land v = 0 \land g > 0 \land 1 > c \ge 0 \rightarrow \\
&\qquad \forall t{\ge}0 \Big((H - \frac{g}{2}t^2 = 0 \rightarrow 0 \le H - \frac{g}{2}t^2 \land H - \frac{g}{2}t^2 \le H) \\
&\qquad\qquad \land (H - \frac{g}{2}t^2 \ge 0 \rightarrow 0 \le H - \frac{g}{2}t^2 \land H - \frac{g}{2}t^2 \le H) \Big)
\end{aligned}
$$

In this case, this remaining premise can be easily seen to be valid. The first assumption $H - \frac{g}{2}t^2 = 0 \rightarrow \dots$ in the middle line directly implies the first conjunct of its right-hand side

$$0 \le H - \frac{g}{2}t^2 \land H - \frac{g}{2}t^2 \le H$$

and reduces the second conjunct to $0 \le H$, which the assumption in the first line assumed ($0 \le h = H$). Similarly, the first assumption $H - \frac{g}{2}t^2 \ge 0$ of the last line implies the first conjunct of its right-hand side

$$0 \le H - \frac{g}{2}t^2 \land H - \frac{g}{2}t^2 \le H$$

and the second conjunct holds by assumption $g > 0$ from the first line and the real arithmetic fact that $t^2 \ge 0$.

How first-order logic and first-order real arithmetic formulas such as this one can be proved in general, however, is an interesting topic for a later lecture. For now, we are happy to report that we have just formally verified our very first CPS. Exciting! We have found a proof of (11).

Okay, admittedly, the CPS we just verified was only a bouncing ball. And all we know about it now is that it won't fall through the cracks in the ground nor jump high up to the moon. But most big steps for mankind start with a small step by someone.

Yet, before we get too carried away, we first need to remember that (11) is just a single-hop bouncing ball. So there's still an argument to be made about what happens if the bouncing ball repeats. And a rather crucial argument too, because bouncing balls let loose in the air tend not to jump any higher without hitting the ground first, which is where the model (11) stops prematurely, because it is missing a repetition. So let's put worrying about loops on the agenda for an upcoming lecture.

Yet, there's one more issue with the proof for the bouncing ball that we derived. It works in a somewhat undisciplined chaotic way, by using d$\mathcal{L}$ axioms all over the place. This liberal proof style can be useful for manual proofs and creative shortcuts. Albeit, since the d$\mathcal{L}$ axioms are sound, even such a liberal proof is a proof. But liberal proofs are also somewhat unfocused and non-systematic, which makes them unreasonable for automation purposes and also tends to get people lost if the problems at hand are more complex than the single-hop bouncing ball. That is the reason why we will investigate more focused, more systematic, and more algorithmic proofs next.

10 Summary

The differential dynamic logic axioms that we have seen in this lecture are summarized in Fig. 2. There are further axioms and proof rules of differential dynamic logic that later lectures will examine [Pla12c, Pla12a].

Note 13. *The following axioms of* dL *are sound:*

[:=] $[x := \theta]\phi(x) \leftrightarrow \phi(\theta)$

[?] $[?H]\phi \leftrightarrow (H \rightarrow \phi)$

['] $[x' = \theta]\phi \leftrightarrow \forall t{\geq}0\, [x := y(t)]\phi \qquad (y'(t) = \theta)$

[∪] $[\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \land [\beta]\phi$

[;] $[\alpha; \beta]\phi \leftrightarrow [\alpha][\beta]\phi$

[*] $[\alpha^*]\phi \leftrightarrow \phi \land [\alpha][\alpha^*]\phi$

Figure 2: Summary of differential dynamic logic axioms from this lecture

Exercises

Exercise 1. Explain why the subtle transformation from (1) to (11) was okay in this case.

Exercise 2. Identify which of the assumptions of (11) are actually required for the proof of (11). Which formulas could we have dropped from $0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 > c \geq 0$ and still be able to prove

$$0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 > c \geq 0 \rightarrow [h'' = -g; (?h = 0; v := -cv \cup ?h \geq 0)]0 \leq h \land h \leq H$$

Exercise 3. Develop an axiom for differential equations with evolution domains in a style that is similar to [']. That is, develop an axiom for $[x' = \theta \,\&\, H]\phi$. As in ['], you can assume to have a unique solution for the corresponding symbolic initial-value problem.

Exercise 4. All axioms need to be proved to be sound. These lecture notes only did a proper proof for [;]. Turn the informal arguments for the other axioms into proper soundness proofs using the semantics of dL formulas.

Exercise 5. Would the following be a useful replacement for the [*] axiom?

$$[\alpha^*]\phi \leftrightarrow \phi \land [\alpha^*]\phi$$

References

[CBRZ01] Edmund M. Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Form. Methods Syst. Des.*, 19(1):7–34, 2001.

[DBL12] *Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25–28, 2012*. IEEE, 2012.

[Hoa69] Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.

[PC08] André Platzer and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. In Aarti Gupta and Sharad Malik, editors, *CAV*, volume 5123 of *LNCS*, pages 176–189. Springer, 2008. doi:10.1007/978-3-540-70545-1_17.

[PC09] André Platzer and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. *Form. Methods Syst. Des.*, 35(1):98–120, 2009. Special issue for selected papers from CAV'08. doi:10.1007/s10703-009-0079-8.

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

[Pla10] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14509-4.

[Pla12a] André Platzer. The complete proof theory of hybrid systems. In *LICS* [DBL12], pages 541–550. doi:10.1109/LICS.2012.64.

[Pla12b] André Platzer. Dynamic logics of dynamical systems. *CoRR*, abs/1205.4788, 2012. arXiv:1205.4788.

[Pla12c] André Platzer. Logics of dynamical systems. In *LICS* [DBL12], pages 13–24. doi:10.1109/LICS.2012.13.

[Pla13] André Platzer. A complete axiomatization of differential game logic for hybrid games. Technical Report CMU-CS-13-100R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January, Revised and extended in July 2013.

[SKSC98] Danbing Seto, Bruce Krogh, Lui Sha, and Alongkrit Chutinan. The Simplex architecture for safe online control system upgrades. In *ACC*, volume 6, pages 3504–3508, 1998.

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

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Truth & Proof

André Platzer

Carnegie Mellon University

Lecture 6

1 Introduction

[1] Lecture 5 investigated dynamic axioms for dynamical systems, i.e. axioms in differential dynamic logic ($\mathsf{d}\mathcal{L}$) that characterize operators of the dynamical systems that $\mathsf{d}\mathcal{L}$ describes by hybrid programs in terms of structurally simpler $\mathsf{d}\mathcal{L}$ formulas. That lecture did not show all important axioms yet, but still showed enough to prove a property of a bouncing ball. Yet, there's more to proofs than just axioms. Proofs also have proof rules for combining fragments of arguments into a bigger proof by proof steps.

Recall that our proof about the (single-hop) bouncing ball still suffered from at least two issues. It was a sound proof and an interesting proof. But the way we had come up with the proof was somewhat undisciplined, because we just applied axioms seemingly at random at all kinds of places all over the logical formulas. After we see such a proof, that is not a concern. But better structuring would help us find proofs more constructively. The second issue was that the axioms for the dynamics that Lecture 5 showed us did not actually help in proving the propositional logic and arithmetic parts.

The lecture today addresses both issues by imposing more structure on proofs and, as part of that, handle the operators of first-order logic that differential dynamic logic inherits (propositional connectives such as $\wedge, \vee, \rightarrow$) and quantifiers $\forall, \exists$). As part of the structuring, we will make ample and crucial use of the dynamic axioms from Lecture 5. Yet, they will be used in a more structured way than so far.

These notes are based on [Pla08, Pla10, Chapter 2.5.2], where more information can be found in addition to more information in [Pla10, Appendix A]. Sequent calculus is

[1] By both sheer coincidence and by higher reason, the title of this lecture turns out to be closely related to the subtitle of a well-known book on mathematical logic [And02], which summarizes the philosophy we pursue here in a way that is impossible to improve upon any further: *To truth through proof.*

discussed in more detail also in the handbook of proof theory [Bus98]. More resources and background material on first-order logic is also listed on the course web page.

2 Truth and Proof

Truth is defined by the semantics of logical formulas. The semantics gives a mathematical meaning to formulas that, in theory, could be used to establish truth of a logical formula. In practice, this is usually less feasible, for one thing, because quantifiers of differential dynamic logic quantify over real numbers (after all their variables may represent real quantities like velocities and positions). Yet, there are infinitely many of those, so determining the truth value of a universally quantified logical formula directly by working with its semantics is challenging since that'd require instantiating it with infinitely many real numbers. The same matter is even more difficult for the hybrid dynamics involved in modalities of differential dynamic logic formulas, because hybrid systems have so many possible behaviors.

Yet, we are still interested in establishing whether a logical formula is true. Or, actually, whether the formula is valid, since truth of a logical formula depends on the state (cf. definition of $\nu \models \phi$ in Lecture 4) whereas validity of a logical formula is independent of the state (cf. definition of $\vDash \phi$), because validity means truth in all states.

The validity of logical formulas can be established by other means, namely by producing a proof of that formula. Like the formula itself, but unlike its semantics, a proof is a syntactical object that is amenable, e.g., to representation and manipulation in a computer. This finite syntactical argument represented in a proof witnesses validity of a logical formula. Proofs can be produced in a machine. They can be stored to be recalled as witnesses and evidence for the validity of their conclusion. And they can be checked by humans or machines for correctness. They can also be inspected for analytic insights about the reasons for the validity of a formula, which goes beyond the factual statement of validity. A proof justifies the judgment that a logical formula is valid, which, without such a proof as evidence, is no more than an empty claim.

Truth and proof should be related intimately, because we would only want to accept proofs that imply truth, i.e. proofs that imply their consequences to be valid if their premises are. That is, proof systems should be sound to be reliable. The converse question is that of completeness, whether all true formulas (again in the sense of valid) can be proved, which turns out to be much more subtle.

3 Sequents

Sequent calculus was originally developed by Gerhard Gentzen [Gen35] for studying properties of natural deduction calculi. Sequent calculus has been used very successfully for numerous other purposes since.

Sequents are essentially a standard form for logical formulas that is convenient for proving purposes.

Note 1. *A* sequent *is of the form* $\Gamma \vdash \Delta$*, where the* antecedent Γ *and* succedent Δ *are finite sets of formulas. The semantics of* $\Gamma \vdash \Delta$ *is that of the formula* $\bigwedge_{\phi\in\Gamma} \phi \rightarrow \bigvee_{\psi\in\Delta} \psi$*.*

For quantifier elimination rules, we will later make use of this fact by considering sequent $\Gamma \vdash \Delta$ as an abbreviation for the latter formula. Empty conjunctions are equivalent to *true*. Empty disjunctions are equivalent to *false*. Hence, the sequent $\vdash A$ means the same as the formula A. The empty sequent $\vdash$ means the same as the formula *false*.

The antecedent Γ can be thought of as the formulas we assume to be true, whereas the succedent Δ can be understood as formulas for which we want to show that at least one of them is true assuming all formulas of Γ are true. So for proving a sequent $\Gamma \vdash \Delta$, we assume all Γ and want to show that one of the Δ is true. For some simple sequents like $\Gamma, \phi \vdash \phi, \Delta$, we directly know that they are valid, because we can certainly show ϕ if we assume ϕ (in fact, we will use this as an axiom). For other sequents, it is more difficult to see whether they are valid (true under all circumstances) and it is the purpose of a proof calculus to provide a means to find out.

The antecedent and succedent of a sequent are considered as sets. So the order of formulas is irrelevant, so we implicitly adopt what is called the *exchange rule* and do not distinguish between the following two sequents

$$\Gamma, A, B \vdash \Delta \qquad \text{and} \qquad \Gamma, B, A \vdash \Delta$$

nor do we distinguish between

$$\Gamma \vdash C, D, \Delta \qquad \text{and} \qquad \Gamma \vdash D, C, \Delta$$

Antecedent and succedent are considered as sets, not multisets, so we implicitly adopt what is called the *contraction rule* and do not distinguish between the following two sequents

$$\Gamma, A, A \vdash \Delta \qquad \text{and} \qquad \Gamma, A \vdash \Delta$$

nor do we distinguish between

$$\Gamma \vdash C, C, \Delta \qquad \text{and} \qquad \Gamma \vdash C, \Delta$$

The only structural rule of sequent calculus that we will find reason to use explicitly in practice is the *weakening* proof rule (alias *hiding* proof rule) that can be used to remove or hide formulas from the antecedent (Wl) or succedent (Wr), respectively:

$$\text{(Wr)}\ \frac{\Gamma \vdash \Delta}{\Gamma \vdash \phi, \Delta}$$

$$\text{(Wl)}\ \frac{\Gamma \vdash \Delta}{\Gamma, \phi \vdash \Delta}$$

Weakening rules are sound, since it is fine in structural logics to prove a sequent with more formulas in the antecedent or succedent by a proof that uses only some of those formulas. This is different in substructural logics such as linear logic.

$$
(\neg r)\ \frac{\Gamma, \phi \vdash \Delta}{\Gamma \vdash \neg\phi, \Delta}
\qquad
(\vee r)\ \frac{\Gamma \vdash \phi, \psi, \Delta}{\Gamma \vdash \phi \vee \psi, \Delta}
\qquad
(\wedge r)\ \frac{\Gamma \vdash \phi, \Delta \quad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \phi \wedge \psi, \Delta}
$$

$$
(\neg l)\ \frac{\Gamma \vdash \phi, \Delta}{\Gamma, \neg\phi \vdash \Delta}
\qquad
(\vee l)\ \frac{\Gamma, \phi \vdash \Delta \quad \Gamma, \psi \vdash \Delta}{\Gamma, \phi \vee \psi \vdash \Delta}
\qquad
(\wedge l)\ \frac{\Gamma, \phi, \psi \vdash \Delta}{\Gamma, \phi \wedge \psi \vdash \Delta}
$$

$$
(\rightarrow r)\ \frac{\Gamma, \phi \vdash \psi, \Delta}{\Gamma \vdash \phi \rightarrow \psi, \Delta}
\qquad
(ax)\ \frac{}{\Gamma, \phi \vdash \phi, \Delta}
$$

$$
(\rightarrow l)\ \frac{\Gamma \vdash \phi, \Delta \quad \Gamma, \psi \vdash \Delta}{\Gamma, \phi \rightarrow \psi \vdash \Delta}
\qquad
(cut)\ \frac{\Gamma \vdash \phi, \Delta \quad \Gamma, \phi \vdash \Delta}{\Gamma \vdash \Delta}
$$

Figure 1: Propositional proof rules of sequent calculus

4 Propositional Proof Rules

For propositional logic, standard propositional rules $\neg$r–*cut* with the cut rule are listed in Fig. 1. They decompose the propositional structure of formulas. Rules $\neg$r and $\neg$l use simple dualities caused by the implicative semantics of sequents. Essentially, instead of showing $\neg\phi$ in the succedent, we assume the contrary ϕ in the antecedent with rule $\neg$r. In rule $\neg$l, instead of assuming $\neg\phi$ in the antecedent, we show the contrary ϕ in the succedent. Rule $\vee$r uses the fact that formulas are combined disjunctively in succedents, rule $\wedge$l that they are conjunctive in antecedents. The comma between formulas in an antecedent has the same effect as a conjunction, and the comma between formulas in the succedent has the same effect as a disjunction. Rules $\vee$l and $\wedge$r split the proof into two cases, because conjuncts in the succedent can be proven separately ($\wedge$r) and, dually, disjuncts of the antecedent can be assumed separately ($\vee$l). For $\wedge$r we want to show conjunction $\phi \wedge \psi$, so in the left branch we proceed to show $\Gamma \vdash \phi, \Delta$ and, in addition, in the right branch we show $\Gamma \vdash \psi, \Delta$, which, together, entail $\Gamma \vdash \phi \wedge \psi, \Delta$. If, as in rule $\vee$l, we assume disjunction $\phi \vee \psi$ as part of the antecedent, then we do not know if we can assume ϕ to hold or if we can assume ψ to hold in the antecedent, but know only that one of them holds. Hence, as in a case distinction, $\vee$l considers both cases, the case where we assume ϕ in the antecedent, and the case where we assume ψ. If both subgoals can be proven, this entails $\Gamma, \phi \vee \psi \vdash \Delta$. Rules $\rightarrow$r and $\rightarrow$l can be derived from the equivalence of $\phi \rightarrow \psi$ and $\neg\phi \vee \psi$. Rule $\rightarrow$r uses the fact that implication $\rightarrow$ has the same meaning as the sequent arrow $\vdash$ of a sequent. Intuitively, to show implication $\phi \rightarrow \psi$, rule $\rightarrow$r assumes ϕ (in the antecedent) and shows ψ (in the succedent). Rule $\rightarrow$l assumes an implication $\phi \rightarrow \psi$ to hold in the antecedent, but we do not know if this implication holds because ϕ is false, or because ψ is true, so $\rightarrow$l splits into those two branches.

The axiom rule *ax* closes a goal (there are no further subgoals, which we sometimes mark $*$ explicitly), because assumption ϕ in the antecedent trivially entails ϕ in the succedent (sequent $\Gamma, \phi \vdash \phi, \Delta$ is a simple syntactic tautology).

Rule *cut* is the *cut* rule that can be used for case distinctions: The right subgoal assumes any additional formula ϕ in the antecedent that the left subgoal shows in the succedent. Dually: regardless of whether ϕ is actually true or false, both cases are covered by proof branches. We only use cuts in an orderly fashion to derive simple rule dualities and to simplify meta-proofs. In practical applications, cuts are not needed in theory. But in practice, complex practical applications make use of cuts for efficiency reasons. Cuts an be used, for example, to simplify arithmetic.

Even though we write sequent rules as if the principal formula (like $\phi \wedge \psi$ in $\wedge$r,$\wedge$l) were at the end of the antecedent or at the beginning of the succedent, respectively, the sequent proof rules can be applied to other formulas in the antecedent or succedent, respectively, because we consider their order to be irrelevant.

5 Proofs

The d$\mathcal{L}$ calculus has further proof rules. But before investigating those, let us first understand already what a proof is and what it means to prove a logical formula. The same notion of proof and provability works for propositional logic as it does for differential dynamic logic, except that the latter has more proof rules.[2]

A formula ϕ is provable or derivable (in the d$\mathcal{L}$ calculus) if we can find a d$\mathcal{L}$ proof for it that starts with axioms (rule ax) at the leaves and ends with a sequent $\vdash \phi$ at the bottom and that has only used d$\mathcal{L}$ proof rules in between. While constructing proofs, however, we would start with the desired goal $\vdash \phi$ at the bottom and work our way backwards to the subgoals until they can be proven to be valid as axioms (ax). Once all subgoals have been proven to be valid axioms, they entail their consequences, which, recursively, entail the original goal $\vdash \phi$. This property of preserving truth or preserving entailment is called soundness. Thus, while constructing proofs, we work bottom-up from the goal. When we have found a proof, we justify formulas from the axioms top-down to the original goal.

We write $\vdash_{\text{d}\mathcal{L}} \phi$ iff d$\mathcal{L}$ formula ϕ can be *proved* with d$\mathcal{L}$ rules from d$\mathcal{L}$ axioms. That is, a d$\mathcal{L}$ formula is inductively defined to be *provable* in the d$\mathcal{L}$ sequent calculus if it is the conclusion (below the rule bar) of an instance of one of the d$\mathcal{L}$ sequent proof rules, whose premises (above the rule bar) are all provable. A formula ψ is *provable* from a set Φ of formulas, denoted by $\Phi \vdash_{\text{d}\mathcal{L}} \psi$, iff there is a finite subset $\Phi_0 \subseteq \Phi$ for which the sequent $\Phi_0 \vdash \psi$ is provable.

Example 1. A very simple (in fact propositional) proof of the formula

$$v^2 \leq 10 \wedge b > 0 \rightarrow b > 0 \wedge (\neg(v \geq 0) \vee v^2 \leq 10) \tag{1}$$

is shown in Fig. 2. The proof starts with the proof goal as a sequent at the bottom:

$$\vdash v^2 \leq 10 \wedge b > 0 \rightarrow b > 0 \wedge (\neg(v \geq 0) \vee v^2 \leq 10).$$

and proceeds applying proof rules upwards.

The first (i.e., bottom most) proof step applies proof rule $\rightarrow$r to turn the implication ($\rightarrow$) to the sequent level by moving the assumption into the antecedent. The next

[2] There is one subtlety with quantifier elimination that

$$\land r\,\dfrac{\land l\,\dfrac{ax\,\dfrac{*}{v^2 \leq 10, b > 0 \vdash b > 0}}{v^2 \leq 10 \land b > 0 \vdash b > 0} \qquad \lor r\,\dfrac{\land l\,\dfrac{ax\,\dfrac{*}{v^2 \leq 10, b > 0 \vdash \neg(v \geq 0), v^2 \leq 10}}{v^2 \leq 10 \land b > 0 \vdash \neg(v \geq 0), v^2 \leq 10}}{v^2 \leq 10 \land b > 0 \vdash \neg(v \geq 0) \lor v^2 \leq 10}}{\rightarrow r\,\dfrac{v^2 \leq 10 \land b > 0 \vdash b > 0 \land (\neg(v \geq 0) \lor v^2 \leq 10)}{\vdash v^2 \leq 10 \land b > 0 \rightarrow b > 0 \land (\neg(v \geq 0) \lor v^2 \leq 10)}}$$

Figure 2: Simple propositional example proof

proof step applies rule $\land$r to split the proof into the left branch for showing that conjunct $b > 0$ follows from the assumptions in the antecedent and into the right branch for showing that conjunct $\neg(v \geq 0) \lor v^2 \leq 10$ follows from the antecedent also. On the left branch, the proof closes with an axiom ax after splitting the conjunction $\land$ on the antecedent with rule $\land$l. We mark closed proof goals with $*$, just to indicate that we did not just stopped writing. The right branch closes with an axiom ax after splitting the disjunction ($\lor$) in the succedent with rule $\lor$r and then splitting the conjunction ($\land$) in the antecedent with rule $\land$l. Now that all branches of the proof have closed (with ax), we know that all leaves at the top are valid, and, hence, since the premises are valid, each application of a proof rule ensures that their respective conclusions are valid also. By recursively following this derivation from the leaves at the top to the original root at the bottom, we see that the original goal is valid and formula (1) is, indeed, true under all circumstances (valid).

While this proof does not show anything particularly exciting, because it only uses propositional rules, it shows how a proof can be built systematically in the d$\mathcal{L}$ calculus and gives an intuition as to how validity is inherited from the premises to the conclusions.

6 Dynamic Proof Rules

Lecture 5 has shown axioms for dynamical systems that correspond to the operators of hybrid programs in $[\cdot]$ modalities of differential dynamic logic [Pla12]. These were equivalence axioms which represent schemata of valid formulas such as

$$([\cup])\ [\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \land [\beta]\phi$$

How can such valid equivalences be used in the context of a sequent calculus? There is more than one productive way to do that.

The d$\mathcal{L}$ axioms such as $[\cup]$ are primarily meant to be used for replacing the left-hand side $[\alpha \cup \beta]\phi$ by the structurally simpler right-hand side $[\alpha]\phi \land [\beta]\phi$, because that direction of use assigns meaning to $[\alpha \cup \beta]\phi$ in logically simpler terms, i.e. as a structurally simpler logical formula. The following two sequent proof rules allow replacements in that direction for formulas in the antecedent ($[\cup]l$) and succedent ($[\cup]r$), respectively.

$$([\cup]r)\ \frac{\Gamma \vdash [\alpha]\phi \land [\beta]\phi, \Delta}{\Gamma \vdash [\alpha \cup \beta]\phi, \Delta}$$

$$([\cup]l)\ \frac{\Gamma, [\alpha]\phi \land [\beta]\phi \vdash \Delta}{\Gamma, [\alpha \cup \beta]\phi \vdash \Delta}$$

The sequent proof rules $[\cup]r$,$[\cup]l$ are more systematic in that they orient the use of the axiom $[\cup]$ in the direction that makes formulas structurally simpler. Without such direction, proofs could apply axiom $[\cup]$ from left to right and then from right to left and from left to right again forever without making any progress. That does not happen with $[\cup]r$,$[\cup]l$, because they cannot simply go back.[3] Furthermore, the sequent rules $[\cup]r$,$[\cup]l$ focus the application of axiom $[\cup]$ to the top level of sequents. That is, $[\cup]r$,$[\cup]l$ can only be used for formulas of the succedent or antecedent, respectively, that are of the form $[\alpha \cup \beta]\phi$, not to any subformulas within that happen to be of this form. Abiding both of those restrictions imposes more structure on the proof, compared to the proof we produced in Lecture 5.

Reconsidering the contract-type rules from Lecture 4, we could have turned $[\cup]$ into the following two sequent proof rules instead of into $[\cup]r$,$[\cup]l$:

$$(\text{R14})\ \frac{\Gamma \vdash [\alpha]\phi, \Delta \quad \Gamma \vdash [\beta]\phi, \Delta}{\Gamma \vdash [\alpha \cup \beta]\phi, \Delta}$$

$$(\text{R15})\ \frac{\Gamma, [\alpha]\phi, [\beta]\phi \vdash \Delta}{\Gamma, [\alpha \cup \beta]\phi \vdash \Delta}$$

These rules R14,R15 already split into separate subgoals (R14) or separate formulas (R15), respectively. It would be fine to use sequent rules R14,R15 instead of $[\cup]r$,$[\cup]l$, and, in fact, earlier versions of KeYmaera did. The disadvantage of rules R14,R15 compared to $[\cup]r$,$[\cup]l$ is that rules R14,R15 have a less obvious relation to axiom $[\cup]$ and that they are asymmetric (they both look surprisingly different). This nuisance is overcome in $[\cup]r$,$[\cup]l$, from which rules R14,R15 follow immediately with just one more application of rules $\land$r or $\land$l, respectively. Thus, $[\cup]r$,$[\cup]l$ are more elementary and more atomic in that they isolate the proof-theoretical meaning of $[\alpha \cup \beta]\phi$, as opposed to already incorporating parts of the meaning of $\land$ as well, which is what propositional rules $\land$r,$\land$l are supposed to capture.

The other d$\mathcal{L}$ axioms from Lecture 5 translate into sequent calculus proof rules in the same way. The dynamic modality rules transform a hybrid program into structurally simpler logical formulas by symbolic decomposition.

For Fig. 3, we adopt a convention to simplify notation. Instead of rules $[\cup]r$,$[\cup]l$, Fig. 3 shows a single *symmetric rule* $[\cup]$ that does not mention the sequent sign $\vdash$:

$$\frac{[\alpha]\phi \land [\beta]\phi}{[\alpha \cup \beta]\phi}$$

This is abbreviated notation to say that the same rule from a conclusion with a formula $[\alpha \cup \beta]$ in either antecedent or succedent can be proved from a premise with formula

[3] Albeit, going back is still possible indirectly when using a reasonably creative *cut*. But that requires an intentional extra effort to do so.

$[\alpha]\phi \land [\beta]\phi$ in the antecedent or succedent, respectively. That is, we consider the symmetric rule [∪] as an abbreviation for the two rules [∪]r,[∪]l. Fig. 3 lists a single symmetric rule [∪] but we pretend it had both rules [∪]r,[∪]l. The same applies to the other symmetric rules in Fig. 3, which each have a version of the rule for the antecedent and a version of the rule for the succedent. The antecedent version of [;] is called [;]l, its succedent version is called [;]r. The antecedent version of [′] is called [′]l, its succedent version is called [′]r and so on.

$$
\begin{array}{llllll}
(\langle;\rangle) & \dfrac{\langle\alpha\rangle\langle\beta\rangle\phi}{\langle\alpha;\beta\rangle\phi} &
(\langle{*^n}\rangle) & \dfrac{\phi \lor \langle\alpha\rangle\langle\alpha^*\rangle\phi}{\langle\alpha^*\rangle\phi} &
(\langle{:=}\rangle) & \dfrac{\phi_x^\theta}{\langle x := \theta\rangle\phi} \\[2ex]
([;]) & \dfrac{[\alpha][\beta]\phi}{[\alpha;\beta]\phi} &
([{*^n}]) & \dfrac{\phi \land [\alpha][\alpha^*]\phi}{[\alpha^*]\phi} &
([{:=}]) & \dfrac{\phi_x^\theta}{[x := \theta]\phi} \\[2ex]
(\langle\cup\rangle) & \dfrac{\langle\alpha\rangle\phi \lor \langle\beta\rangle\phi}{\langle\alpha\cup\beta\rangle\phi} &
(\langle?\rangle) & \dfrac{H \land \psi}{\langle ?H\rangle\psi} &
(\langle{'}\rangle) & \dfrac{\exists t{\geq}0\, \big((\forall 0{\leq}\tilde{t}{\leq}t\, \langle x := y(\tilde{t})\rangle H) \land \langle x := y(t)\rangle\phi\big)}{\langle x' = \theta \,\&\, H\rangle\phi}\,{}^{1} \\[2ex]
([\cup]) & \dfrac{[\alpha]\phi \land [\beta]\phi}{[\alpha\cup\beta]\phi} &
([?]) & \dfrac{H \rightarrow \psi}{[?H]\psi} &
([{'}]) & \dfrac{\forall t{\geq}0\, \big((\forall 0{\leq}\tilde{t}{\leq}t\, [x := y(\tilde{t})] H) \rightarrow [x := y(t)]\phi\big)}{[x' = \theta \,\&\, H]\phi}\,{}^{1}
\end{array}
$$

[1] t and $\tilde{t}$ are fresh logical variables and $\langle x := y(t)\rangle$ is the discrete assignment belonging to the solution y of the differential equation with constant symbol x as symbolic initial value.

Figure 3: Dynamic proof rules of d$\mathcal{L}$ sequent calculus

Nondeterministic choices split into their alternatives (⟨∪⟩,[∪]). For rule [∪]: If all α transitions lead to states satisfying ϕ (i.e., $[\alpha]\phi$ holds) and all β transitions lead to states satisfying ϕ (i.e., $[\beta]\phi$ holds), then, all transitions of program $\alpha \cup \beta$ that choose between following α and following β also lead to states satisfying ϕ (i.e., $[\alpha \cup \beta]\phi$ holds). Dually for rule ⟨∪⟩, if there is an α transition to a ϕ state ($\langle\alpha\rangle\phi$) or a β-transition to a ϕ state ($\langle\beta\rangle\phi$), then, in either case, there is a transition of $\alpha \cup \beta$ to ϕ ($\langle\alpha \cup \beta\rangle\phi$ holds), because $\alpha \cup \beta$ can choose which of those transitions to follow. A general principle behind the d$\mathcal{L}$ proof rules that is most noticeable in ⟨∪⟩,[∪] is that these proof rules symbolically decompose the reasoning into two separate parts and analyse the fragments α and β separately, which is good for scalability. For these symbolic structural decompositions, it is very helpful that d$\mathcal{L}$ is a full logic that is closed under all logical operators, including disjunction and conjunction, for then the premises in [∪],⟨∪⟩ are d$\mathcal{L}$ formulas again (unlike in Hoare logic [Hoa69]).

Sequential compositions are proven using nested modalities (⟨;⟩,[;]). For rule [;]: If after all α-transitions, all β-transitions lead to states satisfying ϕ (i.e., $[\alpha][\beta]\phi$ holds), then also all transitions of the sequential composition $\alpha;\beta$ lead to states satisfying ϕ (i.e., $[\alpha;\beta]\phi$ holds). The dual rule ⟨;⟩ uses the fact that if there is an α-transition, after which there is a β-transition leading to ϕ (i.e., $\langle\alpha\rangle\langle\beta\rangle\phi$), then there is a transition of $\alpha;\beta$ leading to ϕ (that is, $\langle\alpha;\beta\rangle\phi$), because the transitions of $\alpha;\beta$ are just those that first do any α-transition, followed by any β-transition.

Rules $\langle *^n\rangle$,$[*^n]$ are the usual iteration rules, which partially unwind loops. Rule $\langle *^n\rangle$

uses the fact that ϕ holds after repeating α (i.e., $\langle\alpha^*\rangle\phi$), if ϕ holds at the beginning (for ϕ holds after zero repetitions then), or if, after one execution of α, ϕ holds after any number of repetitions of α, including zero repetitions (i.e., $\langle\alpha\rangle\langle\alpha^*\rangle\phi$). So rule $\langle *^n\rangle$ expresses that for $\langle\alpha^*\rangle\phi$ to hold, ϕ must hold either immediately or after one or more repetitions of α. Rule $[*^n]$ is the dual rule expressing that ϕ must hold after all of those combinations for $[\alpha^*]\phi$ to hold.

Tests are proven by showing (with a conjunction in rule $\langle ?\rangle$) or assuming (with an implication in rule $[?]$) that the test succeeds, because test $?H$ can only make a transition when condition H actually holds true. Thus, for d$\mathcal{L}$ formula $\langle ?H\rangle\phi$, rule $\langle ?\rangle$ is used to prove that H holds true (otherwise there is no transition and thus the reachability property is false) and that ϕ holds after the resulting no-op. Rule $[?]$ for d$\mathcal{L}$ formula $[?H]\phi$, in contrast, assumes that H holds true (otherwise there is no transition and thus nothing to show) and shows that ϕ holds after the resulting no-op.

Given first-order definable flows for their differential equations, proof rules $\langle'\rangle$,$[']$ handle continuous evolutions. These flows are combined in the discrete jump set $x := y(t)$. Given a solution $x := y(t)$ for the differential equation system with symbolic initial values $x_1, \dots, x_n$, continuous evolution along differential equations can be replaced by a discrete jump $\langle x := y(t)\rangle$ with an additional quantifier for the evolution time t. The effect of the constraint on H is to restrict the continuous evolution such that its solution $x := y(\tilde{t})$ remains in the evolution domain H at all intermediate times $\tilde{t} \leq t$. This constraint simplifies to $true$ if the evolution domain restriction H is $true$, which makes sense, because there are no special constraints on the evolution (other than the differential equations) if the evolution domain region is described by $true$, hence the full space $\mathbb{R}^n$. A notable special case of rules $[']$ and $\langle'\rangle$ is when the evolution domain H is $true$:

$$\frac{\forall t{\geq}0\, \langle x := y(t)\rangle\phi}{[x_1' = \theta_1, .., x_n' = \theta_n]\phi} \qquad \frac{\exists t{\geq}0\, \langle x := y(t)\rangle\phi}{\langle x_1' = \theta_1, .., x_n' = \theta_n\rangle\phi} \tag{2}$$

7 Quantifier Proof Rules

$$(\exists r)\ \frac{\Gamma \vdash \phi(\theta), \exists x\, \phi(x), \Delta}{\Gamma \vdash \exists x\, \phi(x), \Delta}\,{}^{1} \qquad (\forall r)\ \frac{\Gamma \vdash \phi(s(X_1, .., X_n)), \Delta}{\Gamma \vdash \forall x\, \phi(x), \Delta}\,{}^{2}$$

$$(\forall l)\ \frac{\Gamma, \phi(\theta), \forall x\, \phi(x) \vdash \Delta}{\Gamma, \forall x\, \phi(x) \vdash \Delta}\,{}^{1} \qquad (\exists l)\ \frac{\Gamma, \phi(s(X_1, .., X_n)) \vdash \Delta}{\Gamma, \exists x\, \phi(x) \vdash \Delta}\,{}^{2}$$

[1] θ is an arbitrary term, often a new (existential) logical variable X.
[2] s is a new (Skolem) function and $X_1, .., X_n$ are all (existential) free logical variables of $\forall x\, \phi(x)$.

Figure 4: Proof rules for first-order quantifiers

Rules ∃r,∀l,∀r,∃l are standard proof rules for first-order logic. For explaining these quantifier proof rules, let us first assume for a moment there are no (existential) free variables $X_1, \dots, X_n$ (i.e. $n = 0$) and use what is known as the ground calculus.

The quantifier proof rules work much as in mathematics. Consider $\forall$r, where we want to show a universally quantified property. When a mathematician wants to show a universally quantified property $\forall x\, \phi(x)$ to hold, he could choose a fresh symbol s (called Skolem function symbol) and prove that $\phi(s)$ holds (for s). Then the mathematician would remember that s was arbitrary and his proof did not assume anything special about the value of s. So he would conclude that $\phi(s)$ must indeed hold for all s, and that hence $\forall x\, \phi(x)$ holds true. For example, to show that the square of all numbers is nonnegative, a mathematician could start out by saying "let s be an arbitrary number", prove $s^2 \geq 0$ for s, and then conclude $\forall x\, (x^2 \geq 0)$, since s was arbitrary. Proof rule $\forall$r essentially makes this reasoning formal. It chooses a *new* (function) symbol s and replaces the universally quantified formula in the succedent by a formula for s (with all free logical variables $X_1, \dots, X_n$ added as arguments, as we explain below). Notice, of course, that it is important to choose a new symbol s that has not been used (in the sequent) before. Otherwise, we would assume special properties about s that may not be justified.

Consider $\exists$r, where we want to show an existentially quantified property. When a mathematician proves $\exists x\, \phi(x)$, he could directly produce any witness θ for this existential property and prove that, indeed, $\phi(\theta)$, for then he would have shown $\exists x\, \phi(x)$ with this witness. For example, to show that there is a number whose cube is less than its square, a mathematician could start by saying "let me choose 0.5 and show the property for 0.5". Then he could prove $0.5^3 < 0.5^2$, because $0.125 < 0.25$, and conclude that there, thus, is such a number, i.e., $\exists x\, (x^3 < x^2)$. Proof rule $\exists$r does that. It allows the choice of *any* term θ for x and accepts a proof of $\phi(\theta)$ as a proof of $\exists x\, \phi(x)$. However note that the claim "θ is a witness" may turn out to be wrong, for example, the choice 2 for x would be a bad start for attempting to show $\exists x\, (x^3 < x^2)$. Consequently, proof rule $\exists$r keeps both options $\phi(\theta)$ and $\exists x\, \phi(x)$ in the succedent.[4] If the proof with θ is successful, the sequent is valid and the part of the proof can be closed successfully. If the proof with θ later turns out to be unsuccessful, another attempt can be used to prove $\exists x\, \phi(x)$, e.g., by applying $\exists$r again with another attempt for a different witness θ_2.

Rules $\forall$l,$\exists$l are dual to $\exists$r,$\forall$l. Consider $\forall$l, where we have a universally quantified formula in the assumptions (antecedent) that we can use, and not in the succedent, which we want to show. In mathematics, when we know a universal fact, we can use this knowledge for any particular instance. If we know that all positive numbers have a square root, then we can also use the fact that 5 has a square root, because 5 is a positive number. Hence from assumption $\forall x\, (x > 0 \to \mathit{hasSqrt}(x))$ in the antecedent, we can also assume instance $5 > 0 \to \mathit{hasSqrt}(5))$. Rule $\forall$l can produce an instance $\phi(\theta)$ for arbitrary terms θ of the assumption $\forall x\, \phi(x)$. Since we may need the universal fact $\forall x\, \phi(x)$ for multiple instantiations with $\theta_1, \theta_2, \theta_3$ during the proof, rule $\forall$l keeps the

[4] KeYmaera does not actually keep $\exists x\, \phi(x)$ around in the succedent for rule $\exists$r and, for a fundamental reason [Pla08], does not have to. The same holds for rule $\forall$l, where KeYmaera does not keep $\forall x\, \phi(x)$ around in the antecedent, because it does not have to. That means, however, that if you conjecture θ to produce the right instance, and your conjecture turns out wrong during the proof, then you have to go back in the proof and undo your instantiation with θ.

assumption $\forall x\, \phi(x)$ in the antecedent so that it can be used repeatedly.

Consider rule ∃l in which we can use an existentially quantified formula from the antecedent. In mathematics, if we know an existential fact, then we can give a name to the object that we then know does exist. If we know that there is a smallest integer less than 10 that is a square, we can call it s, but we cannot denote it by a different term like 5, because 5 may be (and in fact is) the wrong answer. Rule ∃l gives a fresh name s (with all logical variables $X_1, \dots, X_n$ as arguments) to the object that exists. Since it does not make sense to give a different name for the same existing object later, $\exists x\, \phi(x)$ is removed from the antecedent when adding $\phi(s(X_1, \dots, X_n))$.

There are two ways of using the proof rules in Fig. 4. One way is to avoid free variables X_i altogether and only choose ground terms without variables for instantiations θ in ∃r,∀l. Then the Skolem functions used in ∀r,∃l have $n = 0$ free logical variables $X_1, \dots, X_n$ as arguments. This case is called a *ground calculus*, because free variables are never used and all term instantiations are ground (no free variables).

The other way is to work with free variables and always use some fresh (existential) logical variable X for instantiation of θ every time ∃r,∀l are used. This is a free-variable calculus [HS94, Fit96, FM99] where ∃r,∀l are called γ-rules and ∀r,∃l are called δ^+-rules [HS94], which is an improvement of what is known as the δ-rule [Fit96, FM99]. This case is called a *free-variable calculus*, because instantiations are with free variables. Later in the proof, these free variables can be requantified [Pla08]. The free variables $X_1, \dots, X_n$ in the Skolem terms keep track of the dependencies of symbols and prevent instantiations where we instantiate X_1 by a term such as $s(X_1, \dots, X_n)$ depending on X_1. The ground calculus and free-variable calculus uses of Fig. 4 can also be mixed.

8 Real Arithmetic

We will see more details on the handling of real arithmetic in a later lecture. In a nutshell, $\mathrm{QE}(\phi)$ denotes the use of real arithmetic on formula ϕ. That is, for a formula ϕ of first-order real arithmetic, $\mathrm{QE}(\phi)$ is a logical formula that is equivalent to ϕ but simpler, because $\mathrm{QE}(\phi)$ is quantifier-free.

Theorem 2 (Quantifier elimination)**.** *The first-order theory of real arithmetic admits* quantifier elimination *that is, with each formula ϕ, a quantifier-free formula $\mathrm{QE}(\phi)$ can be associated effectively that is equivalent (i.e., $\phi \leftrightarrow \mathrm{QE}(\phi)$ is valid) and has no additional free variables or function symbols. The operation QE is further assumed to evaluate ground formulas (i.e., without variables), yielding a decision procedure for closed formulas of this theory (i.e., formulas without free variables).*

Quantifier elimination yields, e.g., the following equivalence by real arithmetic:

$$\mathrm{QE}(\exists x\, (ax + b = 0)) \;\equiv\; (a \neq 0 \lor b = 0).$$

Both sides are easily seen to be equivalent, i.e.

$$\vDash \exists x\, (ax + b = 0) \leftrightarrow (a \neq 0 \lor b = 0)$$

because a linear equation with nonzero inhomogeneous part has a solution iff its linear part is nonzero as well. Real arithmetic equivalences can be used in differential dynamic logic to eliminate quantifiers (or otherwise simplify arithmetic).

With the rule i∀, we can reintroduce a universal quantifier for a Skolem term $s(X_1, \dots, X_n)$, which corresponds to a previously universally quantified variable in the succedent or a previously existentially quantified variable in the antecedent. The point of reintroducing the quantifier is that this makes sense when the remaining formulas are first-order in the quantified variable so that they can be handled equivalently by quantifier elimination in real-closed fields. When we have proven the subgoal (with for all X) then this entails the goal for the particular $s(X_1, \dots, X_n)$. In particular, when we remove a quantifier with ∀r,∃l to obtain a Skolem term, we can continue with other proof rules to handle the dynamic modalities and then reintroduce the quantifier for the Skolem term with i∀ once quantifier elimination for real arithmetic becomes applicable.

The dual rule i∃ can reintroduce an existential quantifier for a free logical variable that was previously existentially quantified in the succedent or previously universally quantified in the antecedent. Again, this makes sense when the resulting formula in the premise is first-order in the quantified variable X so that quantifier elimination can eliminate the quantifier equivalently. When we remove a quantifier with ∃r,∀l to obtain a free logical variable, we can continue using other proof rules to handle the dynamic modalities and then reintroduce the quantifier for the free logical variable with i∃ once quantifier elimination is applicable.

$$(\mathrm{i}\forall)\ \frac{\vdash \mathrm{QE}(\forall X\, (\Phi(X) \vdash \Psi(X)))}{\Phi(s(X_1,..,X_n)) \vdash \Psi(s(X_1,..,X_n))}\,{}^{1} \qquad (\mathrm{i}\exists)\ \frac{\vdash \mathrm{QE}(\exists X \bigwedge_i (\Phi_i \vdash \Psi_i))}{\Phi_1 \vdash \Psi_1 \quad \dots \quad \Phi_n \vdash \Psi_n}\,{}^{2}$$

[1] X is a new logical variable. Further, QE needs to be defined for the formula in the premise.

[2] Among all open branches, free logical variable X only occurs in the branches $\Phi_i \vdash \Psi_i$. Further, QE needs to be defined for the formula in the premise, especially, no Skolem dependencies on X can occur.

Recall abbreviations from Lecture 5:

$$\begin{aligned} A_{h,v} &\overset{\text{def}}{\equiv} 0 \le h \land h = H \land v = 0 \land g > 0 \land 1 > c \ge 0 \\ B_{h,v} &\overset{\text{def}}{\equiv} 0 \le h \land h \le H \\ (h'' = -g) &\overset{\text{def}}{\equiv} (h' = v, v' = -g) \end{aligned}$$

And the single-hop bouncing ball formula from Lecture 5:

$$A_{h,v} \to [h'' = -g; (?h = 0; v := -cv \cup ?h \ge 0)]B_{h,v}$$

We only consider a simpler formula instead:

$$A_{h,v} \to [h'' = -g]B_{h,v} \tag{3}$$

Let there be sequent proof:

$$
{\scriptstyle\to\mathrm{r}}\,\dfrac{{\scriptstyle[']\mathrm{r}}\,\dfrac{{\scriptstyle\forall\mathrm{r}}\,\dfrac{{\scriptstyle\to\mathrm{r}}\,\dfrac{{\scriptstyle\to\mathrm{r}}\,\dfrac{{\scriptstyle\forall\mathrm{l}}\,\dfrac{{\scriptstyle\to\mathrm{l}}\,\dfrac{{\scriptstyle ax}\,\dfrac{*}{A_{h,v}, r{\geq}0 \vdash 0{\leq}r{\leq}r} \qquad {\scriptstyle[:=]\mathrm{r}}\,\dfrac{A_{h,v}, r{\geq}0, H - \frac{g}{2}s^2 h \geq 0 \vdash B_{H-\frac{g}{2}r^2,-gt}}{A_{h,v}, r{\geq}0, [h := H - \frac{g}{2}s^2]h \geq 0 \vdash [h := H - \frac{g}{2}r^2]B_{h,v}}}{A_{h,v}, r{\geq}0, 0{\leq}r{\leq}r \to [h := H - \frac{g}{2}s^2]h \geq 0 \vdash [h := H - \frac{g}{2}r^2]B_{h,v}}}{A_{h,v}, r{\geq}0, \forall 0{\leq}s{\leq}r\, [h := H - \frac{g}{2}s^2]h \geq 0 \vdash [h := H - \frac{g}{2}r^2]B_{h,v}}}{A_{h,v}, r{\geq}0 \vdash \forall 0{\leq}s{\leq}r\, [h := H - \frac{g}{2}s^2]h \geq 0 \to [h := H - \frac{g}{2}r^2]B_{h,v}}}{A_{h,v} \vdash r{\geq}0 \to (\forall 0{\leq}s{\leq}r\, [h := H - \frac{g}{2}s^2]h \geq 0 \to [h := H - \frac{g}{2}r^2]B_{h,v})}}{A_{h,v} \vdash \forall t{\geq}0\, (\forall 0{\leq}s{\leq}t\, [h := H - \frac{g}{2}s^2]h \geq 0 \to [h := H - \frac{g}{2}t^2]B_{h,v})}}{A_{h,v} \vdash [h'' = -g \,\&\, h \geq 0]B_{h,v}}}{\vdash A_{h,v} \to [h'' = -g \,\&\, h \geq 0]B_{h,v}}
$$

We just wrote that the left premise closes by ax, except that

$$A_{h,v}, r{\geq}0 \vdash 0{\leq}r{\leq}r$$

is not exactly an instance of the ax rule, so even here we need simple arithmetic to conclude that $0 \leq r \leq r$ is the same as $r \geq 0$, at which point that premise turns into a literal instance of ax

$$A_{h,v}, r{\geq}0 \vdash r{\geq}0$$

A full formal proof and a KeYmaera proof, thus, need an extra proof step of arithmetic in the left premise.

The right premise is

$$A_{h,v}, r{\geq}0, H - \frac{g}{2}s^2 h \geq 0 \vdash B_{H-\frac{g}{2}r^2,-gt}$$

which, when resolving abbreviations turns into

$$0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 > c \geq 0, r{\geq}0, H - \frac{g}{2}s^2 h \geq 0 \vdash 0 \leq H - \frac{g}{2}r^2 \land H - \frac{g}{2}r^2 \leq H$$

This sequent proves using $\land$r plus simple arithmetic for the left branch resulting from $\land$r and a little more arithmetic on the right branch resulting from $\land$r. Finishing the above sequent proof up as indicated shows that d$\mathcal{L}$ formula (3) is provable.

9 Instantiating Real Arithmetic

Providing instantiations for quantifier rules $\exists$r,$\forall$l can speed up real arithmetic decision procedures. The proof in Sect. 8 instantiated the universal quantifier $\forall s$ for an evolution domain constraint by the end point r of the time interval using quantifier proof rule $\forall$l. This is a very common simplification that usually speeds up arithmetic significantly. It does not always work, because the instance one guesses may not always be the right one. Even worse, there may not always be a single instance that is sufficient for the proof, but that is a phenomenon that later lectures will examine.

10 Weakening Real Arithmetic

Weakening rules Wl,Wr can be useful to hide irrelevant parts of a sequent to make sure they do not be a distraction for real arithmetic decision procedures.

In the proof in Sect. 8, the left premise was

$$A_{h,v}, r{\geq}0 \vdash 0{\leq}r{\leq}r$$

The proof of this sequent did not make use of $A_{h,v}$ at all. Here, the proof worked easily. But if $A_{h,v}$ were a very complicated formula, then proving the same sequent might have been very difficult, because our proving attempts could have been distracted by the presence of $A_{h,v}$. We might have applied lots of proof rules to $A_{h,v}$ before finally realising that the sequent proves because of $r{\geq}0 \vdash 0{\leq}r{\leq}r$ alone.

The same kind of distraction can happen in decision procedures for real arithmetic, sometimes shockingly so [Pla10, Chapter 5]. Consequently, it can sometimes save a lot of proof effort to simplify irrelevant assumptions away as soon as they have become unnecessary. Fortunately, there already is a proof rule for that purpose called weakening, which we can use on our example from the left premise in the proof of Sect. 8:

$$\text{Wl}\,\frac{r{\geq}0 \vdash 0{\leq}r{\leq}r}{A_{h,v}, r{\geq}0 \vdash 0{\leq}r{\leq}r}$$

11 Summary

The differential dynamic logic sequent proof rules that we have seen in this lecture are summarized in Fig. 5. They turn out to be sound [Pla08]. Yet, the notion of soundness for axioms that we investigated in Lecture 5 does not directly apply to proof rules. We will investigate soundness of the proof rules in Fig. 5 in a later lecture. There are further proof rules of differential dynamic logic that later lectures will examine [Pla08].

References

[And02] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Kluwer, 2nd edition, 2002.

[Bus98] Samuel R. Buss. An introduction to proof theory. In Samuel R. Buss, editor, *Handbook of Proof Theory*, chapter 1, pages 1–78. Elsevier, 1998.

[Fit96] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, New York, 2nd edition, 1996.

[FM99] Melvin Fitting and Richard L. Mendelsohn. *First-Order Modal Logic*. Kluwer, Norwell, MA, USA, 1999.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen. I. *Math. Zeit.*, 39(2):176–210, 1935.

(¬r) $\dfrac{\Gamma, \phi \vdash \Delta}{\Gamma \vdash \neg\phi, \Delta}$ (∨r) $\dfrac{\Gamma \vdash \phi, \psi, \Delta}{\Gamma \vdash \phi \lor \psi, \Delta}$ (∧r) $\dfrac{\Gamma \vdash \phi, \Delta \quad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \phi \land \psi, \Delta}$

(¬l) $\dfrac{\Gamma \vdash \phi, \Delta}{\Gamma, \neg\phi \vdash \Delta}$ (∨l) $\dfrac{\Gamma, \phi \vdash \Delta \quad \Gamma, \psi \vdash \Delta}{\Gamma, \phi \lor \psi \vdash \Delta}$ (∧l) $\dfrac{\Gamma, \phi, \psi \vdash \Delta}{\Gamma, \phi \land \psi \vdash \Delta}$

(→r) $\dfrac{\Gamma, \phi \vdash \psi, \Delta}{\Gamma \vdash \phi \to \psi, \Delta}$ (ax) $\dfrac{}{\Gamma, \phi \vdash \phi, \Delta}$ (Wr) $\dfrac{\Gamma \vdash \Delta}{\Gamma \vdash \phi, \Delta}$

(→l) $\dfrac{\Gamma \vdash \phi, \Delta \quad \Gamma, \psi \vdash \Delta}{\Gamma, \phi \to \psi \vdash \Delta}$ (cut) $\dfrac{\Gamma \vdash \phi, \Delta \;\; \Gamma, \phi \vdash \Delta}{\Gamma \vdash \Delta}$ (Wl) $\dfrac{\Gamma \vdash \Delta}{\Gamma, \phi \vdash \Delta}$

(⟨;⟩) $\dfrac{\langle\alpha\rangle\langle\beta\rangle\phi}{\langle\alpha;\beta\rangle\phi}$ ($\langle{}^{*n}\rangle$) $\dfrac{\phi \lor \langle\alpha\rangle\langle\alpha^*\rangle\phi}{\langle\alpha^*\rangle\phi}$ (⟨:=⟩) $\dfrac{\phi_x^\theta}{\langle x := \theta\rangle\phi}$

([;]) $\dfrac{[\alpha][\beta]\phi}{[\alpha;\beta]\phi}$ ($[{}^{*n}]$) $\dfrac{\phi \land [\alpha][\alpha^*]\phi}{[\alpha^*]\phi}$ ([:=]) $\dfrac{\phi_x^\theta}{[x := \theta]\phi}$

(⟨∪⟩) $\dfrac{\langle\alpha\rangle\phi \lor \langle\beta\rangle\phi}{\langle\alpha \cup \beta\rangle\phi}$ (⟨?⟩) $\dfrac{H \land \psi}{\langle ?H\rangle\psi}$ (⟨′⟩) $\dfrac{\exists t{\geq}0\,\big((\forall 0{\leq}\tilde{t}{\leq}t\,\langle x := y(\tilde{t})\rangle H) \land \langle x := y(t)\rangle\phi\big)}{\langle x' = \theta \,\&\, H\rangle\phi}$ [1]

([∪]) $\dfrac{[\alpha]\phi \land [\beta]\phi}{[\alpha \cup \beta]\phi}$ ([?]) $\dfrac{H \to \psi}{[?H]\psi}$ ([′]) $\dfrac{\forall t{\geq}0\,\big((\forall 0{\leq}\tilde{t}{\leq}t\,[x := y(\tilde{t})]H) \to [x := y(t)]\phi\big)}{[x' = \theta \,\&\, H]\phi}$ [1]

(∃r) $\dfrac{\Gamma \vdash \phi(\theta), \exists x\,\phi(x), \Delta}{\Gamma \vdash \exists x\,\phi(x), \Delta}$ [2] (∀r) $\dfrac{\Gamma \vdash \phi(s(X_1, .., X_n)), \Delta}{\Gamma \vdash \forall x\,\phi(x), \Delta}$ [3]

(∀l) $\dfrac{\Gamma, \phi(\theta), \forall x\,\phi(x) \vdash \Delta}{\Gamma, \forall x\,\phi(x) \vdash \Delta}$ [2] (∃l) $\dfrac{\Gamma, \phi(s(X_1, .., X_n)) \vdash \Delta}{\Gamma, \exists x\,\phi(x) \vdash \Delta}$ [3]

(i∀) $\dfrac{\Gamma \vdash \mathrm{QE}(\forall X\,(\Phi(X) \vdash \Psi(X))), \Delta}{\Gamma, \Phi(s(X_1, .., X_n)) \vdash \Psi(s(X_1, .., X_n)), \Delta}$ [4] (i∃) $\dfrac{\Gamma \vdash \mathrm{QE}(\exists X\,\bigwedge_i(\Phi_i \vdash \Psi_i)), \Delta}{\Gamma, \Phi_1 \vdash \Psi_1, \Delta \;\; \ldots \;\; \Gamma, \Phi_n \vdash \Psi_n, \Delta}$ [5]

[1] t and $\tilde{t}$ are fresh logical variables and $\langle x := y(t)\rangle$ is the discrete assignment belonging to the solution y of the differential equation with constant symbol x as symbolic initial value.

[2] θ is an arbitrary term, often a new (existential) logical variable X.

[3] s is a new (Skolem) function and $X_1, .., X_n$ are all (existential) free logical variables of $\forall x\,\phi(x)$.

[4] X is a new logical variable. Further, QE needs to be defined for the formula in the premise.

[5] Among all open branches, free logical variable X only occurs in the branches $\Gamma, \Phi_i \vdash \Psi_i, \Delta$. Further, QE needs to be defined for the formula in the premise, especially, no Skolem dependencies on X can occur.

Figure 5: Some proof rules of the d$\mathcal{L}$ sequent calculus

[Hoa69] Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.

[HS94] Reiner Hähnle and Peter H. Schmitt. The liberalized δ-rule in free variable semantic tableaux. *J. Autom. Reasoning*, 13(2):211–221, 1994.

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

[Pla10] 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. Logics of dynamical systems. In *LICS*, pages 13–24. IEEE, 2012. doi:10.1109/LICS.2012.13.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Control Loops & Invariants

André Platzer

Carnegie Mellon University

Lecture 7

1 Introduction

Lecture 3 on Choice & Control demonstrated how important control is in CPS and that control loops are a very important feature for making this control happen. Without loops, CPS controllers are limited to short finite sequences of control actions, which are rarely sufficient. With loops, CPS controllers shine, because they can inspect the current state of the system, take action to control the system, let the physics evolve, and then repeat these steps in a loop over and over again to slowly get the state where the controller wants the system to be. Think of programming a robot to drive on a highway. Would you be able to do that without some means of repetition or iteration? Probably not, because you'll need to write a CPS program that monitors the traffic situation frequently and reacts in response to what the other cars do on the highway.

Hybrid programs' way of exercising repetitive control actions is the repetition operator * that can be applied to any hybrid program α. The resulting hybrid program α^* repeats α any number of times, nondeterministically.

More information can be found in [Pla12b, Pla12a] as well as [Pla10, Chapter 2.5.2,2.5.4].

2 Control Loops

Recall the little acrophobic bouncing ball from Lecture 4 on Safety & Contracts.

$$\begin{aligned}
&\texttt{@requires}(0 \leq h \land h = H \land v = 0)\\
&\texttt{@requires}(g > 0 \land 1 \geq c \geq 0)\\
&\texttt{@ensures}(0 \leq h \land h \leq H)\\
&\big(h' = v, v' = -g \,\&\, h \geq 0;\\
&\ \texttt{if}(h = 0)\, v := -cv\big)^*
\end{aligned} \tag{1}$$

The contracts above have been augmented with the ones that we have identified in Lecture 4 by converting the initial contract specification into a logical formula in differential dynamic logic and then identifying the required assumptions to make it true in all states:

$$0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 \geq c \geq 0 \rightarrow \\ \big[\big(h' = v, v' = -g \,\&\, h \geq 0;\ \mathtt{if}(h = 0)\, v := -cv\big)^*\big]\,(0 \leq h \land h \leq H) \qquad (2)$$

Because we did not want to be bothered by the presence of the additional `if-then-else` operator, which is not officially part of the minimal set of operators of $\mathsf{d}\mathcal{L}$, we simplified (2) to:

$$0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 \geq c \geq 0 \rightarrow \\ \big[\big(h' = v, v' = -g \,\&\, h \geq 0;\ (?h = 0; v := -cv \cup ?h \neq 0)\big)^*\big]\,(0 \leq h \land h \leq H) \qquad (3)$$

In Lecture 4, we had an informal understanding why (3) is valid (true in all states), but no formal proof, albeit we proved a much simplified version of (3) in which we simply threw away the loop. Ignorance is clearly not a correct way of understanding loops. Let's make up for that now by properly proving (3) in the $\mathsf{d}\mathcal{L}$ calculus.

Yet, before going for a proof, let us take a step back and understand the role of loops in more general terms. Their semantics has been explored in Lecture 3 on Choice & Control and more formally in Lecture 5 on Dynamical Systems & Dynamic Axioms.

The little bouncing ball had a loop in which physics and its bounce control alternated. The bouncing ball desperately needs a loop for it wouldn't know ahead of time how often it would bounce. When falling from great heights, it bounces quite a bit. The bouncing ball also has a controller, albeit a rather impoverished one. All it could do is inspect the current height, compare it to the ground floor (at height 0) and, if $h = 0$, flip its velocity vector around after a little damping by factor c. That is not a whole lot of flexibility for control choices, but the bouncing ball was still rather proud to serve such an important role in controlling the bouncing ball's behavior. Indeed, without the control action, the ball would never bounce back from the ground but would keep on falling forever—what a frightful thought for the acrophobic bouncing ball. On second thought, the ball would not fall for very long without its controller, because of the evolution domain $h \geq 0$ for physics $h'' = -g \,\&\, h \geq 0$, which would only allow physics to evolve for time zero if the ball is already at height 0, because gravity would otherwise try to pull it further down, except that $h \geq 0$ won't have it. So, in summary, without the bouncing ball's control statement, it would simply fall and then lie flat on the ground without time being allowed to proceed. That would not sound very reassuring and certainly not as much fun as bouncing back up, so the bouncing ball is really quite proud of its control.

This principle is not specific to the bouncing ball, but, rather, quite common in CPS. The controller performs a crucial task, without which physics would not evolve in the way that we want it to. After all, if physics did already always do what we want it to without any input from our side, we would not need a controller in the first place.

Hence, control is crucial and understanding and analyzing its effect on physics one of the primary responsibilities in CPS.

Before proving (3), we apply one more simplification that we have also done in Lecture 5, just to save space on the page. We boldly drop the evolution domain constraint and make up for it by modifying the condition in the second test (Exercise 1):

$$0 \le h \land h = H \land v = 0 \land g > 0 \land 1 \ge c \ge 0 \rightarrow \\ \big[\big(h' = v, v' = -g;\ (?h = 0; v := -cv \cup ?h \ge 0)\big)^*\big]\,(0 \le h \land h \le H) \quad (4)$$

Hold on, why is that okay? Doesn't our previous investigation say that the ball could suddenly fall through the cracks in the floor if physics insists on evolving for hours before giving the poor bouncing ball controller a chance to react? To make sure the bouncing ball does not panic in light of this threat, solve Exercise 1 to investigate this.

3 Proofs of Loops

There is a loop in (4). As we have seen, its behavior is crucial to the bouncing ball. So let's prove to understand what it does and to see whether we have to be just as nervous as the bouncing ball about losing it to the earth (if postcondition $0 \le h$ is not ensured) or to the sky (if $h \le H$ is not ensured).

Abbreviations have served us well in trying to keep proofs onto one page.

$$\begin{aligned} A_{h,v} &\overset{\text{def}}{\equiv} 0 \le h \land h = H \land v = 0 \land g > 0 \land 1 \ge c \ge 0 \\ B_{h,v} &\overset{\text{def}}{\equiv} 0 \le h \land h \le H \\ (h'' = -g) &\overset{\text{def}}{\equiv} (h' = v, v' = -g) \end{aligned}$$

With these abbreviations, the bouncing ball formula (4) turns into:

$$A_{h,v} \rightarrow [(h'' = -g; (?h = 0; v := -cv \cup ?h \ge 0))^*]B_{h,v} \quad (4)$$

This formula is swiftly turned into the sequent at the top using proof rule →r:

$$\rightarrow\text{r}\ \frac{A_{h,v} \vdash [(h'' = -g; (?h = 0; v := -cv \cup ?h \ge 0))^*]B_{h,v}}{\vdash A_{h,v} \rightarrow [(h'' = -g; (?h = 0; v := -cv \cup ?h \ge 0))^*]B_{h,v}}$$

This leaves a loop to be worried about. Inspecting our d$\mathcal{L}$ proof rules from Lecture 6 on Truth there is exactly one that addresses loops:

$$([^{*n}])\ \frac{\phi \land [\alpha][\alpha^*]\phi}{[\alpha^*]\phi}$$

Using this one to continue the sequent derivation proceeds as follows:

$$\text{[}^{*n}\text{]r}\;\dfrac{\text{∧r}\;\dfrac{\dfrac{*}{A_{h,v} \vdash B_{h,v}} \qquad \text{[;]r}\;\dfrac{A_{h,v} \vdash [h'' = -g][?h = 0; v := -cv \cup ?h \geq 0][(h'' = -g; (?h = 0; v := -cv \cup ?h \geq 0))^*]B_{h,v}}{A_{h,v} \vdash [h'' = -g; (?h = 0; v := -cv \cup ?h \geq 0)][(h'' = -g; (?h = 0; v := -cv \cup ?h \geq 0))^*]B_{h,v}}}{A_{h,v} \vdash B_{h,v} \wedge [h'' = -g; (?h = 0; v := -cv \cup ?h \geq 0)][(h'' = -g; (?h = 0; v := -cv \cup ?h \geq 0))^*]B_{h,v}}}{A_{h,v} \vdash [(h'' = -g; (?h = 0; v := -cv \cup ?h \geq 0))^*]B_{h,v}}$$

The left subgoal that results from using ∧r closes by very simple arithmetic. The right subgoal is more of a challenge to prove. We can solve the differential equation and proceed using [′]r, which will produce a quantifier that ∀r can handle and leaves us with a sequent that we need to consider further to prove.

4 Loops of Proofs

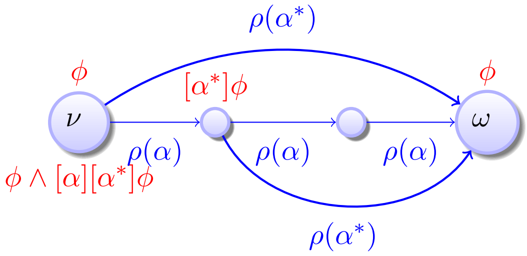
After a lot of proof effort, the above sequent prove continues so that the modalities

$$\ldots [h'' = -g][?h = 0; v := -cv \cup ?h \geq 0]\psi$$

can be handled. But there is still a loop in the postcondition ψ. How can we prove that postcondition, then? Investigating our proof rules, there is exactly one that addresses loops: $[^{*n}]$r again. If we use $[^{*n}]$r again, what will happen?

Recall from Lecture 5

$$\rho(\alpha^*) = \bigcup_{n\in\mathbb{N}} \rho(\alpha^n) \qquad \text{with} \quad \alpha^{n+1} \equiv \alpha^n; \alpha \text{ and } \alpha^0 \equiv ?true$$



Lemma 1 ([*] soundness)**.** *The iteration axiom is sound:*

$$([^*]) \;\; [\alpha^*]\phi \leftrightarrow \phi \wedge [\alpha][\alpha^*]\phi$$

Using proof rule $[^{*n}]$r on the succedent of a sequent has the same effect as using axiom [*] from left-hand side to right-hand side. Axiom [*] can be used to turn a formula

$$A \to [\alpha^*]B \tag{5}$$

into

$$A \to B \wedge [\alpha][\alpha^*]B$$

What happens if we use that axiom [*] again?

Recall that, unlike sequent proof rules such as $[^{*n}]$r, axioms do not say where they can be used, so we might as well use them anywhere in the middle of the formula. Hence using axiom [*] on the inner loop yields:

$$A \to B \land [\alpha](B \land [\alpha][\alpha^*]B)$$

Let's do that again and use [*] to obtain

$$A \to B \land [\alpha](B \land [\alpha](B \land [\alpha][\alpha^*]B)) \tag{6}$$

This is all very interesting but won't exactly get us any closer to a proof, because we could keep expanding the * star forever that way. How do we ever break out of this loop of never-ending proofs?

Before we get too disillusioned about our progress with [*] so far, notice that (6) still allows us to learn something about α and whether it always satisfies B when repeating α. Since [*] is an equivalence axiom, formula (6) still expresses the same thing as (5), i.e. that B always holds after repeating α when A was true in the beginning. Yet, (6) explicitly singles out the first 3 runs of α. Let's make this more apparent by recalling

$$([]\land)\ [\alpha](B \land \psi) \leftrightarrow [\alpha]B \land [\alpha]\psi$$

Using this valid equivalence turns (6) into

$$A \to B \land [\alpha]B \land [\alpha][\alpha](B \land [\alpha][\alpha^*]B)$$

Using $[]\land$ again gives us

$$A \to B \land [\alpha]B \land [\alpha]([\alpha]B \land [\alpha][\alpha][\alpha^*]B)$$

Using $[]\land$ once more gives

$$A \to B \land [\alpha]B \land [\alpha][\alpha]B \land [\alpha][\alpha][\alpha][\alpha^*]B \tag{7}$$

Looking at it this way, (7) could be more useful than the original (5), because, even though both are equivalent, (7) explicitly singles out the fact that B has to hold initially, after doing α once, after doing α twice, and that $[\alpha^*]B$ has to hold after doing α three times. Even if we are not quite sure what to make of the latter $[\alpha][\alpha][\alpha][\alpha^*]B$, because it still involves a loop, we are quite certain how to understand and handle the first three:

$$A \to B \land [\alpha]B \land [\alpha][\alpha]B \tag{8}$$

If this formula is not valid, then, certainly, neither is (7) and, thus, neither is the original (5). Hence, if we find a counterexample to (8), we disproved (7) and (5). That can actually be rather useful.

Yet, if (8) is still valid, we do not know whether (7) and (5) are, since they involve stronger requirements (B holds after any number of repetitions of α). What can we do then? Simply unroll the loop once more by using [*] on (6) to obtain

$$A \to B \land [\alpha](B \land [\alpha](B \land [\alpha](B \land [\alpha][\alpha^*]B))) \tag{9}$$

Or, equivalently, use axiom [*] on (7) to obtain the equivalent

$$A \rightarrow B \land [\alpha]B \land [\alpha][\alpha]B \land [\alpha][\alpha][\alpha](B \land [\alpha][\alpha^*]B) \tag{10}$$

By sufficiently many uses of axiom []∧, (9) and (10) are both equivalent to

$$A \rightarrow B \land [\alpha]B \land [\alpha][\alpha]B \land [\alpha][\alpha][\alpha]B \land [\alpha][\alpha][\alpha^*]B \tag{11}$$

which we can again examine to see if we can find a counterexample to the first part

$$A \rightarrow B \land [\alpha]B \land [\alpha][\alpha]B \land [\alpha][\alpha][\alpha]B$$

If yes, we disproved (5), otherwise we use [*] once more.

This process of iteratively unrolling a loop with either axiom [*] or rule $[^{*n}]$r is called *Bounded Model Checking* and has been used very successfully, e.g., in the context of finite-state systems [CBRZ01]. The same principle can be useful to disprove properties of loops in differential dynamic logic by unwinding the loop.

5 Breaking Loops for Proofs

Proving properties of loops by unwinding them forever with $[^{*n}]$r is not a promising strategy, unless we find that the conjecture is not valid after a number of unwindings. One way or another, we will have to find a way to break the loop apart to complete our reasoning.

Consider the formula (11) again that we got from (5) by unwinding the loop with axiom [*] a number of times and then flattening the formula with the help of []∧:

$$A \rightarrow B \land [\alpha]B \land [\alpha][\alpha]B \land [\alpha][\alpha][\alpha]B \land [\alpha][\alpha][\alpha^*]B \tag{11}$$

Using →r and ∧r on (11) leads to

$$\to\mathrm{r}\,\dfrac{\land\mathrm{r}\,\dfrac{A \vdash B \quad \land\mathrm{r}\,\dfrac{A \vdash [\alpha]B \quad \land\mathrm{r}\,\dfrac{A \vdash [\alpha][\alpha]B \quad \land\mathrm{r}\,\dfrac{A \vdash [\alpha][\alpha][\alpha]B \quad A \vdash [\alpha][\alpha][\alpha^*]B}{A \vdash [\alpha][\alpha][\alpha]B \land [\alpha][\alpha][\alpha^*]B}}{A \vdash [\alpha][\alpha]B \land [\alpha][\alpha][\alpha]B \land [\alpha][\alpha][\alpha^*]B}}{A \vdash [\alpha]B \land [\alpha][\alpha]B \land [\alpha][\alpha][\alpha]B \land [\alpha][\alpha][\alpha^*]B}}{A \vdash B \land [\alpha]B \land [\alpha][\alpha]B \land [\alpha][\alpha][\alpha]B \land [\alpha][\alpha][\alpha^*]B}}{\vdash A \rightarrow B \land [\alpha]B \land [\alpha][\alpha]B \land [\alpha][\alpha][\alpha]B \land [\alpha][\alpha][\alpha^*]B}$$

Let us summarize this notationally by the following

$$\to\mathrm{r},\land\mathrm{r},\land\mathrm{r},\land\mathrm{r},\land\mathrm{r}\,\dfrac{A \vdash B \quad A \vdash [\alpha]B \quad A \vdash [\alpha][\alpha]B \quad A \vdash [\alpha][\alpha][\alpha]B \quad A \vdash [\alpha][\alpha][\alpha^*]B}{\vdash A \rightarrow B \land [\alpha]B \land [\alpha][\alpha]B \land [\alpha][\alpha][\alpha]B \land [\alpha][\alpha][\alpha^*]B}$$

to recall that there was a derivation involving one use of →r and 4 uses of ∧r from the four premises to the single conclusion without saying which derivation it was exactly. Mentioning ∧r 4 times seems a bit repetitive, so simply abbreviate this as:

$$\to\mathrm{r},\land\mathrm{r}\,\dfrac{A \vdash B \quad A \vdash [\alpha]B \quad A \vdash [\alpha][\alpha]B \quad A \vdash [\alpha][\alpha][\alpha]B \quad A \vdash [\alpha][\alpha][\alpha^*]B}{\vdash A \rightarrow B \land [\alpha]B \land [\alpha][\alpha]B \land [\alpha][\alpha][\alpha]B \land [\alpha][\alpha][\alpha^*]B}$$

How could we prove the premises? Sect. 4 investigated one way, which essentially amounts to Bounded Model Checking. Can we be more clever and prove the same premises in a different way? Preferably one that is more efficient?

There is to much we can do to improve the way we prove the first premise. We simply have to bite the bullet and do it, armed with all our knowledge of arithmetic. But it's actually very easy at least for the bouncing ball. Besides, no dynamics has actually happened yet in the first premise, so if we despair in proving this one, the rest cannot become any easier either. For the second premise, there is not much that we can do, because we will have to analyze the effect of the loop body α running once at least in order to be able to understand what happens if we run α repeatedly.

Yet, what's with the third premise $A \vdash [\alpha][\alpha]B$? We could just approach it as is and try to prove it directly using the d$\mathcal{L}$ proof rules. Alternatively, however, we could try to take advantage of the fact that it is the same hybrid program α that is running in the first and the second modality. Maybe they should have something in common that we can exploit as part of our proof?

How could that work? Can we possibly find something that the is true after the first run of α and is all we need to know about the state for $[\alpha]B$ to hold? Can we characterize the intermediate state after the first α and before the second α? Suppose we manage to do that and identify a formula E that characterizes the intermediate state in this way. How do we use intermediate condition E to simplify our proof?

Recall the intermediate condition contract version of the sequential composition proof rule from Lecture 4 and Lecture 5.

$$\text{(R4)}\ \frac{A \rightarrow [\alpha]E \quad E \rightarrow [\beta]B}{A \rightarrow [\alpha;\beta]B}$$

Lecture 5 ended up dismissing the intermediate contract rule R4 in favor of the more general axiom

$$([;])\ [\alpha;\beta]\phi \leftrightarrow [\alpha][\beta]\phi$$

But, let us revisit R4 and see if we can learn something from its way of using intermediate condition E. The first obstacle is that the conclusion of R4 does not match the form we need for $A \vdash [\alpha][\alpha]B$. That's not a problem in principle, because we could use axiom [;] backwards from right-hand side to left-hand side in order to turn $A \vdash [\alpha][\alpha]B$ into

$$A \vdash [\alpha;\alpha]B$$

and then use rule R4. However, this is what we wanted to stay away from, because using the axioms both forwards and backwards can get our proof search into trouble because we might loop around trying to find a proof forever without making any progress by simply using [;] forwards and then backwards and then forwards again and so on until the end of time. That does not strike us as useful. Instead, we'll adopt a proof rule that has some of the thoughts of R4 but is more general. It is called *generalization*:

$$([]gen')\ \frac{\Gamma \vdash [\alpha]\phi, \Delta \quad \phi \vdash \psi}{\Gamma \vdash [\alpha]\psi, \Delta}$$

Rule $[]gen'$ on the third premise $A \vdash [\alpha][\alpha]B$ with the intermediate condition E for ϕ that we assume to have identified

$$[]gen'\ \frac{A \vdash [\alpha]E \qquad E \vdash [\alpha]B}{A \vdash [\alpha][\alpha]B}$$

Let us try to use this principle to see if we can find a way to prove

$$A \to B \land [\alpha](B \land [\alpha](B \land [\alpha](B \land [\alpha][\alpha^*]B))) \tag{9}$$

Using $\land$r and $[]gen'$ a number of times for a sequence of intermediate conditions E_1, E_2, E_3 derives:

$$\rightarrow\!\mathrm{r}\ \dfrac{\land\mathrm{r}\ \dfrac{A \vdash B \qquad []gen'\ \dfrac{A \vdash [\alpha]E_1 \qquad \land\mathrm{r}\ \dfrac{E_1 \vdash B \qquad []gen'\ \dfrac{E_1 \vdash [\alpha]E_2 \qquad \land\mathrm{r}\ \dfrac{E_2 \vdash B \qquad []gen'\ \dfrac{E_2 \vdash [\alpha]E_3 \qquad \land\mathrm{r}\ \dfrac{E_3 \vdash B \qquad E_3 \vdash [\alpha][\alpha^*]B}{E_3 \vdash B \land [\alpha][\alpha^*]B}}{E_2 \vdash [\alpha](B \land [\alpha][\alpha^*]B)}}{E_2 \vdash B \land [\alpha](B \land [\alpha][\alpha^*]B)}}{E_1 \vdash [\alpha](B \land [\alpha](B \land [\alpha][\alpha^*]B))}}{E_1 \vdash B \land [\alpha](B \land [\alpha](B \land [\alpha][\alpha^*]B))}}{A \vdash [\alpha](B \land [\alpha](B \land [\alpha](B \land [\alpha][\alpha^*]B)))}}{A \vdash B \land [\alpha](B \land [\alpha](B \land [\alpha](B \land [\alpha][\alpha^*]B)))}}{\vdash A \to B \land [\alpha](B \land [\alpha](B \land [\alpha](B \land [\alpha][\alpha^*]B)))}$$

This particular derivation is still not very useful because it still has a loop in one of the premises, which is what we had originally started out with in (5) in the first place. But the derivation hints at a useful way how we could possibly shortcut proofs. To lead to a proof of the conclusion, the above derivation requires us to prove the premises

$$A \vdash [\alpha]E_1$$
$$E_1 \vdash [\alpha]E_2$$
$$E_2 \vdash [\alpha]E_3$$

as well as some other premises. What if all the intermediate conditions E_i were the same? Let's assume they are all the same condition E, that is, $E_1 \equiv E_2 \equiv E_3 \equiv E$. Then most of the premises turn out to be the same:

$$E \vdash B$$
$$E \vdash [\alpha]E$$

except for the two left-most and the right-most premise. Let us leverage this observation and develop a proof rule for which the same intermediate condition is used for all iterates of the loop. Furthermore, we would even know the first premise

$$A \vdash [\alpha]E$$

if we could prove that the precondition A implies E:

$$A \vdash E$$

because, we already have $E \vdash [\alpha]E$.

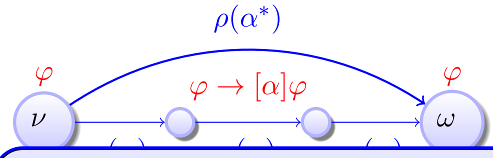
6 Invariant Proofs of Loops

The condition $E \vdash [\alpha]E$ identified in the previous section seems particularly useful, because it basically says that whenever the system α starts in a state satisfying E, it will stay in E. It sounds like the system α^* couldn't get out of E either if it starts in E since all that α^* can do is to repeat α some number of times. But every time we repeat α, the sequent $E \vdash [\alpha]E$ expresses that we cannot leave E that way.

The other condition that the previous section identified as crucial is $E \vdash B$. And, indeed, if E does not imply the postcondition B that we have been interested in in the first place, then E is not necessarily very useful to prove B.

Recall from Lecture 3

$$\rho(\alpha^*) = \bigcup_{n\in\mathbb{N}} \rho(\alpha^n) \qquad \text{with} \quad \alpha^{n+1} \equiv \alpha^n; \alpha \text{ and } \alpha^0 \equiv ?\mathit{true}$$



Lemma 2 (Induction). *The induction rule is sound:*

$$(ind')\ \frac{\Gamma \vdash \varphi, \Delta \quad \varphi \vdash [\alpha]\varphi \quad \varphi \vdash \psi}{\Gamma \vdash [\alpha^*]\psi, \Delta}$$

First observe that the *inductive invariant* φ (which we called E in the previous examples) occurs in all premises but not in the conclusion of ind'. The first premise of ind' says that the initial state, about which we assume Γ (and that Δ does not hold), satisfies the invariant φ. The second premise of ind' shows that the invariant φ is inductive. That is, whenever φ was true before running the loop body α, then φ is always true again after running α. The third premise of ind' shows that the invariant φ is strong enough to imply the postcondition ψ that the conclusion was interested in.

Rule ind' says that ψ holds after any number of repetitions of α if an invariant φ holds initially (left premise) and invariant φ remains true after one iteration of α (middle premise), and invariant φ finally implies the desired postcondition ψ (right premise). If φ is true after executing α whenever φ has been true before (middle premise), then, if φ holds in the beginning (left premise), φ will continue to hold, no matter how often we repeat α in $[\alpha^*]\psi$, which is enough to imply $[\alpha^*]\psi$ if φ implies ψ.

Taking a step back, these three premises correspond exactly to the proof steps that 15-122 Principles of Imperative Computation used to show that the contract of a function with a `@requires` contract Γ (and not Δ), `@ensures` contract ψ, and a loop invariant φ is correct. Now, we have this reasoning in a more general and formally more precisely defined context.

7 A Proof of a Repetitive Bouncing Ball

$$
\begin{aligned}
&\texttt{@requires}(0 \le h \wedge h = H \wedge v = 0)\\
&\texttt{@requires}(g > 0 \wedge 1 \ge c \ge 0)\\
&\texttt{@ensures}(0 \le h \wedge h \le H)\\
&\big(h' = v, v' = -g \,\&\, h \ge 0;\\
&\ (?h = 0; v := -cv \cup ?h \ge 0)\big)^* \texttt{@invariant}(2gh = 2gH - v^2 \wedge h \ge 0)
\end{aligned}
\tag{12}
$$

Let us again use abbreviations:

$$
\begin{aligned}
A_{h,v} &\overset{\text{def}}{\equiv} 0 \le h \wedge h = H \wedge v = 0 \wedge g > 0 \wedge 1 \ge c \ge 0\\
B_{h,v} &\overset{\text{def}}{\equiv} 0 \le h \wedge h \le H\\
(h'' = \dots) &\overset{\text{def}}{\equiv} (h' = v, v' = -g \,\&\, h \ge 0)\\
E_{h,v} &\overset{\text{def}}{\equiv} 2gh = 2gH - v^2 \wedge h \ge 0
\end{aligned}
$$

Note the somewhat odd abbreviation for the differential equation just to simplify notation. Also note the invariant $E_{h,v}$ that we identified as an intermediate condition for the single-hop bouncing ball in Lecture 4 on Safety & Contracts. After the considerations in Sect. 5, it should no longer be a big surprise why we try to use an intermediate condition as an invariant. We are not sure whether this will work but it seems worth trying.

$$A_{h,v} \to [(h' = \dots; (?h = 0; v := -cv \cup ?h \ge 0)^*]B_{h,v}$$

Let there be proof.

$$
\scriptstyle\to\mathrm{r}\
\dfrac{
\scriptstyle ind'\
\dfrac{
A_{h,v} \vdash E_{h,v}
\quad
\scriptstyle [;]\mathrm{r}\
\dfrac{
\scriptstyle []gen'\
\dfrac{
E_{h,v} \vdash [h' = \dots]E_{h,v}
\quad
\scriptstyle [\cup]\mathrm{r}\
\dfrac{
\scriptstyle \wedge\mathrm{r}\
\dfrac{
\scriptstyle [;]\mathrm{r}\
\dfrac{
\scriptstyle [?]\mathrm{r}\
\dfrac{
\scriptstyle [:=]\mathrm{r}\
\dfrac{E_{h,v}, h = 0 \vdash E_{h,-cv}}{E_{h,v}, h = 0 \vdash [v := -cv]E_{h,v}}
}{E_{h,v} \vdash [?h = 0][v := -cv]E_{h,v}}
}{E_{h,v} \vdash [?h = 0; v := -cv]E_{h,v}}
\quad
\scriptstyle [?]\mathrm{r}\
\dfrac{E_{h,v}, h \ge 0 \vdash E_{h,v}}{E_{h,v} \vdash [?h \ge 0]E_{h,v}}
}{E_{h,v} \vdash [?h = 0; v := -cv]E_{h,v} \wedge [?h \ge 0]E_{h,v}}
}{E_{h,v} \vdash [?h = 0; v := -cv \cup ?h \ge 0]E_{h,v}}
}{E_{h,v} \vdash [h' = \dots][?h = 0; v := -cv \cup ?h \ge 0]E_{h,v}}
}{E_{h,v} \vdash [h' = \dots; (?h = 0; v := -cv \cup ?h \ge 0]E_{h,v}}
\quad
E_{h,v} \vdash B_{h,v}
}{A_{h,v} \vdash [(h' = \dots; (?h = 0; v := -cv \cup ?h \ge 0)^*]B_{h,v}}
}{\vdash A_{h,v} \to [(h' = \dots; (?h = 0; v := -cv \cup ?h \ge 0)^*]B_{h,v}}
$$

The remaining 5 premises are prove easily. The first premise $A_{h,v} \vdash E_{h,v}$ proves easily using $h = H$ and $v = 0$:

$$0 \le h \wedge h = H \wedge v = 0 \wedge g > 0 \wedge 1 \ge c \ge 0 \vdash 2gh = 2gH - v^2 \wedge h \ge 0$$

Recalling the unusual abbreviations, the second premise $E_{h,v} \vdash [h' = \dots]E_{h,v}$ is

$$2gh = 2gH - v^2 \wedge h \ge 0 \vdash [h' = v, v' = -g \,\&\, h \ge 0](2gh = 2gH - v^2 \wedge h \ge 0)$$

a proof whose pieces we have seen in previous lectures (Exercise 2). The third premise $E_{h,v}, h = 0 \vdash E_{h,-cv}$ is

$$2gh = 2gH - v^2 \land h \geq 0, h = 0 \vdash 2gh = 2gH - (-cv)^2 \land h \geq 0$$

which would prove easily if we knew $c = 1$. Do we know $c = 1$? No we do not know $c = 1$, because we only assumed $1 \geq c \geq 0$ in $A_{h,v}$. But we could prove this third premise easily if we would change the definition of $A_{h,v}$ around to include $c = 1$. Note that even then, however, we still need to augment $E_{h,v}$ to include $c = 1$ as well, since we otherwise would have lost this knowledge before we need it in the third premise. The fourth premise, $E_{h,v}, h \geq 0 \vdash E_{h,v}$ proves whatever the abbreviations stand for simply using the axiom rule ax. Finally, the fifth premise $E_{h,v} \vdash B_{h,v}$, which is

$$2gh = 2gH - v^2 \land h \geq 0 \vdash 0 \leq h \land h \leq H$$

proves easily with arithmetic as long as we know $g > 0$. This condition is already included in $A_{h,v}$. But we still managed to forget about that in our intermediate condition. So, again, $g > 0$ should have been included in the invariant $E_{h,v}$, which should have been defined as

$$E_{h,v} \overset{\text{def}}{\equiv} 2gh = 2gH - v^2 \land h \geq 0 \land c = 1 \land g > 0$$

Yet, only the last two conjuncts are trivial, because neither c nor g changes while the little bouncing ball falls. We, unfortunately, still have to include it in the invariant. This is one of the downsides of working with intermediate condition style proofs such as what we get with rule $[]gen'$. Later lectures investigate significant simplifications for this nuisance.

For the record, we now have a sequent proof of the undamped bouncing ball with repetitions:

$$\begin{aligned}&0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 = c \rightarrow \\ &\qquad [(h' = v, v' = -g \,\&\, h \geq 0; (?h = 0; v := -cv \cup ?h \geq 0))^*](0 \leq h \land h \leq H)\end{aligned} \tag{13}$$

Looking back, the contract in (12) has almost reflected this, but not quite, because the `@invariant` contract forgot to capture the constant invariants $c = 1 \land g > 0$. And the `@requires` contract forgot to require $c = 1$. Let's capture this contract, which we have now verified by way of proving the corresponding d$\mathcal{L}$ formula (13):

$$\begin{aligned}&\texttt{@requires}(0 \leq h \land h = H \land v = 0)\\ &\texttt{@requires}(g > 0 \land c = 1)\\ &\texttt{@ensures}(0 \leq h \land h \leq H)\\ &\big(h' = v, v' = -g \,\&\, h \geq 0;\\ &\quad (?h = 0; v := -cv \cup ?h \geq 0)\big)^*\texttt{@invariant}(2gh = 2gH - v^2 \land h \geq 0 \land c = 1 \land g > 0)\end{aligned} \tag{14}$$

8 Essentials of Induction & Cuts

The induction rule ind' is very useful in practice. But there is a more elegant and more essential way of stating the induction principle.

Lemma 3 (Induction)**.** *The induction rule is sound:*

$$(ind)\ \dfrac{\varphi \vdash [\alpha]\varphi}{\varphi \vdash [\alpha^*]\varphi}$$

ind is clearly a special case of ind', obtained by specializing $\Gamma \overset{\text{def}}{\equiv} \psi\ \Delta = .$, and $\varphi \overset{\text{def}}{\equiv} \psi$, in which case the left and right premises of ind' are provable directly by ax so that only the middle premise remains. If ind is a special case of ind', why should we still prefer ind from a perspective of essentials? Obviously, ind is more fundamental and easier. But if this came at the cost of being less powerful, ind' should still be preferred. It turns out that ind' is actually a special case of ind with a little extra work. This extra work needs a bit of attention but is insightful.

Let's adopt the following variation of the generalization rule:

$$([]gen)\ \dfrac{\phi \vdash \psi}{[\alpha]\phi \vdash [\alpha]\psi}$$

For example, using a cut with $\varphi \to [\alpha^*]\varphi$, rule ind' can be derived from ind and $[]gen$ as follows (using weakening Wl,Wr without notice):

$$\text{cut}\ \dfrac{{\to}\text{r}\ \dfrac{ind\ \dfrac{\varphi \vdash [\alpha]\varphi}{\varphi \vdash [\alpha^*]\varphi}}{\Gamma \vdash \varphi \to [\alpha^*]\varphi, \Delta} \qquad {\to}\text{l}\ \dfrac{\Gamma \vdash \varphi, \Delta \qquad []gen\ \dfrac{\varphi \vdash \psi}{[\alpha^*]\varphi \vdash [\alpha^*]\psi}}{\Gamma, \varphi \to [\alpha^*]\varphi \vdash [\alpha^*]\psi, \Delta}}{\Gamma \vdash [\alpha^*]\psi, \Delta}$$

Hence ind' is a derived rule, because it can be derived using ind and some other rules. Thus, ind' is not necessary in theory, but still useful in practice.

Yet, now, in order to derive rule ind' out of the more fundamental ind, we had to add the revised generalization rule $[]gen$. Is that any easier? Well it is, because $[]gen$ actually makes $[]gen'$ unnecessary by another smart argument using a cut with the desired formula $[\alpha]\phi$.

$$\text{cut}\ \dfrac{\text{Wr}\ \dfrac{\Gamma \vdash [\alpha]\phi, \Delta}{\Gamma \vdash [\alpha]\phi, [\alpha]\psi, \Delta} \qquad \text{Wl,Wr}\ \dfrac{[]gen\ \dfrac{\phi \vdash \psi}{[\alpha]\phi \vdash [\alpha]\psi}}{\Gamma, [\alpha]\phi \vdash [\alpha]\psi, \Delta}}{\Gamma \vdash [\alpha]\psi, \Delta}$$

This leaves exactly the premises of rule $[]gen'$, making $[]gen'$ a derived rule. Whenever we need $[]gen'$, we could simply expand the proof out in the above form to reduce it just a proof involving $[]gen$ and cut and weakening.

These are two illustrations how creative uses of cuts can suddenly make proves and concepts easier. A phenomenon that we will see in action much more often in this course.

Before you despair that you would have to derive ind' and $[]gen'$ every time you need them: that is not the case. The theorem prover KeYmaera is very well aware of how useful both versions of the proof rules are and has them at your disposal. For theoretical investigations, however, as well as for understanding the truly fundamental reasoning steps, it is instructive to see that ind and $[]gen$ are fundamental, while the others are mere consequences.

Exercises

Exercise 1 (Give bouncing ball back its evolution domain). Explain why the transformation from (3) to (4) was okay in this case.

Exercise 2. Give a sequent proof for

$$2gh = 2gH - v^2 \land h \geq 0 \rightarrow [h' = v, v' = -g \,\&\, h \geq 0](2gh = 2gH - v^2 \land h \geq 0)$$

Does this property also hold if we remove the evolution domain constraint $h \geq 0$? That is, is the following formula valid?

$$2gh = 2gH - v^2 \land h \geq 0 \rightarrow [h' = v, v' = -g](2gh = 2gH - v^2 \land h \geq 0)$$

Exercise 3. To develop an inductive proof rule, we have started systematic unwinding considerations from formula (9) in Sect. 5. In lecture, we started from the form (11) instead and have seen that that takes us to the same inductive principle. Which of the two ways of proceeding is more efficient? Which one produces less premises that are distractions in the argument? Which one has less choices of different intermediate conditions E_i in the first place?

References

[CBRZ01] Edmund M. Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Form. Methods Syst. Des.*, 19(1):7–34, 2001.

[Pla10] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14509-4.

[Pla12a] André Platzer. Dynamic logics of dynamical systems. *CoRR*, abs/1205.4788, 2012. arXiv:1205.4788.

[Pla12b] André Platzer. Logics of dynamical systems. In *LICS*, pages 13–24. IEEE, 2012. doi:10.1109/LICS.2012.13.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Events & Delays

André Platzer

Carnegie Mellon University
Lecture 8

1 Introduction

Lecture 3 on Choice & Control demonstrated the importance of control and loops in CPS models, Lecture 5 on Dynamical Systems & Dynamic Axioms presented a way of unwinding loops iteratively to relate repetition to runs of the loop body, Lecture 6 on Truth & Proof showed a corresponding way of unwinding loops in sequent calculus, and Lecture 7 on Control Loops & Invariants finally explained the central proof principle for loops based on induction.

That has been a lot of attention on loops, but there are even more things to be learned about loops. Not by coincidence, because loops are one of the difficult challenges in CPS. The other difficult challenge comes from the differential equations. If the differential equations are simple and there are no loops, CPS suddenly become easy (they are even decidable).

This lecture will focus on how the two difficult parts of CPS interact: how loops interface with differential equations. That interface is ultimately the connection between the cyber and the physical part, which, as we know since Lecture 2 on Differential Equations & Domains, is fundamentally represented by the evolution domain constraints that determine when physics pauses to let cyber look and act.

Today's lecture focuses on two important paradigms for making cyber interface with physics to form cyber-physical systems, which played an equally important role in classical embedded systems. One paradigm is that of *event-driven architecture*, where reactions to events dominate the behavior of the system. The other paradigm is *time-triggered control*, which use periodic actions to affect the behavior of the system. Both paradigms fall out naturally from an understanding of the hybrid program principle for CPS.

These lecture notes are loosely based on [Pla12b, Pla10].

2 The Need for Control

Having gotten accustomed to the little bouncing ball, this lecture will simply stick to it. Yet, the bouncing ball asks for more action, for it had so far no choice but to wait until it was at height $h = 0$. And when its patience paid off so that it finally observed height $h = 0$, then its only action was to make its velocity bounce back up. Frustrated by this limited menu of actions to choose from, the bouncing ball asks for a ping pong paddle. Thrilled at the opportunities opened up by a ping pong paddle, the bouncing ball first performs some experiments and then settles on using the ping pong paddle high up in the air to push itself back down again. It had high hopes that proper control exerted by the ping pong paddle would allow the ball to go faster without risking the terrified moments inflicted on it by its acrophobic attitude to heights. Setting aside all Münchausian concerns about how effective ping pong paddles can be for the ball if itself is in control of the paddle to be used on itself in light of Newton's third law about opposing forces, let us investigate this situation regardless. After all, it has what it takes to make control interesting: the dynamics of a physical system and decisions on when to react how to the observed status of the system.

Lecture 7 developed a sequent proof of the undamped bouncing ball with repetitions:

$$0 \leq h \land h = H \land v = 0 \land g > 0 \land 1 = c \rightarrow \\ \left[\left(h' = v, v' = -g \,\&\, h \geq 0; (?h = 0; v := -cv \cup ?h \geq 0)\right)^*\right](0 \leq h \land h \leq H) \quad (1)$$

Figure 1: Sample trajectory of a bouncing ball (plotted as position over time)

With this basic understanding of (undamped) bouncing balls, let's examine how to turn the bouncing ball into a ping pong ball using clever actuation of a ping pong paddle. The bouncing ball tried to actuate the ping pong paddle in all kinds of directions. But it never knew where it was going to land if it tried the ping pong paddle sideways. So it quickly gave up the thought of using the ping pong paddle sideways. The ball got so accustomed to its path of going up and down on the spot. With the ping pong paddle, it wanted to do the same, just faster.

By making the ping pong paddle move up and down, the bouncing ball ultimately figured out that the ball would go back down fast as soon as it got a pat on the top by the paddle. It also learned that the other direction turned out to be not just difficult but also rather dangerous. Moving the ping pong paddle up when the ball was above it to give it a pat on the bottom was tricky, but when it worked would even make the ball fly up higher than before. Yet that is what the acrophobic bouncing ball did not enjoy so much, so it tries to control the ping pong paddle so that the ball only bounces down, never up.

As a height that the bouncing ball feels comfortable with, it chose 5 and so it wants to establish $0 \leq h \leq 5$ to always hold as its safety condition. The ball further puts the ping pong paddle at a similar height so that it can actuate somewhere between 4 and 5. It exercises great care to make sure it would every only move the paddle downwards when the ball is underneath, never above. Thus, the effect of the ping pong paddle will be to reverse the ball's direction. For simplicity, the ball figures that being hit by a ping pong paddle might have a similar effect as being hit by the floor, except with a possibly different factor $f > 0$ instead of the damping coefficient c.[1] So the paddle actuated this way is simply assumed to have effect $v := -fv$.

Taking these thoughts into account, the ball devises the following HP model and conjectures safety expressed in the following d$\mathcal{L}$ formula:

$$\begin{aligned}&0 \leq h \land h \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow\\ &\quad \big[\big(h' = v, v' = -g \,\&\, h \geq 0;\\ &\qquad (?h = 0; v := -cv \cup ?4 \leq h \leq 5; v := -fv \cup ?h \geq 0)\big)^*\big](0 \leq h \leq 5)\end{aligned} \tag{2}$$

Having taken the *Principle of Cartesian Doubt* from Lecture 4 on Safety & Contracts to heart, the aspiring ping-pong ball first scrutinizes conjecture (2) before setting out to prove it. What could go wrong?

For one thing, (2) allows the right control options of using the paddle by $?4 \leq h \leq 5; v := -fv$ but also always allows the wrong choice $?h \neq 0$ when above ground. So if the bouncing ball is unlucky, the HP in (2) could run so that the middle choice is never chosen and, if the ball has a large downwards velocity v initially, it will jump back up higher than 5 even if it was below 5 initially. That scenario falsifies (2) and a concrete counterexample can be constructed correspondingly, e.g., from initial state ν with

$$\nu(h) = 5, \nu(v) = -10^{10}, \nu(c) = \frac{1}{2}, \nu(f) = 1, \nu(g) = 10$$

How can the bouncing ball bugfix its control and turn itself into a proper ping pong ball? The problem with the controller in (2) is that it permits too much choice, some of which are unsafe. Restricting these choices and making them more deterministic is what it takes to ensure the ping pong paddle is actuated as intended:

$$\begin{aligned}&0 \leq h \land h \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow\\ &\quad \big[\big(h' = v, v' = -g \,\&\, h \geq 0;\\ &\qquad (?h = 0; v := -cv \cup ?4 \leq h \leq 5; v := -fv \cup ?h \geq 0 \land h < 4 \lor h > 5)\big)^*\big](0 \leq h \leq 5)\end{aligned} \tag{3}$$

[1] The real story is quite a bit more complicated, but the bouncing ball does not know any better.

Recalling the $\texttt{if}(E)\,\alpha\,\texttt{else}\,\beta$ statement, the same system can be modeled equivalently:

$$\begin{aligned}&0 \le h \land h \le 5 \land v \le 0 \land g > 0 \land 1 \ge c \ge 0 \land f > 0 \to\\ &\quad\big[\big(h' = v, v' = -g \,\&\, h \ge 0;\\ &\qquad(?h = 0; v := -cv \cup ?h \ne 0; \texttt{if}(4 \le h \le 5)\, v := -fv)\big)^*\big](0 \le h \le 5)\end{aligned}$$

Or, even shorter as the equivalent

$$\begin{aligned}&0 \le h \land h \le 5 \land v \le 0 \land g > 0 \land 1 \ge c \ge 0 \land f > 0 \to\\ &\quad\big[\big(h' = v, v' = -g \,\&\, h \ge 0;\\ &\qquad\texttt{if}(h = 0)\, v := -cv \,\texttt{else}\,\texttt{if}(4 \le h \le 5)\, v := -fv\big)^*\big](0 \le h \le 5)\end{aligned} \tag{4}$$

Is conjecture (4) valid?

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

3 Events in Control

The problem with (4) is that, even though it exercises the right control choice whenever the controller runs, the model does not ensure the controller would run at all when needed. The paddle control only runs after the differential equation stops. That is guaranteed to happen when the ball bounces down to the ground ($h = 0$) but could otherwise be any time. Recall from Lecture 2 that the semantics of differential equations is nondeterministic. The system can follow a differential equation any amount of time as long as it does not violate the evolution domain constraints. In particular, the HP in (4) could miss the *event* $4 \leq h \leq 5$ that the ping pong ball's paddle control wanted to react to. The system might simply skip over that region by following the differential equation $h' = v, v' = -g \,\&\, h \geq 0$ obliviously.

How can the HP from (4) be modified to make sure the event $4 \leq h \leq 5$ is always noticed and never missed?

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

The "only" way to prevent the system from following a differential equation for too long is to restrict the evolution domain constraint, which is the predominant way to make cyber and physical interact. Indeed, that is what the evolution domain constraint $\dots \,\&\, h \geq 0$ in (4) did in the first place. Even though this domain was introduced for different reasons (first principle arguments that light balls never fall through solid ground), its secondary effect was to make sure that the ground controller $?h = 0; v := -cv$ will never miss the right time to take action and reverse the direction of the ball from falling to climbing.

Note 1 (Evolution domains detect events)**.** *Evolution domain constraints of differential equations in hybrid programs can detect events. That is, they can make sure the system evolution stops whenever an event happens on which the control wants to take action. Without such evolution domain constraints, the controller is not necessarily guaranteed to execute but may miss the event.*

Following these thoughts further indicates that the evolution domain somehow ought to be augmented with more constraints that ensure the interesting event $4 \leq h \leq 5$ will never be missed accidentally. How can this be done? Should the event be conjoined to the evolution domain as follows

$$\begin{aligned}
&0 \leq h \land h \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow \\
&\qquad \big[\big(h' = v, v' = -g \,\&\, h \geq 0 \land 4 \leq h \leq 5; \\
&\qquad\quad \texttt{if}(h = 0)\, v := -cv \,\texttt{else}\, \texttt{if}(4 \leq h \leq 5)\, v := -fv\big)^*\big](0 \leq h \leq 5)
\end{aligned}$$

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

Of course not! This evolution domain would require the ball to always be at height between 4 and 5, which is hardly the right model. How could the ball ever fall on the ground and bounce back, this way? It couldn't.

Yet, on second thought, the way the event $\ldots \,\&\, h = 0$ got detected by the HP in the first place was not by including $h = 0$ in the evolution domain constraint, but by including the inclusive limiting constraint $\ldots \,\&\, h \geq 0$, which made sure the system could perfectly well evolve outside this event domain $h = 0$, but that it couldn't just miss the event rushing past $h = 0$. What would the inclusion of such an inclusivelimiting constraint correspond to for the event $4 \leq h \leq 5$?

When the ball is hurled up into the sky, the last point at which action has to be taken to make sure not to miss the event $4 \leq h \leq 5$ is $h = 5$. The corresponding inclusive limiting constraint $h \leq 5$ thus should be somewhere in the evolution domain constraint.

$$\begin{aligned} &0 \leq h \land h \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow \\ &\quad \big[\big(h' = v, v' = -g \,\&\, h \geq 0 \land h \leq 5; \\ &\qquad \mathtt{if}(h = 0)\, v := -cv \;\mathtt{else}\; \mathtt{if}(4 \leq h \leq 5)\, v := -fv\big)^*\big](0 \leq h \leq 5) \end{aligned} \tag{5}$$

Is this the right model? Is dℒ formula (5) valid? Will its HP ensure that the critical event $4 \leq h \leq 5$ will not be missed out on?

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

Formula (5) is valid. And, yet, (5) is not at all the appropriate formula to consider. It is crucial to understand why.

So, formula (5) is valid. But why? Because all runs of the differential equation $h' = v, v' = -g \,\&\, h \geq 0 \land h \leq 5$ remain within the safety condition $0 \leq h \leq 5$ *by construction*. None of them are ever allowed to leave the region $h \geq 0 \land h \leq 5$, which, after all, is their evolution domain constraint. So formula (5) is trivially safe. A more careful argument involves that, every time around the loop, the postcondition holds trivially, because the differential equation's evolution constraint maintains it by definition, the subsequent discrete control never changes the only variable h on which the postcondition depends. Hold on, the loop does not have to run but could be skipped over by zero iterations as well. Yet, in that case, the precondition ensures the postcondition, so, indeed, (5) is valid, but trivially so.

Note 2 (Non-negotiability of Physics)**.** *Usually, it is a good idea to make systems safe by construction. For computer programs, that is a great idea. But we need to remember that physics is unpleasantly non-negotiable. So if the only reason why a CPS model is safe is because we forgot to model all relevant behavior of the real system, then correctness statements about those inadequate models are not particularly applicable to reality.*

One common cause for counterfactual models are too generous evolution domain constraints that rule out physically realistic behavior.

And that is what happened in (5). The bouncing ball got so carried away with trying not to miss the event $4 \leq h \leq 5$ that it forgot to include a behavior in the model that happens after the event has happened. The evolution domain constraint $\ldots \& \, h \geq 0$ came was in the system for physical reasons: to model the guaranteed bouncing back on the ground and to prevent the ball from falling through the ground. We added the evolution domain constraint $h \leq 5$ for an entirely different reason. It came into play to model what our controller does, and inaptly so, because our feeble attempt ruled out physical behavior that could actually have happened in reality.

Let's make up for this by developing a model that has both behaviors, just in different continuous programs so that the decisive event in the middle could not accidentally have been missed.

$$\begin{aligned} & 0 \leq h \land h \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow \\ & \quad \big[\big(((h' = v, v' = -g \,\&\, h \geq 0 \land h \leq 5) \cup (h' = v, v' = -g \,\&\, h \geq 5)); \\ & \quad\quad \texttt{if}(h = 0)\, v := -cv \,\texttt{else}\,\texttt{if}(4 \leq h \leq 5)\, v := -fv\big)^*\big](0 \leq h \leq 5) \end{aligned} \tag{6}$$

Now (6) has a much better model of events than the ill-advised (5). Is (6) valid?

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

When the ball is jumping up from the ground, the model in (6) makes it impossible for the controller to miss the event $4 \leq h \leq 5$, because the only evolution domain constraint in the HP that applies at the ground is $h \geq 0 \land h \leq 5$. And that evolution domain stops being true above 5. Yet, suppose the ping pong ball was jumping up from the ground following the continuous program in the left choice and then stopped its evolution at height $h = 4.5$, which always remains perfectly within the evolution domain $h \geq 0 \land h \leq 5$ and is, thus, allowed. Then, after the sequential composition between the middle and last line of (6), the controller in the last line of (6) runs, notices that the formula $4 \leq h \leq 5$ for the event checking is true, and changes the velocity according to $v := -fv$, corresponding to the assumed effect of a pat with the paddle. That is actually its only choice in such a state, because the controller is deterministic, much unlike the differential equation. Consequently, the velocity has just become negative since it was positive before as the ball was climbing up. So the loop can repeat and the differential equation runs again. Yet, then the differential equation might evolve until the ball is at height $h = 4.25$, which will happen since its velocity is negative. If the differential equation stops then, the controller will run again, determine that $4 \leq h \leq 5$ is true still and so take action to change the velocity to $v := -fv$. That will, however, make the velocity positive again, since it was previously negative as the ball was in the process of falling. Hence, the ball will keep on climbing now, which, again, threatens the postcondition $0 \leq h \leq 5$. Will this falsify (6) or is it valid?

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

On second thought, that alone still will not cause the postcondition to evaluate to $false$, because the only way the bouncing ball can evolve continuously from $h = 4.25$ is still by the continuous program in the left choice of (6). And that differential equation is restricted to the evolution domain $h \geq 0 \land h \leq 5$, which causes the controller to run before leaving $h \leq 5$. That is, the event $4 \leq h \leq 5$ will again be noticed by the controller so that the ball is ping pong paddle pats the ball back down.

However, the exact same reasoning applies also to the case where the ball successfully made it up to height $h = 5$, which is the height at which any climbing ball has to stop its continuous evolution, because it would otherwise violate the evolution domain $h \geq 0 \land h \leq 5$. As soon as that happens, the controller runs, notices that the event $4 \leq h \leq 5$ came true and reacts with a ping pong paddle to cause $v := -fv$. If, now, the loop repeats, yet the continuous evolution evolves for duration zero only, which is perfectly allowed, then the condition $4 \leq h \leq 5$ will still be true so that the controller again notices this "event" and reacts with ping pong paddle $v := -fv$. That will make the velocity positive, the loop can repeat, the continuous program on the right of the choice can be chosen since $h \geq 5$ holds true, and then the bouncing ball can climb and disappear into nothingness high up in the sky if only its velocity has been large enough.

Ergo, (6) is not valid. What a pity. And the bouncing ball would have to be afraid of heights when following the control in (6). How can this problem be resolved?

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

The problem in (6) is that its left differential equation makes sure never to miss out on the event $4 \leq h \leq 5$ but its control may react to it multiple times. It is not even sure whether each occasion of $4 \leq h \leq 5$ should be called an event. But certainly repeated reaction to the same event according to control (6) causes trouble.

One way of solving this problem is to change the condition in the controller to make sure it only reacts to the $4 \leq h \leq 5$ event when the ball is on its way up, i.e. when its velocity is not negative. That is what the bouncing ball wanted to ensure in any case. The ping pong paddle should only be actuated downwards when the ball is flying up.

These thoughts lead to the following variation:

$$\begin{aligned}
&0 \leq h \land h \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow \\
&\qquad \big[\big(((h' = v, v' = -g \,\&\, h \geq 0 \land h \leq 5) \cup (h' = v, v' = -g \,\&\, h \geq 5)); \\
&\qquad\quad \texttt{if}(h = 0)\, v := -cv \,\texttt{else}\,\texttt{if}(4 \leq h \leq 5 \land v \geq 0)\, v := -fv\big)^*\big](0 \leq h \leq 5)
\end{aligned} \tag{7}$$

Because the paddle action $v := -fv$ will disable the condition $v \geq 0$ for nonzero velocities, the controller in (7) can only react once to the event $4 \leq h \leq 5$ to turn the upwards velocity into a downwards velocity, scaled by f. Unlike in (6), this control decision cannot be reverted inadvertently by the controller.

Is $\mathsf{d}\mathcal{L}$ formula (7) valid?

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

In order to convince ourselves that the ping pong paddle control works as expected, we simplify the assumptions in formula (7) so that they match the ones in our prior proofs about bouncing balls in Lecture 7 on Control Loops & Invariants. Those additional assumptions are not all strictly necessary, but simplify the argument somewhat.

$$\begin{aligned}&0 \leq h \leq 5 \land v \leq 0 \land 1 \geq c \geq 0 \land g > 0 \land f \geq 0 \rightarrow \\ &\quad \big[\big(((h' = v, v' = -g \,\&\, h \geq 0 \land h \leq 5) \cup (h' = v, v' = -g \,\&\, h \geq 5)); \\ &\quad\quad \mathtt{if}\,(h = 0)\, v := -cv\; \mathtt{else}\; \mathtt{if}\,(4 \leq h \leq 5 \land v \geq 0)\, v := -fv\big)^*\big](0 \leq h \leq 5)\end{aligned} \tag{8}$$

How could d$\mathcal{L}$ formula (8) be proved? The most critical element of a proof is finding a suitable invariant. What could be the invariant for proving (8)?

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

The formula

$$5 \geq h \geq 0 \tag{9}$$

is an obvious candidate for an invariant. If it is true, it trivially implies the postcondition $0 \leq h \leq 5$ and it holds in the initial state. It is not inductive, though, because a state that satisfies (9) could follow the right differential equation if it satisfies $h \geq 5$. In that case, if the velocity is positive, the invariant (9) would be violated immediately. Hence, at the height $h = 5$, the control has to make sure that the velocity is negative, so that the right differential equation in (8) has to stop immediately. Could (9) be augmented with a conjunction $v \leq 0$? No that would not work either, because the bounce on the ground violates that invariant. In fact, the controller literally only ensures $v \leq 0$ at the event, which is detected at $h = 5$ at the latest. Indeed, the d$\mathcal{L}$ formula (7) can be proved in the d$\mathcal{L}$ calculus using the invariant

$$5 \geq h \geq 0 \land (h = 5 \rightarrow v \leq 0)$$

This invariant is just strong enough to remember the control choice at the event $h = 5$ and that the possible range of h is safe. Recall that (global) invariants need to be augmented with the usual assumptions about the unchanged variables, like $c \geq 0 \land g > 0 \land f \geq 0$.

The model that (8) and the other controllers in this section adhere to is called event-driven control or also event-driven architecture.

Note 3 (Event-driven control)**.** *One common paradigm for designing controllers is the event-driven architecture, in which the controller runs in response to certain events that happen in the system. The controller could possibly run under other circumstances as well—when in doubt, the controller simply skips over without any effect if it does not want to change anything about the behavior of the system. But event-driven controllers assume they will run for sure whenever certain events in the system happen.*

These events cannot be all too narrow, or else the system will not be implementable, though. For example, it is nearly impossible to build a controller that reacts exactly at the point in time when the height of the bouncing ball is $h = 4.12345$. Chances are high that any particular execution of the system will have missed this particular height. Care must be taken in event-driven design models also that the events do not inadvertently restrict the evolution of the system to the behavioral cases outside or after the events have happened. Those executions must still be verified.

Are we sure in model (8) that events are taken into account faithfully? That depends on what exactly we mean by an event like $4 \leq h \leq 5$. Do we mean that this event happens for the first time? Or do we mean every time this event happens? If multiple successive runs of the ping pong ball's controller see this condition satisfied, do these count as the same or separate instances of that event happening? Comparing the validity of (6) with the non-validity of (7) illustrates that these subtleties can have considerable impact on the system. Hence, a precise understanding of events and careful modeling is required.

The controller in (8) only takes an action for event $4 \leq h \leq 5$ when the ball is on the way up. Hence, the evolution domain constraint in the right continuous evolution is $h \geq 5$. Had we wanted to model the occurrence of event $4 \leq h \leq 5$ also when the ball is on its way down, then we would have to have a differential equation with evolution domain $h \geq 4$ to make sure the system does not miss $4 \leq h \leq 5$ when the ball is on its way down either, without imposing that it would have to notice $h = 5$ already. This could be achieved by splitting the evolution domain regions appropriately, but was not necessary for (8) since it never reacts to balls falling down, only those climbing up.

Note 4. *Events are a slippery slope and great care needs to be exercised to use them without introducing an inadequate executional bias into the model.*

There is a highly disciplined way of defining, detecting, and reacting to general events in differential dynamic logic based on the there and back again axiom [Pla12a]. That is, however, much more complicated than the simpler account shown here.

4 Delays in Control

Event-driven control is a useful and intuitive model matching our expectation of having controllers react in response to certain critical conditions or events that necessitate intervention by the controller. Yet, one of its difficulties is that event-driven control can be hard or impossible to implement in reality. On a higher level of abstraction, it is very intuitive to design controllers that react to certain events and change the control actuation in response to what events have happened. Closer to the implementation, this turns out to be difficult, because actual computer control algorithms do not actually run all the time, only sporadically every once in a while, albeit sometimes very often. Implementing event-driven control faithfully would, in principle, require permanent continuous monitoring of the state to check whether an event has happened. That is not quite realistic.

Back to the drawing desk. Let us reconsider the original d$\mathcal{L}$ formula (4) that we started out from for designing the event-driven version in (8).

$$\begin{aligned}&0 \leq h \land h \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow\\ &\quad\big[\big(h' = v, v' = -g \,\&\, h \geq 0;\\ &\qquad \texttt{if}(h = 0)\, v := -cv \,\texttt{else}\, \texttt{if}(4 \leq h \leq 5)\, v := -fv\big)^*\big](0 \leq h \leq 5)\end{aligned} \tag{4}$$

This simplistic formula (4) turned out not to be valid, because its differential equation was not guaranteed to be interrupted when the event $4 \leq h \leq 5$ happens. Consequently, (4) needs some other evolution domain constraint to make sure all continuous evolutions are stopped at some point for the control to have a chance to react to situation changes. Yet, it should not be something like $\ldots \,\&\, h \leq 5$ as in (8), because continuously monitoring for $h \leq 5$ requires permanent sensing of the height, which is difficult to implement.

How else could the continuous evolution of physics be interrupted to make sure the controller runs? By bounding the amount of time that physics is allowed to evolve before running the controller again. Before we can talk about time, the model needs to be changed to include a variable, say t, that reflects the progress of time with a differential equation $t' = 1$.

$$\begin{aligned} &0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 > 0 \land 1 \geq c \geq 0 \land f > 0 \to \\ &\quad \big[\big(t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0 \land t \leq 1; \\ &\quad\quad \texttt{if}(h = 0)\, v := -cv \,\texttt{else}\,\texttt{if}(4 \leq h \leq 5)\, v := -fv\big)^*\big](0 \leq h \leq 5) \end{aligned} \tag{10}$$

In order to bound time by 1, the evolution domain now includes $\ldots \,\&\, t \leq 1$ and the variable t is reset to 0 by $t := 0$ right before the differential equation. Hence, t represents a local clock measuring how long the evolution of the differential equation was. Its bound of 1 ensures that physics gives the controller a chance to react at least once per second. The system could very well stop the continuous evolution more often and earlier, because there is no lower bound on t in (10). Also see Exercise 1.

Before going any further, let's take a step back to notice an annoyance in the way the control in (10) was written. It is written in the style that the original bouncing ball and the event-driven ping pong ball were phrased: continuous dynamics followed by control. That has the unfortunate effect that (10) lets physics happen before control does anything, which is not a very safe start. In other words, the initial condition would have to be modified to assume the initial control was fine. That is a nuisance duplicating part of the control into the assumptions on the initial state. Instead, let's switch the statements around to make sure control always happens before physics.

$$\begin{aligned} &0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 > 0 \land 1 \geq c \geq 0 \land f > 0 \to \\ &\quad \big[\big(\texttt{if}(h = 0)\, v := -cv \,\texttt{else}\,\texttt{if}(4 \leq h \leq 5)\, v := -fv; \\ &\quad\quad t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0 \land t \leq 1\big)^*\big](0 \leq h \leq 5) \end{aligned} \tag{11}$$

Now that d$\mathcal{L}$ formula (11) has an upper bound on the time it takes between two subsequent control actions, is it valid?

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

Even though (11) ensures a bound on how long it may take at most until the controller inspects the state and reacts, there is still a fundamental issue with (11). We can try to prove (11) and inspect the non-provable cases in the proof to find out what the issue is. The controller of (11) runs at least after one second (hence at least once per second) and then checks whether $4 \leq h \leq 5$. But if $4 \leq h \leq 5$ was not true when the controller ran last, there is no guarantee that it will be true when the controller runs next. In fact, the ball might very well have been at $h = 3$ at the last controller run, then evolved continuously to $h = 6$ within a second and so missed the event $4 \leq h \leq 5$ that it was supposed to detect (Exercise 2). Worse than that, the ping pong ball has then already become unsafe.

For illustration, driving a car would be similarly unsafe if you would only open your eyes once a second and monitor whether there is a car right in front of you. Too many things could have happened in between that should have prompted you to brake.

Note 5 (Delays may miss events)**.** *Delays in controller reactions may cause events to be missed that they were supposed to monitor. When that happens, there is a discrepancy between an event-driven understanding of a CPS and the real time-triggered implementation. That happens especially for slow controllers monitoring small regions of a fast moving system. This relationship deserves special attention to make sure the impact of delays on a system controller cannot make it unsafe.*

It is often a good idea to first understand and verify an event-driven design of a CPS controller and then refine it to a time-triggered controller to analyze and verify that CPS in light of its reaction time. Discrepancies in this analysis hint at problems that event-driven designs will likely experience at runtime and they indicate a poor event abstraction.

How can this problem of (11) be solved? How can the CPS model make sure the controller does not miss its time to take action? Waiting until $4 \leq h \leq 5$ holds true is not guaranteed to be the right course of action for the controller.

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

The problem with (11) is that its controller is unaware of its own delay. It does not take into account how the ping pong ball could have moved further before it gets a chance to react next. If the ball is already close to the ping pong paddle's intended range of actuation, then the controller had better take action already if it is not sure whether next time will still be fine.

The controller would be in trouble if, in its next control cycle after the continuous evolution, $h > 5$. The continuous evolution can take at most 1 time unit, after which the ball will be at position $h + v - \frac{g}{2}$ as we have observed in Lecture 4 by solving the differential equation. We chose $g = 1$ for the time-triggered case, so the controller could be in trouble in the next control cycle if $h > 5\frac{1}{2} - v$ holds now. Hence, the idea is to make the controller now act based on how it estimates the state might have evolved until the next control cycle. The difference of (6) vs. (7) in the event-driven case indicates that the controller only wants to trigger action if the ball is flying up. Thus, making (11) aware of the future in this way leads to:

$$\begin{aligned}&0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow \\ &\quad \big[\big(\mathtt{if}(h = 0)\, v := -cv\, \mathtt{else}\, \mathtt{if}((h > 5\frac{1}{2} - v) \land v \geq 0)\, v := -fv; \\ &\qquad t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0 \land t \leq 1\big)^*\big](0 \leq h \leq 5)\end{aligned} \tag{12}$$

Is conjecture (12) about its future-aware controller valid?

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

The controller in formula (12) has been designed based on the prediction that the future may evolve for 1 time unit. If action will no longer be possible in 1 time unit, because the event $h \leq 5$ has passed in that future instant, the controller in (12) takes action right now already. The issue with that is there is no guarantee that the ping pong ball will fly for exactly 1 time unit before the controller is asked to act again (and the postcondition is checked). The controller in (12) checks whether the ping pong ball could be too far up after one time unit and does not intervene unless that is the case. Yet, what if the ball flies for $\frac{1}{2}$ time units? Clearly, if the ball will be safe after 1 time unit, which is what the controller in (12) checks, it will also be save after just $\frac{1}{2}$ time unit, right?.

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

Wrong! The ball may well be below 5 after 1 time unit but still could have been above 5 in between the current point of time and 1 time unit from now. Recall Fig. 1 to see how this can happen.

In order to understand this further, we use the invariant that we have derived for the bouncing ball in an earlier lecture and then used in Lecture 7 to prove (1).

$$2gh = 2gH - v^2 \land h \geq 0 \land c = 1 \land g > 0 \tag{13}$$

We assume this invariant to hold in the beginning of the ping pong ball's life and also adopt the global assumptions $c = 1 \land g = 1 \land f = 1$ to simplify the arithmetic.

Substituting the critical height 5 for H in (13) for this instance of parameter choices leads to the following condition which indicates that the ball could end up climbing too high

$$2h > 2 \cdot 5 - v^2 \tag{14}$$

Adding this condition to the controller (12) leads to:

$$\begin{aligned}2h = 2H - v^2 \land 0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 > 0 \land 1 = c \geq 0 \land 1 = f > 0 \to\\ \big[\big(\texttt{if}\,(h = 0)\, v := -cv\, \texttt{else}\, \texttt{if}((h > 5\frac{1}{2} - v \lor 2h > 2 \cdot 5 - v^2) \land v \geq 0)\, v := -fv;\\ t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0 \land t \leq 1\big)^*\big](0 \leq h \leq 5)\end{aligned} \tag{15}$$

Is d$\mathcal{L}$ formula (15) about its time-triggered controller valid?

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

Formula (15) is almost valid. But it is still not valid for a very subtle reason. It is great to have proof to catch those subtle issues. The controller in (15) takes action for two different conditions on the height h. However, the ping pong paddle controller actually only runs in (15) if the ball is not at height $h = 0$, for otherwise ground control takes action of reversing the direction of the ball. Now, if the ball is flat on the floor ($h = 0$) yet its velocity so incredibly high that it will rush past height 5 in less than 1 time unit, then the ping pong paddle controller will not have had a chance to react before it is too late, because it does not run on the ground according to (15).

Fortunately, these thoughts already indicate how that problem can be fixed. By turning the nested `if-then-else` cascade into a sequential compositions of `if-then` that will ensure the ping pong paddle controller to run for sure.

$$\begin{aligned}&2h = 2H - v^2 \land 0 \le h \land h \le 5 \land v \le 0 \land g = 1 > 0 \land 1 = c \ge 0 \land 1 = f > 0 \to\\ &\quad \big[\big(\texttt{if}(h=0)\, v := -cv; \texttt{if}((h > 5\frac{1}{2} - v \lor 2h > 2\cdot 5 - v^2) \land v \ge 0)\, v := -fv;\\ &\quad\; t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \ge 0 \land t \le 1\big)^*\big](0 \le h \le 5)\end{aligned} \tag{16}$$

Now, is formula (16) finally valid, please?

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

Yes, formula (16) is valid and can be proved with the invariant

$$2h = 2H - v^2 \land h \geq 0 \land h \leq 5 \tag{17}$$

Yet, is the controller in (16) useful? That is where the problem lies now. The condition (14) checks whether the ping pong ball could possibly ever fly up to height 5. If this is ever true, it might be true long before the bouncing ball approaches the critical control cycle where ping pong paddle action needs to be taken. In fact, if (14) is ever true, it will also be true in the beginning. After all, the formula (13), from which (14) derived, is an invariant. That would cause the controller in (16) to take action right away even if the ping pong ball is still close to the ground and far away from height 5. That would make the ping pong ball safe, after all (16) is valid, but also rather conservative, and would not allow the ping pong ball to bounce around as much as it would have loved to. How can the controller in (16) be modified to resolve this problem?

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

Restrict the use of condition Exercise 1 to slow velocities to only make up for the occasions that the first controller condition $h > 5\frac{1}{2} - v$ misses. Only with slow velocities $(v < 1)$ does the ball move so slowly that it is near its turning point to start falling down, and only then could the first condition miss out on the ball being able to have evolve above 5 before 1 time unit.

$$\begin{aligned}&2h = 2H - v^2 \land 0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 > 0 \land 1 = c \geq 0 \land 1 = f > 0 \rightarrow\\ &\quad \big[\big(\mathtt{if}(h = 0)\, v := -cv; \mathtt{if}((h > 5\frac{1}{2} - v \lor 2h > 2 \cdot 5 - v^2 \land v < 1) \land v \geq 0)\, v := -fv;\\ &\quad\; t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0 \land t \leq 1\big)^*\big](0 \leq h \leq 5)\end{aligned} \tag{18}$$

This d$\mathcal{L}$ formula is valid and provable with the same invariant (17) that was used to prove (16). It has a much more aggressive controller than (16), though, so it is more fun for the ping pong ball to bounce around with it. Recall that (global) invariants need to be augmented with the usual assumptions about the unchanged variables, like $g = 1 \land 1 = c \land 1 = f$.

Note 6 (Time-triggered control)**.** *One common paradigm for designing controllers is time-triggered control, in which controllers run periodically or pseudo-periodically with certain frequencies to inspect the state of the system. Time-triggered systems are closer to implementation than event-driven control. They can be harder to build, however, because they invariably require the designer to understand the impact of delay on control decisions. That impact is important in reality, however, and, thus, effort invested in understanding the impact of time delays usually pays off in designing a safer system that is robust to bounded time delays.*

Exercises

Exercise 1. The HP in (11) imposes an upper bound on the duration of a continuous evolution. How can you impose an upper bound 1 and a lower bound 0.5?

Exercise 2. Give an initial state for which the controller in (11) would skip over the event without noticing it.

Exercise 3. The formula (18) with the time-triggered controller of reaction time at most 1 time unit is valid. Yet, if a ball is let loose a wee bit above ground with a very fast negative velocity, couldn't it possibly bounce back and exceed the safe height 5 faster than the reaction time of 1 time unit? Does that mean the formula ought to have been falsifiable? No! Identify why and give a physical interpretation.

Exercise 4. The event-driven controller we designed in Sect. 3 monitored the event $4 \leq h \leq 5$. The time-triggered controller in Sect. 4, however, ultimately only took the upper bound 5 into account. How and under which circumstances can you modify the controller so that it really only reacts for the event $4 \leq h \leq 5$.

Exercise 5. Devise a controller that reacts if the height changes by 1 when comparing the height before the continuous evolution to the height after. Can you make it safe? Can you implement it? Is it an event-driven or a time-triggered controller? How does it compare to the controllers developed in this lecture?

References

[Pla10] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14509-4.

[Pla12a] André Platzer. The complete proof theory of hybrid systems. In *LICS*, pages 541–550. IEEE, 2012. doi:10.1109/LICS.2012.64.

[Pla12b] André Platzer. Dynamic logics of dynamical systems. *CoRR*, abs/1205.4788, 2012. arXiv:1205.4788.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Proofs & Arithmetic

André Platzer

Carnegie Mellon University

Lecture 9

1 Introduction

Lecture 8 on Events & Delays discussed and developed two models for the ping pong ball whose control was a ping pong paddle. First an event-driven controller and then a time-triggered controller. Invariants have been identified in Lecture 8 but not proved. This lecture will study proofs.

This lecture discusses:

- What is a proof?
- How does arithmetic happen in a proof?
- Arithmetic hiding to weaken arithmetic
- Instantiating arithmetic quantifiers to the extreme
- Updates as delayed substitutions/assignments
- Creative cuts for arithmetic
- Substituting equations
- If-then-else proofs

There are many more interesting things to be discussed about the mysteries of arithmetic and how real arithmetic properties themselves can be proved [PQR09, Pla10, Appendix D]. That is a topic for a later lecture, though. This lecture focuses on how arithmetic interfaces with proofs.

2 Proving Events in Control

$$\begin{aligned}&0 \leq h \leq 5 \land v \leq 0 \land 1 \geq c \geq 0 \land g > 0 \land f \geq 0 \rightarrow \\ &\quad\big[\big(((h' = v, v' = -g \,\&\, h \geq 0 \land h \leq 5) \cup (h' = v, v' = -g \,\&\, h \geq 5)); \\ &\qquad \texttt{if}(h = 0)\, v := -cv \,\texttt{else}\, \texttt{if}(4 \leq h \leq 5 \land v \geq 0)\, v := -fv\big)^*\big](0 \leq h \leq 5)\end{aligned} \tag{1}$$

Lecture 8 on Events & Delays identified the following invariant for this system:

$$5 \geq h \geq 0 \land (h = 5 \rightarrow v \leq 0)$$

This invariant is just strong enough to remember the control choice at the event $h = 5$ and that the possible range of h is safe. Recall that (global) invariants need to be augmented with the usual assumptions about the unchanged variables, like $c \geq 0 \land g > 0 \land f \geq 0$.

$$\varphi \overset{\text{def}}{\equiv} 0 \leq h \leq 5 \land (h = 5 \rightarrow v \leq 0) \land 1 \geq c \geq 0 \land g > 0 \land f \geq 0$$

Let's use some (slightly awkward) abbreviations to keep proofs onto one page.

$$\begin{aligned}A_{h,v} &\overset{\text{def}}{\equiv} 2h = 2H - v^2 \land 0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 \land 1 = c \land 1 = f \\ B_{h,v} &\overset{\text{def}}{\equiv} 0 \leq h \land h \leq H \\ h''{=}..{\leq}5 &\overset{\text{def}}{\equiv} (h' = v, v' = -g \,\&\, h \geq 0 \land h \leq 5) \\ h''{=}..{\geq}5 &\overset{\text{def}}{\equiv} (h' = v, v' = -g \,\&\, h \geq 5) \\ \texttt{if}(h{=}0)\,.. &\overset{\text{def}}{\equiv} \texttt{if}(h = 0)\, v := -cv \,\texttt{else} \\ \texttt{if}(4, h{\leq}5)\,.. &\overset{\text{def}}{\equiv} \texttt{if}(4 \leq h \leq 5 \land v \geq 0)\, v := -fv\end{aligned}$$

With these abbreviations, the event-driven ping pong ball formula (1) turns into:

$$A_{h,v} \rightarrow [\big((h''{=}..{\leq}5 \cup h''{=}..{\geq}5); \texttt{if}(h{=}0)\,..\texttt{if}(4, h{\leq}5)\,..\big)^*]B_{h,v}$$

Let's set out to prove (1) by converting it into a sequent and applying d$\mathcal{L}$ proof rules:

$$\text{ind}'\,\dfrac{\text{$\land$l,$\land$r},ax\,\dfrac{*}{A_{h,v} \vdash \varphi} \quad \varphi \vdash [(h''{=}..{\leq}5 \cup h''{=}..{\geq}5); \texttt{if}(h{=}0)\,..\texttt{if}(4, h{\leq}5)\,..]\varphi \quad \text{$\land$l,$\land$r},ax\,\dfrac{*}{\varphi \vdash B_{h,v}}}{\rightarrow\text{r}\,\dfrac{A_{h,v} \vdash [\big((h''{=}..{\leq}5 \cup h''{=}..{\geq}5); \texttt{if}(h{=}0)\,..\texttt{if}(4, h{\leq}5)\,..\big)^*]B_{h,v}}{\vdash A_{h,v} \rightarrow [\big((h''{=}..{\leq}5 \cup h''{=}..{\geq}5); \texttt{if}(h{=}0)\,..\texttt{if}(4, h{\leq}5)\,..\big)^*]B_{h,v}}}$$

The left premise (initial case) and the right premise (use case) prove directly by splitting the conjunctions with ∧l,∧r and then closing by axiom ax. The middle premise (inductive step for preserving the invariant) requires more work:

$$
{\scriptstyle [;]\mathrm{r}}\,\cfrac{{\scriptstyle [\cup]\mathrm{r}}\,\cfrac{\cfrac{\dots}{\varphi \vdash [h''{=}..{\le}5][\mathtt{if}(h{=}0)\,..\mathtt{if}(4,h{\le}5)\,..]\varphi} \qquad \cfrac{\dots}{\varphi \vdash [h''{=}..{\ge}5][\mathtt{if}(h{=}0)\,..\mathtt{if}(4,h{\le}5)\,..]\varphi}}{\varphi \vdash [h''{=}..{\le}5 \cup h''{=}..{\ge}5][\mathtt{if}(h{=}0)\,..\mathtt{if}(4,h{\le}5)\,..]\varphi}}{\varphi \vdash [(h''{=}..{\le}5 \cup h''{=}..{\ge}5);\mathtt{if}(h{=}0)\,..\mathtt{if}(4,h{\le}5)\,..]\varphi}
$$

The right premise will be considered later. The left premise needs the solution of the differential equation

$$
h := ..(t) \overset{\text{def}}{\equiv} (h := h - \frac{g}{2}t^2 - vt;\ v := v - gt) \tag{2}
$$

The left premise continues as follows:

$$
{\scriptstyle [']\mathrm{r}}\,\cfrac{{\scriptstyle \forall\mathrm{r}}\,\cfrac{{\scriptstyle \rightarrow\mathrm{r}}\,\cfrac{{\scriptstyle \rightarrow\mathrm{r}}\,\cfrac{{\scriptstyle \forall\mathrm{l}}\,\cfrac{\varphi, \mathfrak{t}{\ge}0, 0{\le}\mathfrak{t}{\le}\mathfrak{t} \rightarrow [h := ..(s)](h \ge 0 \land h \le 5) \vdash [h := ..(\mathfrak{t})][\mathtt{if}(h{=}0)\,..\mathtt{if}(4,h{\le}5)\,..]\varphi}{\varphi, \mathfrak{t}{\ge}0, \forall 0{\le}s{\le}\mathfrak{t}\,[h := ..(s)](h \ge 0 \land h \le 5) \vdash [h := ..(\mathfrak{t})][\mathtt{if}(h{=}0)\,..\mathtt{if}(4,h{\le}5)\,..]\varphi}}{\varphi, \mathfrak{t}{\ge}0 \vdash \forall 0{\le}s{\le}\mathfrak{t}\,[h := ..(s)](h \ge 0 \land h \le 5) \rightarrow [h := ..(\mathfrak{t})][\mathtt{if}(h{=}0)\,..\mathtt{if}(4,h{\le}5)\,..]\varphi}}{\varphi \vdash \mathfrak{t}{\ge}0 \rightarrow \big(\forall 0{\le}s{\le}\mathfrak{t}\,[h := ..(s)](h \ge 0 \land h \le 5) \rightarrow [h := ..(\mathfrak{t})][\mathtt{if}(h{=}0)\,..\mathtt{if}(4,h{\le}5)\,..]\varphi\big)}}{\varphi \vdash \forall t{\ge}0\,\big(\forall 0{\le}s{\le}t\,[h := ..(s)](h \ge 0 \land h \le 5) \rightarrow [h := ..(t)][\mathtt{if}(h{=}0)\,..\mathtt{if}(4,h{\le}5)\,..]\varphi\big)}}{\varphi \vdash [h''{=}..{\le}5][\mathtt{if}(h{=}0)\,..\mathtt{if}(4,h{\le}5)\,..]\varphi}
$$

The top-most step instantiates the universal quantifier $\forall s$ in the antecedent by a smart choice. That formula in the antecedent expresses that the evolution domain $h \ge 0 \land h \le 5$ holds at all times s between 0 and the duration $\mathfrak{t}$ of the continuous evolution. That may very well be true, but what our thinking actually only depends on is that the evolution domain still holds at the end time, $\mathfrak{t}$ of the continuous evolution. The fact that the evolution domain was also true before is not so crucial for our argument here, so we simply instantiate the universally quantifier variable s in the antecedent by the time endpoint $\mathfrak{t}$ using rule $\forall$l.

Note 1 (Extreme instantiation). *The proof rule ∀l for universal quantifiers in the antecedent as well as the rule ∃r for existential quantifiers in the succedent allow instantiation of the quantified variable x with any term θ.*

$$(\forall\mathrm{l})\ \dfrac{\Gamma, \phi(\theta), \forall x\, \phi(x) \vdash \Delta}{\Gamma, \forall x\, \phi(x) \vdash \Delta}\ ^{a}$$

The way this rule is used in KeYmaera is with a direct use of weakening rule Wl to hide the quantified formula:

$$(\forall\mathrm{l})\ \dfrac{\Gamma, \phi(\theta) \vdash \Delta}{\Gamma, \forall x\, \phi(x) \vdash \Delta}\ ^{b}$$

This instantiation is very helpful if only a single instance θ is important for the argument. Often, an extremal value for x is all it takes for the proof.

This happens often for quantifiers coming from the handling of evolution domains in proof rule [′]r. The proof steps that often help then is instantiation of intermediate time s by the end time $\mathfrak{t}$:

$$\scriptstyle\forall r\ \dfrac{\scriptstyle\to r\ \dfrac{\scriptstyle\to r\ \dfrac{\scriptstyle\forall l\ \dfrac{\scriptstyle\to l\ \dfrac{\dfrac{*}{\Gamma, \mathfrak{t}\geq 0 \vdash 0\leq\mathfrak{t}\leq\mathfrak{t}, [x := y(\mathfrak{t})]\phi} \qquad \dfrac{\dots}{\Gamma, \mathfrak{t}\geq 0, [x := y(\mathfrak{t})]H \vdash [x := y(\mathfrak{t})]\phi}}{\Gamma, \mathfrak{t}\geq 0, 0\leq\mathfrak{t}\leq\mathfrak{t} \to [x := y(\mathfrak{t})]H \vdash [x := y(\mathfrak{t})]\phi}}{\Gamma, \mathfrak{t}\geq 0, \forall 0\leq s\leq\mathfrak{t}\, [x := y(s)]H \vdash [x := y(\mathfrak{t})]\phi}}{\Gamma, \mathfrak{t}\geq 0 \vdash (\forall 0\leq s\leq\mathfrak{t}\, [x := y(s)]H) \to [x := y(\mathfrak{t})]\phi}}{\Gamma \vdash \mathfrak{t}\geq 0 \to \big((\forall 0\leq s\leq\mathfrak{t}\, [x := y(s)]H) \to [x := y(\mathfrak{t})]\phi\big)}}{\Gamma \vdash \forall t\geq 0\, \big((\forall 0\leq s\leq t\, [x := y(s)]H) \to [x := y(t)]\phi\big)}$$

Similar instantiations can simplify arithmetic in other cases as well.

[a] θ is an arbitrary term, often a new (existential) logical variable X.
[b] θ is an arbitrary term, often a new (existential) logical variable X.

Continuing the above proof as explained in Note 1 recalls that $h := ..(\mathfrak{t})$ abbreviates the solution (2) and leads to:

$$\dfrac{\scriptstyle[:=]\mathrm{l}\ \dfrac{\scriptstyle\wedge\mathrm{l}\ \dfrac{\varphi, \mathfrak{t}\geq 0, h - \frac{g}{2}\mathfrak{t}^2 - v\mathfrak{t} \geq 0, h - \frac{g}{2}\mathfrak{t}^2 - v\mathfrak{t} \leq 5 \vdash [h := ..(\mathfrak{t})][\mathtt{if}(h{=}0)\, ..\mathtt{if}(4, h{\leq}5)\, ..]\varphi}{\varphi, \mathfrak{t}\geq 0, h - \frac{g}{2}\mathfrak{t}^2 - v\mathfrak{t} \geq 0 \wedge h - \frac{g}{2}\mathfrak{t}^2 - v\mathfrak{t} \leq 5 \vdash [h := ..(\mathfrak{t})][\mathtt{if}(h{=}0)\, ..\mathtt{if}(4, h{\leq}5)\, ..]\varphi}}{\varphi, \mathfrak{t}\geq 0, [h := ..(s)](h\geq 0 \wedge h \leq 5) \vdash [h := ..(\mathfrak{t})][\mathtt{if}(h{=}0)\, ..\mathtt{if}(4, h{\leq}5)\, ..]\varphi}}{\varphi, \mathfrak{t}\geq 0, 0\leq\mathfrak{t}\leq\mathfrak{t} \to [h := ..(s)](h\geq 0 \wedge h \leq 5) \vdash [h := ..(\mathfrak{t})][\mathtt{if}(h{=}0)\, ..\mathtt{if}(4, h{\leq}5)\, ..]\varphi}$$

This formula has gotten a bit lengthy, so abbreviate[1] $h - \frac{g}{2}\mathfrak{t}^2 - v\mathfrak{t}$ by $\hat{h}$ and abbreviate $v - gt$ by $\hat{v}$. Hence,

$$h := ..(\mathfrak{t}) \equiv (h := \hat{h}; v := \hat{v})$$

But there also is a problem that we have not noticed before. Which proof rule do we apply next? Sequent proof rules insist on being applied only to formulas on the top level of the sequent, i.e. directly as a formula of the antecedent or directly to a formula

[1] Abbreviating long terms or long formulas by short names can help simplify KeYmaera proofs as well.

in the succedent. Except for splitting conjunctions in φ by $\wedge$l, the only other formula to apply a proof rule to is the single formula in the succedent, which has a $[\cdot]$ modality with an assignment as the top-level operator. Thus, the only proof rule that applies is [:=]r. Rule [:=]r substitutes the right-hand side θ of an assignment for the variable x assigned to.

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

For simple arithmetic and propositional formulas, it is obvious what such a substitution does. It just replaces x by θ everywhere in the scope of the substitution. That is exactly what we have done with $h := \hat{h}$ when using rule [:=]l in the antecedent in the above proof.

Yet, the above formula

$$[h := ..(\mathtt{t})][\mathtt{if}(h{=}0)\,..\mathtt{if}(4, h{\le}5)\,..]\varphi$$

in the succedent has a postcondition $[\mathtt{if}(h{=}0)\,..\mathtt{if}(4, h{\le}5)\,..]\varphi$ with a modality. It is not necessarily entirely obvious how to substitute $\hat{h}$ for h in such a modality which involves a HP. In this particular case, we could actually perform such a substitution without much difficulty.

Even though such *substitutions* can be defined [Pla10, Chapter 2.5.1] with a little bit of care, we usually stay away from using them.[2]

[2] KeYmaera would even need to be persuaded to use these substitutions on HPs at all by setting the advanced option *update modalities*.

Note 2 (Excursion: Updates). *For that reason, KeYmaera simply 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 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*.*

$$\text{(R4)}\ \frac{\{x := \theta\}\phi}{[x := \theta]\phi} \qquad\qquad \text{(R5)}\ \frac{\phi_x^\theta}{\{x := \theta\}\phi}$$

The meaning of the formula $\{x := \theta\}\phi$ in the premise of R4 is exactly the same as the formula $[x := \theta]\phi$ in the conclusion of R4. 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 $\{x := \theta\}$ around with it and apply the sequent proof rules directly to postcondition ϕ until the substitution can be applied (R5) which will make the update disappear again. Thus, KeYmaera splits the assignment rule [:=] into two parts: R4 followed by R5.

More information on updates can be found in [Pla08, Pla10, Chapter 2.2,2.3,2.5].

Recall that we use the abbreviated notation $\hat{h}$ and $\hat{v}$ and, hence $h :=$ ..(t) is just $h := \hat{h}; v := \hat{v}$.

After using rule R4 to changing the assignment notation into an update notation (remember that this only changes notation because both are equivalent) the above sequent reads

$$\varphi, \mathrm{t}{\geq}0, \hat{h} \geq 0, \hat{h} \leq 5 \vdash \{h := \hat{h}; v := \hat{v}\}[\mathtt{if}(h{=}0)\,..\mathtt{if}(4, h{\leq}5)\,..]\varphi \tag{3}$$

Before proceeding with any proof, we need to figure out what to do with the `if-then-else` statements. Before doing any proofs, previous lectures, replaced `if-then-else` statements by other hybrid program statements, which is always possible. In this lecture, we decide differently and develop a direct proof rule for `if-then-else`.

$$(\langle\text{if}\rangle)\ \frac{(H \to \langle\alpha\rangle\phi) \land (\neg H \to \langle\beta\rangle\phi)}{\langle\mathtt{if}(H)\,\alpha\,\mathtt{else}\,\beta\rangle\phi} \qquad ([\text{if}])\ \frac{(H \to [\alpha]\phi) \land (\neg H \to [\beta]\phi)}{[\mathtt{if}(H)\,\alpha\,\mathtt{else}\,\beta]\phi}$$

When following up on a use of the [if] rule in the succedent of a sequent (call the corresponding sequent rule [if]r) with propositional rules ∧r,→r, the sequent splits into two cases as expected:[3]

[3] These propositional steps following the [if]r rule are so useful that KeYmaera does them for you right away. In fact, KeYmaera even jumps from the formula at the bottom directly to the two premises.

$$\text{[if]r}\;\dfrac{\text{∧r}\;\dfrac{\text{→r}\;\dfrac{\Gamma, H \vdash [\alpha]\phi}{\Gamma \vdash H \to [\alpha]\phi} \qquad \text{→r}\;\dfrac{\Gamma, \neg H \vdash [\beta]\phi}{\Gamma \vdash \neg H \to [\beta]\phi}}{\Gamma \vdash (H \to [\alpha]\phi) \land (\neg H \to [\beta]\phi)}}{\Gamma \vdash [\mathtt{if}(H)\,\alpha\,\mathtt{else}\,\beta]\phi, \Delta}$$

Indeed, the conjecture at the bottom says that we want to show that all behavior of a system whose behavior branches by an `if-then-else` is safe (satisfies ϕ). We do not know which state we are in, except that we get to assume it satisfies Γ (and the negation of Δ by the sequent semantics). So there are usually many possible states. Hence, there is generally no way of knowing whether if-condition H evaluates to $true$ or $false$. Hence, we need to consider both options. If H evaluates to $true$, then α runs, so all α behavior needs to be shown to be safe in that case (left premise). If H evaluates to $false$, then β runs instead, so all β behavior needs to be shown to be safe (right premise).

Applying the [if]r rule two times to the sequent (3) yields 3 premises corresponding to the 3 possible outcomes of the `if-then-else` statements (Exercise 1):

$$\begin{aligned}
&\varphi, \mathrm{t}{\geq}0, \hat{h} \geq 0, \hat{h} \leq 5, \hat{h} = 0 \vdash \{h := \hat{h}; v := \hat{v}\}[v := -cv]\varphi \\
&\varphi, \mathrm{t}{\geq}0, \hat{h} \geq 0, \hat{h} \leq 5, \hat{h} \neq 0, 4 \leq \hat{h} \leq 5 \land \hat{v} \geq 0 \vdash \{h := \hat{h}; v := \hat{v}\}[v := -fv]\varphi \\
&\varphi, \mathrm{t}{\geq}0, \hat{h} \geq 0, \hat{h} \leq 5, \hat{h} \neq 0, \neg(4 \leq \hat{h} \leq 5 \land \hat{v} \geq 0) \vdash \{h := \hat{h}; v := \hat{v}\}\varphi
\end{aligned} \tag{4}$$

Let's address the three branches separately. The first branch of (4) turns into the following using either [:=]r or R4 via R5:

$$\varphi, \mathrm{t}{\geq}0, \hat{h} \geq 0, \hat{h} \leq 5, \hat{h} = 0 \vdash \{h := \hat{h}; v := -c\hat{v}\}\varphi$$

which gives the following by applying the update using R5 (can also be obtained directly by [:=]r):

$$\varphi, \mathrm{t}{\geq}0, \hat{h} \geq 0, \hat{h} \leq 5, \hat{h} = 0 \vdash 0 \leq \hat{h} \leq 5 \land (\hat{h} = 5 \to -c\hat{v} \leq 0) \land 1 \geq c \geq 0 \land g > 0 \land f \geq 0$$

That proves by arithmetic, because $\hat{h} = 0$ and implies $0 \leq \hat{h} \leq 5$ and $\hat{h} \neq 5$ and the other parts prove similarly.[4]

The second branch of (4) turns by either [:=] or via R4 and R5 into:

$$\varphi, \mathrm{t}{\geq}0, \hat{h} \geq 0, \hat{h} \leq 5, \hat{h} \neq 0, 4 \leq \hat{h} \leq 5 \land \hat{v} \geq 0 \vdash \{h := \hat{h}; v := -f\hat{v}\}\varphi$$

which R5 turns into

$$\varphi, \mathrm{t}{\geq}0, \hat{h} \geq 0, \hat{h} \leq 5, \hat{h} \neq 0, 4 \leq \hat{h} \leq 5 \land \hat{v} \geq 0 \vdash 0 \leq \hat{h} \leq 5 \land (\hat{h} = 5 \to -f\hat{v} \leq 0) \land 1 \geq c \geq 0 \land g > 0 \land f \geq 0$$

which proves by arithmetic using that $f \geq 0$ and $\hat{v} \geq 0$ as well as the fact that $4 \leq \hat{h} \leq 5$ trivially implies $0 \leq \hat{h} \leq 5$, which is obvious thanks to the abbreviations.

The third branch of (4) turns with [:=]r or R5 into:

$$\varphi, \mathrm{t}{\geq}0, \hat{h} \geq 0, \hat{h} \leq 5, \hat{h} \neq 0, \neg(4 \leq \hat{h} \leq 5 \land \hat{v} \geq 0) \vdash 0 \leq \hat{h} \leq 5 \land (\hat{h} = 5 \to \hat{v} \leq 0) \land 1 \geq c \geq 0 \land g > 0 \land f \geq 0$$

[4] Note how abbreviations simplify this proof step compared to what would have happened when expanding $\hat{h}$.

which a combination of propositional rules and/or arithmetic proves (Exercise!)

All this reasoning was for just the branch of the proof that came from the dynamics $h''=..\leq 5$. There is a second branch with the dynamics $h''=..\geq 5$. In that one, the proof is quite similar, except that it makes crucial use of the conjunct $h = 5 \to v \leq 0$ of the invariant φ. Without that condition available as an assumption from the invariant φ, the upper physics $h''=..\geq 5$ would obviously violate the safety condition $0 \leq h \leq 5$ if the velocity at $h = 5$ were positive $v > 0$.

Stepping back, it is crucial to observe this general phenomenon. We have to be able to assume the turning-point part $h = 5 \to v \leq 0$ of invariant φ for the proof of the upper dynamics $h''=..\geq 5$. But we also need to prove that this turning-point invariant $h = 5 \to v \leq 0$ holds along with the rest of the invariant φ after all runs of the lower physics $h''=..\leq 5$. That is, this part of the invariant φ transports knowledge about the behavior of the controller in the lower physics $h''=..\leq 5$ to be used in the proof parts about the upper physics $h''=..\geq 5$.

Note 3 (Invariants transport knowledge)**.** *Invariants can be used to gather knowledge about the individual bits and pieces of a system and make them accessible to the other parts.*

3 Proving Systems with Delays in Control

$$2h = 2H - v^2 \land 0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 > 0 \land 1 = c \geq 0 \land 1 = f > 0 \rightarrow$$
$$\big[\big(\texttt{if}(h = 0)\, v := -cv; \texttt{if}((h > 5\frac{1}{2} - v \lor 2h > 2 \cdot 5 - v^2 \land v < 1) \land v \geq 0)\, v := -fv; \tag{5}$$
$$t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0 \land t \leq 1\big)^*\big](0 \leq h \leq 5)$$

Lecture 8 on Events & Delays identified the following invariant for this system:

$$2h = 2H - v^2 \land h \geq 0 \land h \leq 5 \tag{6}$$

although there was no proof yet. Recall that (global) invariants need to be augmented with the usual assumptions about the unchanged variables, like $g = 1 \land 1 = c \land 1 = f$. So let's define the formula we conjecture to be an invariant as:

$$\varphi \overset{\text{def}}{\equiv} 2h = 2H - v^2 \land h \geq 0 \land h \leq 5 \land g = 1 \land 1 = c \land 1 = f$$

With this invariant, (5) is provable in KeYmaera.

4 Cutting Real Arithmetic

The *cut* rule from Lecture 6 on Truth & Proof is not just a curiosity, but can be very helpful in practice. It can speed up real arithmetic a lot when using a cut to replace a difficult arithmetic formula by a simpler one that is sufficient for the proof.

For example, suppose $\psi(x)$ is a very complicated formula of first-order real arithmetic. Then proving the following formula

$$(x - y)^2 \leq 0 \land \psi(x) \rightarrow \psi(y)$$

by just real arithmetic will turn out to be surprisingly difficult and can take ages. Yet, thinking about it, $(x - y)^2 \leq 0$ implies that $y = x$, which should make the rest of the proof easy since, $\psi(x)$ should easily imply $\psi(y)$ if $y = x$. How do we exhibit a proof based on these thoughts?

The critical idea to make such a proof work is to use *cut* for a creative cut with the suitable arithmetic. So we choose $y = x$ as the cut formula ϕ in *cut* and proceed as follows:

$$\text{→r}\,\dfrac{\text{∧l}\,\dfrac{\textit{cut}\,\dfrac{\text{Wl}\,\dfrac{\text{Wr}\,\dfrac{(x-y)^2 \leq 0 \vdash y = x}{(x-y)^2 \leq 0 \vdash y = x, \psi(y)}}{(x-y)^2 \leq 0, \psi(x) \vdash y = x, \psi(y)} \qquad \text{Wl}\,\dfrac{\text{=r}\,\dfrac{\textit{ax}\,\dfrac{*}{\psi(x), y = x \vdash \psi(x)}}{\psi(x), y = x \vdash \psi(y)}}{(x-y)^2 \leq 0, \psi(x), y = x \vdash \psi(y)}}{(x-y)^2 \leq 0, \psi(x) \vdash \psi(y)}}{(x-y)^2 \leq 0 \land \psi(x) \vdash \psi(y)}}{\vdash (x-y)^2 \leq 0 \land \psi(x) \rightarrow \psi(y)}$$

Indeed, the left premise proves easily using real arithmetic. The right premise proves comparably easily as well. This proof uses proof rule =r that we discuss next.

5 Applying Equations by Substitution

The above cut proof uses the following proof rule for applying an equation to a formula ϕ by substituting the left-hand side x of an equation by its right-hand side θ. This substitution is sound, because x is assumed to be equal to θ in the antecedent. The same rule works applies to formulas ϕ that are in the antecedent (=l) as well as in the succedent (=r). Obviously, the assumed equality $x = \theta$ has to be in the antecedent for the rule to be sound.

$$(=r)\ \frac{\Gamma, x = \theta \vdash \phi_x^\theta, \Delta}{\Gamma, x = \theta \vdash \phi, \Delta} \qquad (=l)\ \frac{\Gamma, x = \theta, \phi_x^\theta \vdash \Delta}{\Gamma, x = \theta, \phi \vdash \Delta}$$

It would be okay to use the equation in the other direction for replacing all occurrences of θ by x, because the equation $\theta = x$ is equivalent to $x = \theta$.

Exercises

Exercise 1. Explicitly complete the proof steps that lead from (3) to the 3 branches identified in the lecture notes by writing a proper sequent derivation. Recall how updates are delayed substitutions and that they "hang around" until they can be applied.

Exercise 2. The sequent proof shown in these lecture notes is for the case coming from the lower dynamics h''=..≤ 5. This alone does not prove (1). Write a sequent proof for the missing branches coming from the upper dynamics h''=..≥ 5.

Exercise 3. Develop a sequent proof for the time-triggered ping pong ball (5). Is it easier or more difficult than the proof for (1)?

Exercise 4. Relate the event-driven system proof for (1) discussed in lecture to the proof that KeYmaera produces. What do they have in common? Where do they differ?

References

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

[Pla10] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14509-4.

[PQR09] André Platzer, Jan-David Quesel, and Philipp Rümmer. Real world verification. In Renate A. Schmidt, editor, *CADE*, volume 5663 of *LNCS*, pages 485–501. Springer, 2009. doi:10.1007/978-3-642-02959-2_35.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Differential Equations & Differential Invariants

André Platzer

Carnegie Mellon University

Lecture 10

1 Introduction

Lecture 5 on Dynamical Systems & Dynamic Axioms gave us a first simple proof principle for differential equations if we find a representable solution of the differential equation. The axiom [′] replaces properties of differential equations with suitably quantified properties of solutions, with a universal quantifier over all durations of the solution. Yet, that does not work for all differential equations, because only some of them have explicit closed-form solutions, and, of those, only very few have solutions that are simple enough to be quantified over without leaving the decidable parts of the resulting arithmetic.

Lecture 2 on Differential Equations & Domains allows many more differential equations to be part of CPS models than just the ones that happen to have simple solutions. In fact, in a certain sense, most of the interesting differential equations do not possess useful closed-form solutions. Today's lecture reinvestigates the way we prove properties of differential equations from a much more fundamental perspective, which will lead to a way of proving properties of CPS with more general differential equations.

More details can be found in [Pla10a, Pla10b, Chapter 3.5] and also [Pla12b]. Differential invariants were originally conceived in 2008 [Pla10a, Pla08] and later used for an automatic proof procedure for hybrid systems [PC08].

2 Global Descriptive Power of Local Differential Equations

Differential equations let physics evolve continuously for longer periods of time. They describe such global behavior locally.

Note 1 (Local descriptions of global behavior by differential equations)**.** *The key principle behind the descriptive power of differential equations is that they describe the evolution of a continuous process over time using only a local description of the direction into which the system evolves at any point in space. The solution of a differential equation is a global description of how the system evolves, while the differential equation itself is a local characterization.*

This difference between local description and global behavior can be exploited for proofs.

The semantics of a differential equation was described in Lecture 2 as:

$$\begin{aligned}\rho(x' = \theta \,\&\, H) = \{(\varphi(0), \varphi(r)) \quad : \quad & \varphi(t) \models x' = \theta \text{ and } \varphi(t) \models H \text{ for all } 0 \le t \le r \\ & \text{for a solution } \varphi : [0, r] \to \mathcal{S} \text{ of any duration } r\}\end{aligned}$$

The solution φ describes the global behavior of the system, which is specified locally by the right-hand side θ of the differential equation.

Lecture 2 has shown a number of examples illustrating the descriptive power of differential equations. That is, examples in which the solution was very complicated even though the differential equation was rather simple. This is a strong property of differential equations: they can describe even complicated processes in simple ways. Yet, that representational advantage of differential equations does not carry over into the verification when verification is stuck with proving properties of differential equations only by way of their solutions, which, by the very nature of differential equations, are more complicated.

This lecture, thus, investigates ways of proving properties of differential equations using the differential equations themselves, not their solutions. This technique is called *differential invariants* [Pla10a, Pla12b].

3 Differential Equations vs. Loops

A programmatic way of developing an intuition for differential invariants leads through a comparison of differential equations with loops [Pla12a]. This perhaps surprising relation can be made completely rigorous and is at the heart of a deep connection equating discrete and continuous dynamics proof-theoretically [Pla12a]. We will stay at the surface of this connection but still leverage the relation of differential equations to loops for our intuition.

To get started with relating differential equations to loops, compare

$$x' = \theta \qquad \text{vs.} \qquad (x' = \theta)^*$$

How does the differential equation $x' = \theta$ compare to the same differential equation in a loop $(x' = \theta)^*$ instead? Unlike the differential equation $x' = \theta$, the repeated differential equation $(x' = \theta)^*$ can run the differential equation $x' = \theta$ repeatedly. Albeit, on second thought, does that get the repetitive differential equation $(x' = \theta)^*$ to any more states than where the differential equation $x' = \theta$ could evolve to?

Not really, because chaining lots of solutions of differential equations from a repetitive differential equation $(x' = \theta)^*$ together will give a single solution for the same differential equation $x' = \theta$ that we could have followed just once all the way.[1]

Note 2 (Looping differential equations). *$(x' = \theta)^*$ is equivalent to $x' = \theta$, i.e. both have the same transition semantics. Differential equations "are their own loop".*[2]

In light of Note 2, differential equations look somewhat like loops. Like nondeterministic repetitions, differential equations might stop right away. Like nondeterministic repetitions, differential equations could evolve for longer or shorter durations. Like in nondeterministic repetitions, the outcome of the evolution of the system so far determines what happens next. And, in fact, in a deeper sense, differential equations actually really do correspond to loops [Pla12a].

With this rough relation in mind, let's advance the dictionary translating differential equation phenomena into loop phenomena and back. The local description of a differential equation as a relation $x' = \theta$ of the state to its derivative corresponds to the local description of a loop by a repetition operator α^*. The global behavior of a solution of a differential equation $x' = \theta$ corresponds to the full execution of a system that performs a repetition in a loop α^*. We also say that the local relation $x' = \theta$ is the generator of the global system solution and that the loop body α is the generator of the global behavior of repetition of the loop, because both local generators tell us everything about the system by way of their global interpretation as either differential or repetitive effect. Proving a property of a differential equation in terms of its solution corresponds to proving a property of a loop by unwinding it (infinitely long) by axiom $[{*}^n]$ from Lecture 5 on Dynamical Systems & Dynamic Axioms.

Now Lecture 7 on Control Loops & Invariants made the case that unwinding the iterations of a loop can be a rather tedious way of proving properties about the loop, because there is no good way of ever stopping to unwind, unless a counterexample can be found after a finite number of unwindings. Lecture 7 introduced induction with invariants instead to prove properties of loops, by, essentially, cutting the loop open and arguing that the generic state after any run of the loop body has the same characterization as the generic state before. After all these analogous correspondences between loops and differential equations, the obvious question is what the differential equation analogue proof concept would be that corresponds to proofs by induction for loops, which is the premier technique for proving loops.

Induction can be defined for differential equations using what is called *differential invariants* [Pla10a, Pla12b]. The have a similar principle as the proof rules for induction for loops. Differential invariants prove properties of the solution of the differential

[1] This is related to classical results about the continuation of solutions, e.g., [Pla10b, Proposition B.1].

[2] Beware not to confuse this with the case for differential equations with evolution domain constraints, which is subtly different.

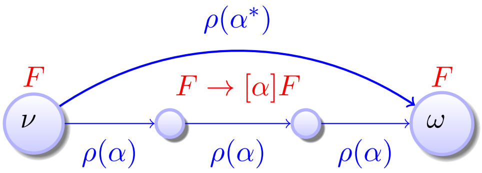
equation using only its local generator: the right-hand side of the differential equation.

Note 3 (Correspondence map between loops and differential equations)**.**

loop α^*	***differential equation*** $x' = \theta$
can skip over	*can evolve for duration 0*
repeat any number $n \in \mathbb{N}$ *of times*	*evolve for any duration* $0 \leq r \in \mathbb{R}$
effect depends on previous iteration	*effect depends on past solution*
local generator α	*local generator* $x' = \theta$
full execution trace	*global solution* φ
proof by unwinding iterations $[^{*n}]$	*proof by solution* $[']$
proof by induction with invariant *ind*	*proofs by differential invariants*

Recall from Lecture 7:

$$\rho(\alpha^*) = \bigcup_{n \in \mathbb{N}} \rho(\alpha^n) \qquad \text{with} \quad \alpha^{n+1} \equiv \alpha^n; \alpha \text{ and } \alpha^0 \equiv ?true$$



$$(ind)\ \frac{\Gamma \vdash F, \Delta \quad F \vdash [\alpha]F \quad F \vdash \psi}{\Gamma \vdash [\alpha^*]\psi, \Delta}$$

4 Intuition of Differential Invariants

Just as inductive invariants are the premier technique for proving properties of loops, differential invariants [Pla10a, Pla12b] provide the primary inductive technique we use for proving properties of differential equations (without having to solve them).

The core principle behind loop induction is that the induction step investigates the local generator α ands shows that it never changes the truth-value of the invariant F (also see the core induction proof rule *ind* from Lecture 7). Let us try to establish the same inductive principle, just for differential equations.

What does the local generator of a differential equation $x' = \theta$ tell us about the evolution of a system? And how does it relate to the truth of a formula F all along the solution of that differential equation? That is, to the truth of the d$\mathcal{L}$ formula $[x' = \theta]F$ expressing that all runs of $x' = \theta$ lead to states satisfying F. Fig. 1 depicts an example of a vector field for a differential equation, a global solution (in red), and an unsafe region $\neg F$ (shown in blue). The safe region F is the complement of the blue unsafe region $\neg F$.

One way of proving that $[x' = \theta]F$ is true in a state ν would be to compute a solution from that state ν, check every point in time along the solution to see if it is in the safe region F or the unsafe region $\neg F$. Unfortunately, these are uncountably infinitely

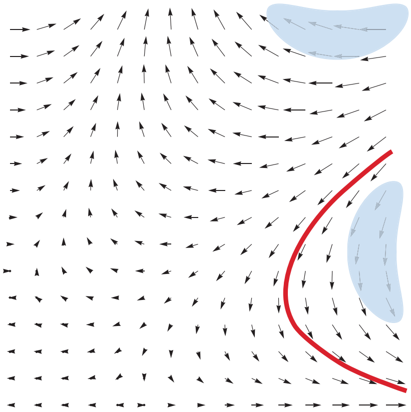


Figure 1: Vector field and one solution of a differential equation that does not enter the blue regions

many points in time to check. Furthermore, that only considers a single initial sate ν, so proving validity of a formula would require considering every of the uncountably infinitely many possible initial states and computing a solution in each of them. That is why this naïve approach would not compute.

A similar idea can still be made to work when the symbolic initial-value problem can be solved with a symbolic initial value x and a quantifier for time can be used, which is what the solution axiom [′] does. Yet, even that only works when a solution to the symbolic initial-value problem can be computed and the arithmetic resulting from the quantifier for time can be decided. For polynomial solutions, this works, for example. But polynomial come from very simple systems (called nilpotent linear differential equation systems).

Reexamining the illustration in Fig. 1, we suggest an entirely different way of checking whether the system could ever lead to an unsafe state in $\neg F$ when following the differential equation $x' = \theta$. The intuition is the following. If there were a vector in Fig. 1 that points from a safe state in F to an unsafe state $\neg F$ (in the blue region), then following that vector could get the system into an unsafe $\neg F$. If, instead, all vectors point from safe states to safe states in F, then, intuitively, following such a chain of vectors will only lead from safe states to safe states. So if the system also started in a safe state, it would stay safe.

Let us make this intuition rigorous to obtain a sound proof principle.

5 Deriving Differential Invariants

How can the intuition about directions of evolution of a logical formula F with respect to a differential equation $x' = \theta$ be made rigorous? We develop this step by step.

As an example, consider a conjecture about the rotational dynamics where d and e

represent the direction of a vector rotating clockwise in a circle of radius r (Fig. 2):

$$d^2 + e^2 = r^2 \rightarrow [d' = e, e' = -d]d^2 + e^2 = r^2 \tag{1}$$

The conjectured d$\mathcal{L}$ formula (1) is valid, because, indeed, if the vector (d, e) is initially at distance r from the origin (0,0), then it will always be when rotating around the origin, which is what the dynamics does. That is, the point (d, e) will always remain on the circle of radius r. But how can we prove that? In this particular case, we could possibly investigate solutions, which are trigonometric functions (although the ones shown in Fig. 2 are not the only solution). With those solutions, we could perhaps find an argument why they stay at distance r from the origin. But the resulting arithmetic will be unnecessarily difficult and, after all, the argument for why the simple d$\mathcal{L}$ formula (1) is valid should be easy. And it is, after we have discovered the right proof principle as this lecture will do.

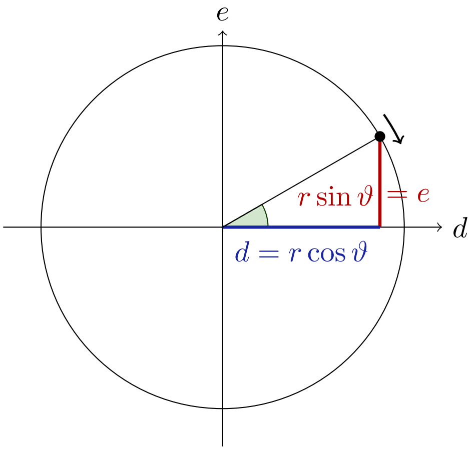


Figure 2: One scenario for the rotational dynamics and relationship of vector (d, e) to radius r and angle ϑ

First, what is the direction into which a continuous dynamical system evolves? The direction is exactly described by the differential equation, because the differential equation describes in which direction the state evolves at every point in space. So the direction into which a continuous system obeying $x' = \theta$ follows from state ν is exactly described by the time-derivative of the state being the term θ, i.e. $[\![\theta]\!]_\nu$. Recall that term θ can mention x and other variables so its value $[\![\theta]\!]_\nu$ depends on the state ν.

Note 4. *Proving* d$\mathcal{L}$ *formula* $[x' = \theta]F$ *does not require us to answer where the system evolves to but how the evolution of the system relates to formula* F *and the set of states* ν *in which* F *evaluates to* $true$.

The logical formula F is built from atomic formulas that are comparisons of (polynomial or rational) terms. Let η denote such a (polynomial) term in the variable (vector)

x. The semantics of a polynomial term η in a state ν is the real number $[\![\eta]\!]_\nu$ that it evaluates to. In which direction does the value of η evolve when following the differential equation $x' = \theta$ for some time? That depends both on the term η that is being evaluated and on the differential equation $x' = \theta$ that describes the evolution of x.

Directions of evolutions are described by derivatives, after all the differential equation $x' = \theta$ describes that the time-derivative of x is θ. Let's derive some term η of interest and see what that tells us about how η evolves over time. How can we derive η? The term η could be built from any of the operators discussed in Lecture 2, to which we now add division for rational terms to make it more interesting. Let Σ denote the set of all variables. *Terms* θ are defined by the grammar (where θ, η are terms, x a variable, and r a rational number constant):

$$\theta, \eta ::= x \mid r \mid \theta + \eta \mid \theta - \eta \mid \theta \cdot \eta \mid \theta / \eta$$

It is, of course, important to take care that division θ/η only makes sense in a context where the divisor η is guaranteed not to be zero in order to avoid undefinedness. Thus, we only allow division to be used in a context where the divisor is ensured not to be zero.

If η is a sum $a + b$, its derivative is the derivative of a plus the derivative of b. If η is a product $a \cdot b$, its derivative is the derivative of a times b plus a times the derivative of b. The derivative of a rational number constant $r \in \mathbb{Q}$ is zero.[3] The other operators are similar, leaving only the case of a single variable x. What is its derivative?

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

[3] Of course, the derivative of real number constants $r \in \mathbb{R}$ is also zero, but only rational number constants are allowed in the first-order logic of real arithmetic, more precisely, of real-closed fields.

The exact value of the derivative of x certainly depends on the state and on the evolution of the system. So for now, we just define the derivative of a variable x to be the symbol x' and consider what to do with it later.

Definition 1 (Derivation)**.** The operator $(\cdot)'$ that is defined as follows on terms is called *syntactic (total) derivation*:

$$
\begin{aligned}
(r)' &= 0 && \text{for numbers } r \in \mathbb{Q} && \text{(2a)}\\
(x)' &= x' && \text{for variable } x \in \Sigma && \text{(2b)}\\
(a+b)' &= (a)' + (b)' && && \text{(2c)}\\
(a-b)' &= (a)' - (b)' && && \text{(2d)}\\
(a \cdot b)' &= (a)' \cdot b + a \cdot (b)' && && \text{(2e)}\\
(a/b)' &= ((a)' \cdot b - a \cdot (b)')/b^2 && && \text{(2f)}
\end{aligned}
$$

Even though the following names are not crucial for the understanding of this course, let's briefly align Def. 1 with the algebraic structures from differential algebra [Kol72]. Case (2a) defines number symbols as *differential constants*, which do not change during continuous evolution. Their total derivative is zero. Equation (2c) and the *Leibniz* or *product rule* (2e) are defining conditions for *derivation operators on rings*. The derivative of a sum is the sum of the derivatives (additivity or a homomorphic property with respect to addition, i.e. the operator $(\cdot)'$ applied to a sum equals the sum of the operator applied to each summand) according to equation (2c). Furthermore, the derivative of a product is the derivative of one factor times the other factor plus the one factor times the derivative of the other factor as in (2e). Equation (2d) is a derived rule for subtraction according to $a - b = a + (-1) \cdot b$ and again expresses a homomorphic property, now with respect to subtraction. In addition, equation (2b) uniquely defines operator $(\cdot)'$ on the *differential polynomial algebra* spanned by the *differential indeterminates* $x \in \Sigma$. It says that we understand the differential symbol x' as the derivative of the symbol x for all state variables $x \in \Sigma$. Equation (2f) canonically extends $(\cdot)'$ to the *differential field of quotients* by the usual *quotient rule*. As the base field $\mathbb{R}$ has no zero divisors[4], the right-hand side of (2f) is defined whenever the original division a/b can be carried out, which, as we assumed, is guarded by $b \neq 0$.

The derivative of a division a/b uses a division, which is where we need to make sure not to accidentally divide by zero. Yet, in the definition of $(a/b)'$, the division is by b^2 which has the same roots that b has. So $b = 0 \leftrightarrow b^2 = 0$ is valid for any term b. Hence, in any context in which a/b was defined, its derivative $(a/b)'$ will also be.

Which of the terms should we derive when trying to prove (1)? Since that is not necessarily clear so far, let's turn the formula (1) around and consider the following equivalent d$\mathcal{L}$ formula instead, which only has a single nontrivial term to worry about:

$$d^2 + e^2 - r^2 = 0 \to [d' = e, e' = -d]d^2 + e^2 - r^2 = 0 \tag{3}$$

[4] In this setting, $\mathbb{R}$ have no zero divisors, because the formula $ab = 0 \to a = 0 \lor b = 0$ is valid, i.e. a product is zero only if a factor is zero.

Derivation of the relevant term $d^2 + e^2 - r^2$ in the postcondition of (3) gives

$$(d^2 + e^2 - r^2)' = 2dd' + 2ee' - 2rr' \tag{4}$$

Def. 1 makes it possible to derive polynomial and rational terms. Deriving them with the total derivative operator $(\cdot)'$ does not result in a term over the signature of the original variables in Σ, but, instead, a differential term, i.e. a term over the extended signature $\Sigma \cup \Sigma'$, where $\Sigma' \stackrel{\text{def}}{=} \{x' \ : \ x \in \Sigma\}$ is the set of all differential symbols x' for variables $x \in \Sigma$. In particular, the total derivative $(\eta)'$ of a polynomial term η is not a polynomial term, but may mention differential symbols such as x'. All syntactic elements of those differential terms are easy to interpret based on the semantics of terms defined in Lecture 2, except for the differential symbols. What is the meaning of a differential symbol x'?

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

6 The Meaning of Prime

The meaning $[\![x]\!]_\nu$ of a variable symbol x is defined by the state ν. The meaning of a differential symbol x' cannot be defined in a state ν, because derivatives do not even exist in isolated points. Along a (differentiable) continuous evolution $\varphi : [0, r] \to \mathcal{S}$ of a system, however, we can make sense of what x' means. At any point in time $\zeta \in [0, r]$ along such a continuous evolution φ, the differential symbol x' can be taken to mean the time-derivative of the value $[\![x]\!]_{\varphi(\zeta)}$ of x at ζ [Pla10a]:

Definition 2 (Differentially augmented state in differential state flow)**.** The value of x' at time $\zeta \in [0, r]$ of a differentiable function $\varphi : [0, r] \to \mathcal{S}$ of some duration $r \in \mathbb{R}$ is defined as:

$$[\![x']\!]_{\varphi(\zeta)} = \frac{\mathrm{d}\varphi(t)(x)}{\mathrm{d}t}(\zeta)$$

Intuitively, $[\![x']\!]_{\varphi(\zeta)}$ is determined by considering how the value $\varphi(\zeta)(x) = [\![x]\!]_{\varphi(\zeta)}$ of x changes along the function φ when we change time ζ "only a little bit". Visually, it corresponds to the slope of the tangent at time ζ; see Fig. 3.

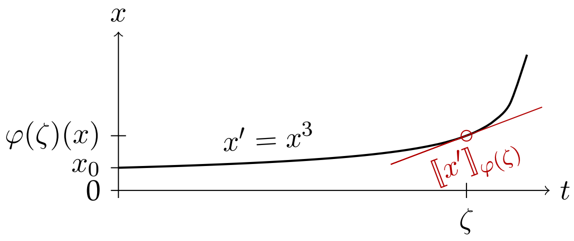


Figure 3: Differential state flow

Yet, what is the right-hand side in Def. 2, i.e. the time-derivative of the value of x along φ at time ζ? For differentiable φ, that is always defined, but that does not mean it would be computable. If, however, the continuous evolution φ is generated by a differential equation $x' = \theta$, i.e. φ solves $x' = \theta$, then $[\![x']\!]_{\varphi(\zeta)}$ can be described easily in terms of that differential equation, because at any time $\zeta \in [0, r]$ the time-derivative of the value of x is $[\![\theta]\!]_{\varphi(\zeta)}$, by definition of what it means for φ to be a solution of $x' = \theta$ (cf. Lecture 2).

Now Def. 1 defines how to derive a term η syntactically and Def. 2 defines how to interpret the differential symbols that occur in the total derivative $(\eta)'$. When interpreting all differential symbols as defined in Def. 2 for an evolution φ that follows the differential equation $x' = \theta$, this defines a value for the derivative $(\eta)'$ of any term η along that function φ. What does this value mean? How does it relate to how the value of η changes over time?

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

When interpreting differential symbols by derivatives along a function φ, the value of $(\eta)'$ at any time ζ coincides with the analytic time-derivative of the value of η at ζ.

The following central lemma, which is the differential counterpart of the substitution lemma, establishes the connection between syntactic derivation of terms and semantic differentiation as an analytic operation to obtain analytic derivatives of valuations along differential state flows. It will allow us to draw analytic conclusions about the behaviour of a system along differential equations from the truth of purely algebraic formulas obtained by syntactic derivation. In a nutshell, the following lemma shows that, along a flow, analytic derivatives of valuations coincide with valuations of syntactic derivations.

Lemma 3 (Derivation lemma). *Let $\varphi : [0, r] \to \mathcal{S}$ be a differentiable function of duration $r > 0$. Then for all terms η that are defined all along φ and all times $\zeta \in [0, r]$:*

$$\frac{\mathsf{d}\,[\![\eta]\!]_{\varphi(t)}}{\mathsf{d}t}(\zeta) = [\![(\eta)']\!]_{\varphi(\zeta)}$$

where differential symbols are interpreted according to Def. 2. In particular, $[\![\eta]\!]_{\varphi(\zeta)}$ is continuously differentiable.

Proof. The proof is an inductive consequence of the correspondence of the semantics of differential symbols and analytic derivatives along a flow (Def. 2). It uses the assumption that φ remains within the domain of definition of η and is continuously differentiable in all variables of η. In particular, all denominators are nonzero during φ.

- If η is a variable x, the conjecture holds immediately by Def. 2:

$$\frac{\mathsf{d}\,[\![x]\!]_{\varphi(t)}}{\mathsf{d}t}(\zeta) = \frac{\mathsf{d}\,\varphi(t)(x)}{\mathsf{d}t}(\zeta) = [\![(x)']\!]_{\varphi(\zeta)}.$$

  The derivative exists, because φ is assumed to be differentiable.

- If η is of the form $a + b$, the desired result can be obtained by using the properties of analytic derivatives, synctactic derivations (Def. 1), and valuation of terms (Lecture 2):

$$\begin{aligned}
&\frac{\mathsf{d}}{\mathsf{d}t}([\![a+b]\!]_{\varphi(t)})(\zeta) && \\
&= \frac{\mathsf{d}}{\mathsf{d}t}([\![a]\!]_{\varphi(t)} + [\![b]\!]_{\varphi(t)})(\zeta) && [\![\cdot]\!]_\nu \text{ homomorphic for } + \\
&= \frac{\mathsf{d}}{\mathsf{d}t}([\![a]\!]_{\varphi(t)})(\zeta) + \frac{\mathsf{d}}{\mathsf{d}t}([\![b]\!]_{\varphi(t)})(\zeta) && \frac{\mathsf{d}}{\mathsf{d}t} \text{ is a (linear) derivation} \\
&= [\![(a)']\!]_{\varphi(\zeta)} + [\![(b)']\!]_{\varphi(\zeta)} && \text{by induction hypothesis} \\
&= [\![(a)' + (b)']\!]_{\varphi(\zeta)} && [\![\cdot]\!]_\nu \text{ homomorphic for } + \\
&= [\![(a+b)']\!]_{\varphi(\zeta)} && (\cdot)' \text{ is a syntactic derivation}
\end{aligned}$$

- The case where η is of the form $a \cdot b$ or $a - b$ is similar, using Leibniz product rule (2e) or subtractivity (2d) of Def. 1, respectively.
- The case where η is of the form a/b uses (2f) of Def. 1 and further depends on the assumption that $b \neq 0$ along φ. This holds as the value of η is assumed to be defined all along state flow φ.
- The values of numbers $r \in \mathbb{Q}$ do not change during a state flow (in fact, they are not affected by the state at all); hence their derivative is $(r)' = 0$. □

Lemma 3 shows that the value of the total derivative of a term coincides with the analytic derivative of the term, provided that differential symbols are interpreted according to Def. 2. Along a differential equation $x' = \theta$, the differential symbols have a simple interpretation, the interpretation determined by the differential equation. Putting these thoughts together leads to replacing differential symbols with the corresponding right-hand sides of their respective differential equations. That is, replacing left-hand sides of differential equations with their right-hand sides.

Note 8. *The direction into which the value of a term η evolves as the system follows as differential equation $x' = \theta$ depends on the term η and the differential equation $x' = \theta$ that locally describes the evolution of x.*

The substitution property can be lifted to differential equations, i.e., differential equations can be used for equivalent substitutions along differential state flows respecting the corresponding differential constraints. In a nutshell, the following lemma can be used to substitute right-hand sides of differential equations for the left-hand side derivatives for flows along which these differential equations hold. For comparison, the classical substitution property says that equals can be substituted for equals, i.e., left-hand sides of equations can be substituted by right-hand sides of equations within formulas in which the equations hold.

Lemma 4 (Differential substitution property for terms)**.** *If $\varphi : [0, r] \to \mathcal{S}$ solves the differential equation $x' = \theta$, i.e. $\varphi \models x' = \theta$, then $\varphi \models (\eta)' = (\eta)'^{\theta}_{x'}$ for all terms η, i.e.:*

$$[\![(\eta)']\!]_{\varphi(\zeta)} = [\![(\eta)'^{\theta}_{x'}]\!]_{\varphi(\zeta)} \quad \textit{for all } \zeta \in [0, r]$$

Proof. The proof is a simple inductive consequence of Lemma 3 using that $[\![x']\!]_{\varphi(\zeta)} = [\![\theta]\!]_{\varphi(\zeta)}$ at each time ζ in the domain of φ. □

The operation mapping term η to $(\eta)'^{\theta}_{x'}$ is called *Lie-derivative* of η with respect to $x' = \theta$.

Differential substitution of the differential equation $d' = e, e' = -d$ from (3) into (4) results in

$$(d^2 + e^2 - r^2)'^{e}_{d'}{}^{-d}_{e'} = (2dd' + 2ee' - 2rr')^{e}_{d'}{}^{-d}_{e'} = 2de + 2e(-d) + 2rr'$$

Oops, that did not make all differential symbols disappear, because r' is still around, since r did not have a differential equation in (3). Stepping back, what we mean by a differential equation like $d' = e, e' = -d$ that does not mention r' is that r is not supposed to change. If r is supposed to change during a continuous evolution, there has to be a differential equation for r.

Note 10 (Explicit change)**.** *Hybrid programs are* explicit change*: nothing changes unless an assignment or differential equation specifies how (compare the semantics from Lecture 3). In particular, if a differential equation (system) $x' = \theta$ does not mention z', then z does not change during $x' = \theta$, so the original system $x' = \theta$ and $x' = \theta, z' = 0$ are equivalent.*

We will often assume $z' = 0$ without further notice for variables z that do not change during a differential equation.

Since (3) does not have a differential equation for r, Note 10 implies that its differential equation $d' = e, e' = -d$ is equivalent to $d' = e, e' = -d, r' = 0$. Hence, when adding zero derivatives for all unchanged variables, differential substitution of the differential equation $d' = e, e' = -d$ along with the explicit-change assumption $r' = 0$ into (4) gives

$$(d^2 + e^2 - r^2)'{}_{d'}^{e}\,{}_{e'}^{-d}\,{}_{r'}^{0} = (2dd' + 2ee' - 2rr')_{d'}^{e}\,{}_{e'}^{-d}\,{}_{r'}^{0} = 2de + 2e(-d) \tag{5}$$

This is good news, because the last part of (5) is a standard term of first-order logic of real arithmetic, because it no longer has any differential symbols. So we can make sense of $2de+2e(-d)$ and, by Lemma 4, its value along a solution of $d' = e, e' = -d$ is the same as that of the derivative $(d^2 + e^2 - r^2)'$, which, by Lemma 3 is the same as the value of the time-derivative of the original term $d^2 + e^2 - r^2$ along such a solution. Simple arithmetic shows that the term $2de + 2e(-d)$ in (5) is 0. Consequently, by Lemma 3 and Lemma 4, the time-derivative of the term $d^2 + e^2 - r^2$ in the postcondition of (3) is 0 along any solution φ of its differential equation:

$$\begin{aligned}
\frac{\mathsf{d}[\![d^2 + e^2 - r^2]\!]_{\varphi(t)}}{\mathsf{d}t}(\zeta) &\overset{\text{Lem}3}{=} [\![(d^2 + e^2 - r^2)']\!]_{\varphi(\zeta)} \\
&\overset{\text{Lem}4}{=} [\![(d^2 + e^2 - r^2)'{}_{d'}^{e}\,{}_{e'}^{-d}\,{}_{r'}^{0}]\!]_{\varphi(\zeta)} \\
&\overset{(5)}{=} [\![2de + 2e(-d)]\!]_{\varphi(\zeta)} = 0
\end{aligned}$$

for all times ζ. That means that the value of $d^2 + e^2 - r^2$ never changes during the rotation, and, hence (3) is valid, because $d^2 + e^2 - r^2$ stays 0 if it was 0 in the beginning, which is what (3) assumes.

7 Differential Invariant Terms

In order to be able to use the above reasoning as part of a sequent proof, we need to capture arguments like these in a proof rule, preferably one that is more general than this particular argument. The argument is not specific to the term $d^2 + e^2 - r^2$ but works for any other term η and for any differential equation $x' = \theta$. This would give us a soundness proof for the following proof rule.

Lemma 5 (Differential invariant terms)**.** *The following special case of the differential invariants proof rule is sound, i.e. if its premise is valid then so is its conclusion:*

$$(DI_{=0})\ \frac{\vdash {\eta'}_{x'}^{\theta} = 0}{\eta = 0 \vdash [x' = \theta]\eta = 0}$$

Proof. Assume the premise ${\eta'}_{x'}^{\theta} = 0$ to be valid, i.e. true in all states. In order to prove that the conclusion $\eta = 0 \vdash [x' = \theta]\eta = 0$ is valid, consider any state ν. Assume that $\nu \models \eta = 0$, as there is otherwise nothing to show (sequent is trivially *true* since antecedent evaluates to *false*). If $\zeta \in [0, r]$ is any time during any solution $\varphi : [0, r] \to \mathcal{S}$ of any duration $r \in \mathbb{R}$ of $x' = \theta$ beginning in initial state $\varphi(0) = \nu$, then

$$\frac{\mathsf{d}[\![\eta]\!]_{\varphi(t)}}{\mathsf{d}t}(\zeta) \overset{\text{Lem3}}{=} [\![(\eta)']\!]_{\varphi(\zeta)} \overset{\text{Lem4}}{=} [\![{(\eta)'}_{x'}^{\theta}]\!]_{\varphi(\zeta)} \overset{\text{premise}}{=} 0$$

By antecedent, $\nu \models \eta = 0$, i.e. $[\![\eta]\!]_\nu = 0$, in the initial state $\nu = \varphi(0)$.

If the duration of φ is $r = 0$, we have $\varphi(0) \models \eta = 0$ immediately, because $\nu \models \eta = 0$. For duration $r > 0$, we show that $\eta = 0$ holds all along the flow φ, i.e., $\varphi(\zeta) \models \eta = 0$ for all $\zeta \in [0, r]$.

Suppose there was a $\zeta \in [0, r]$ with $\varphi(\zeta) \models \eta \neq 0$, which will lead to a contradiction. The function $h : [0, r] \to \mathbb{R}$ defined as $h(t) = [\![\eta]\!]_{\varphi(t)}$ satisfies the relation $h(0) = 0 \neq h(\zeta)$, because $h(0) = [\![\eta]\!]_{\varphi(0)} = [\![\eta]\!]_\nu$ and $\nu \models \eta = 0$ by antecedent of the conclusion. By Lemma 3, h is continuous on $[0, r]$ and differentiable at every $\xi \in (0, r)$. By mean value theorem, there is a $\xi \in (0, \zeta)$ such that $\frac{\mathsf{d}h(t)}{\mathsf{d}t}(\xi) \cdot (\zeta - 0) = h(\zeta) - h(0) \neq 0$. In particular, we can conclude that $\frac{\mathsf{d}h(t)}{\mathsf{d}t}(\xi) \neq 0$. Now Lemma 3 implies that $\frac{\mathsf{d}h(t)}{\mathsf{d}t}(\xi) = [\![(\eta)']\!]_{\varphi(\xi)} \neq 0$. This, however, is a contradiction, because the premise implies that the formula $(\eta)' = 0$ is true in all states along φ, including $\varphi(\xi) \models (\eta)' = 0$, which contradicts $[\![(\eta)']\!] \neq 0$. □

This proof rule enables us to prove (3) easily in d$\mathcal{L}$'s sequent calculus:

$$\begin{array}{ll}
 & * \\ \cline{2-2}
\mathbb{R} & \vdash 2de + 2e(-d) - 0 = 0 \\ \cline{2-2}
 & \vdash (2dd' + 2ee' - 2rr' = 0)_{d'}^{e}{}_{e'}^{-d}{}_{r'}^{-0} \\ \cline{2-2}
\text{DI}_{=0} & d^2 + e^2 - r^2 = 0 \vdash [d' = e, e' = -d]d^2 + e^2 - r^2 = 0 \\ \cline{2-2}
\to\text{r} & \vdash d^2 + e^2 - r^2 = 0 \to [d' = e, e' = -d]d^2 + e^2 - r^2 = 0
\end{array}$$

The line proof step that This is an exciting development, because, thanks to differential invariants, the property (3) of a differential equation with a nontrivial solution has a very simple proof that we can easily check.

8 Summary

This lecture showed one simple special form of differential invariants: the form where the differential invariants are terms whose value always stays 0 along all solutions of a differential equation. The next lecture will investigate more general forms of differential invariants and more advanced proof principles for differential equations.

The most important insight of today's lecture was that complicated behavior of systems defined in terms of real analytic properties and semantics can be captured by purely syntactical proof principles using derivations. The derivation lemma proved that the values of syntactic derivations coincides with the analytic derivatives of the values. The differential substitution lemma allowed us the intuitive operation of substituting differential equations into terms. Proving properties of differential equations using these simple proof principles is much more civilized and effective than working with solutions of differential equations. The proofs are also computationally easier, because the proof arguments are local.

Exercises

Exercise 1. What happens in the proof of Lemma 5 if there is no solution φ? Show that this is not a counterexample to proof rule $\mathrm{DI}_{=0}$, but that the rule is sound in that case.

References

[Kol72] Ellis Robert Kolchin. *Differential Algebra and Algebraic Groups*. Academic Press, New York, 1972.

[PC08] André Platzer and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. In Aarti Gupta and Sharad Malik, editors, *CAV*, volume 5123 of *LNCS*, pages 176–189. Springer, 2008. doi:10.1007/978-3-540-70545-1_17.

[Pla08] André Platzer. *Differential Dynamic Logics: Automated Theorem Proving for Hybrid Systems*. PhD thesis, Department of Computing Science, University of Oldenburg, Dec 2008. Appeared with Springer.

[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.

[Pla12a] André Platzer. The complete proof theory of hybrid systems. In *LICS*, pages 541–550. IEEE, 2012. doi:10.1109/LICS.2012.64.

[Pla12b] 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.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Differential Equations & Proofs

André Platzer

Carnegie Mellon University

Lecture 11

1 Introduction

Lecture 5 on Dynamical Systems & Dynamic Axioms gave us a first simple proof principle for differential equations if we find a representable solution of the differential equation. The axiom [′] replaces properties of differential equations with suitably quantified properties of solutions, with a universal quantifier over all durations of the solution. Yet, that does not work for all differential equations, because only some of them have explicit closed-form solutions let alone solutions that are simple enough to be quantified over without leaving the decidable parts of the resulting arithmetic.

Lecture 2 on Differential Equations & Domains allows many more differential equations to be part of CPS models than just the ones that happen to have simple solutions. In fact, in a certain sense, most of the interesting differential equations do not possess useful closed-form solutions. Today's lecture reinvestigates the way we prove properties of differential equations from a much more fundamental perspective, which will lead to a way of proving properties of CPS with more general differential equations.

More details can be found in [Pla10a, Pla10b, Chapter 3.5] and also [Pla12b]. Differential invariants were originally conceived in 2008 [Pla10a, Pla08] and later used for an automatic proof procedure for hybrid systems [PC08].

2 Recall

Recall the following results from Lecture 10 on Differential Equations & Differential Invariants:

Definition 1 (Derivation)**.** The operator $(\cdot)'$ that is defined as follows on terms is called *syntactic (total) derivation*:

$$
\begin{aligned}
(r)' &= 0 && \text{for numbers } r \in \mathbb{Q} && \text{(1a)}\\
(x)' &= x' && \text{for variable } x \in \Sigma && \text{(1b)}\\
(a+b)' &= (a)' + (b)' && && \text{(1c)}\\
(a-b)' &= (a)' - (b)' && && \text{(1d)}\\
(a\cdot b)' &= (a)'\cdot b + a\cdot (b)' && && \text{(1e)}\\
(a/b)' &= ((a)'\cdot b - a\cdot (b)')/b^2 && && \text{(1f)}
\end{aligned}
$$

Definition 2 (Differentially augmented state in differential state flow)**.** The value of x' at time $\zeta \in [0,r]$ of a differentiable function $\varphi : [0,r] \to \mathcal{S}$ of some duration $r \in \mathbb{R}$ is defined as:

$$[\![x']\!]_{\varphi(\zeta)} = \frac{\mathrm{d}\varphi(t)(x)}{\mathrm{d}t}(\zeta)$$

Lemma 3 (Derivation lemma)**.** *Let $\varphi : [0,r] \to \mathcal{S}$ be a differentiable function of duration $r > 0$. Then for all terms η that are defined all along φ and all times $\zeta \in [0,r]$:*

$$\frac{\mathrm{d}\,[\![\eta]\!]_{\varphi(t)}}{\mathrm{d}t}(\zeta) = [\![(\eta)']\!]_{\varphi(\zeta)}$$

where differential symbols are interpreted according to Def. 2. In particular, $[\![\eta]\!]_{\varphi(\zeta)}$ is continuously differentiable.

Lemma 4 (Differential substitution property for terms)**.** *If $\varphi : [0,r] \to \mathcal{S}$ solves the differential equation $x' = \theta$, i.e. $\varphi \models x' = \theta$, then $\varphi \models (\eta)' = (\eta)'^{\theta}_{x'}$ for all terms η, i.e.:*

$$[\![(\eta)']\!]_{\varphi(\zeta)} = [\![(\eta)'^{\theta}_{x'}]\!]_{\varphi(\zeta)} \quad \textit{for all } \zeta \in [0,r]$$

3 Differential Invariant Terms

Lecture 10 on Differential Equations & Differential Invariants proved soundness for a proof rule for differential invariant terms, which can be used to prove normalized invariant equations of the form $\eta = 0$.

Lemma 5 (Differential invariant terms)**.** *The following special case of the differential invariants proof rule is sound, i.e. if its premise is valid then so is its conclusion:*

$$(DI_{=0})\ \frac{\vdash {\eta'}_{x'}^{\theta} = 0}{\eta = 0 \vdash [x' = \theta]\eta = 0}$$

4 Proof by Generalization

So far, the argument captured in the differential invariant term proof rule $\mathrm{DI}_{=0}$ works for

$$d^2 + e^2 - r^2 = 0 \rightarrow [d' = e, e' = -d]d^2 + e^2 - r^2 = 0 \tag{2}$$

with an equation $d^2 + e^2 - r^2 = 0$ normalized to having 0 on the right-hand side but not for the original formula

$$d^2 + e^2 = r^2 \rightarrow [d' = e, e' = -d]d^2 + e^2 = r^2 \tag{3}$$

because its postcondition is not of the form $\eta = 0$. Yet, the postcondition $d^2 + e^2 - r^2 = 0$ of (2) is trivially equivalent to the postcondition $d^2 + e^2 = r^2$ of (3), just by rewriting the polynomials on one side, which is a minor change. That is an indication, that differential invariants can perhaps do more than what proof rule $\mathrm{DI}_{=0}$ already knows about.

But before we pursue our discovery of what else differential invariants can do for us any further, let us first understand a very important proof principle.

Note 6 (Proof by generalization)**.** *If you do not find a proof of a formula, it can sometimes be easier to prove a more general property from which the one you were looking for follows.*

This principle, which may at first appear paradoxical, turns out to be very helpful. In fact, we have made ample use of Note 6 when proving properties of loops by induction. The loop invariant that needs to be proved is usually more general than the particular postcondition one is interested in. The desirable postcondition follows from having proved a more general inductive invariant.

In its purest form, the principle of generalization is captured in the *generalization* rule from Lecture 7 on Control Loops & Invariants. One of the forms of the generalization rule is:

$$([]gen')\ \frac{\Gamma \vdash [\alpha]\phi, \Delta \quad \phi \vdash \psi}{\Gamma \vdash [\alpha]\psi, \Delta}$$

Instead of proving the desirable postcondition ψ of α (conclusion), proof rule $[]gen'$ makes it possible to prove the postcondition ϕ instead (left premise) and prove that ϕ is more general than the desired ψ (right premise). Generalization $[]gen'$ can help us prove the original d$\mathcal{L}$ formula (3) by first turning the postcondition into the form of the (provable) (2) and adapting the precondition using a corresponding *cut* with $d^2+e^2-r^2=0$:

$$\scriptstyle{\to\mathrm{r}}\dfrac{\scriptstyle{[]gen'}\dfrac{\scriptstyle{cut,\mathrm{Wl},\mathrm{Wr}}\dfrac{\mathbb{R}\dfrac{*}{d^2+e^2=r^2\vdash d^2+e^2-r^2=0}\quad \scriptstyle{\mathrm{DI}_{=0}}\dfrac{\dfrac{\mathbb{R}\dfrac{*}{\vdash 2de+2e(-d)-0=0}}{\vdash (2dd'+2ee'-2rr'=0)_{d'}^{e}{}_{e'}^{-d}{}_{r'}^{-0}}}{d^2+e^2-r^2=0\vdash [d'=e,e'=-d]d^2+e^2-r^2=0}}{d^2+e^2=r^2\vdash [d'=e,e'=-d]d^2+e^2-r^2=0}\quad \mathbb{R}\dfrac{*}{d^2+e^2-r^2=0\vdash d^2+e^2=r^2}}{d^2+e^2=r^2\vdash [d'=e,e'=-d]d^2+e^2=r^2}}{\vdash d^2+e^2=r^2\to [d'=e,e'=-d]d^2+e^2=r^2}$$

This is a possible way of proving the original (3), but also unnecessarily complicated. Differential invariants can prove (3) directly once we generalize proof rule $\mathrm{DI}_{=0}$ appropriately. For other purposes, however, it is still important to have the principle of generalization Note 6 in our repertoire of proof techniques.

5 Equational Differential Invariants

There are more general logical formulas that we would like to prove to be invariants of differential equations, not just the polynomial equations normalized such that they are single terms equaling 0. Thinking back of the soundness proof for $\mathrm{DI}_{=0}$ in Lecture 10, the argument used involving the value of the left-hand side term $h(t)=[\![\eta]\!]_{\varphi(t)}$ as a function of time t. The same argument can be made by considering the difference $h(t)=[\![\theta-\eta]\!]_{\varphi(t)}$ instead to prove postconditions of the form $\theta=\eta$. How does the inductive step for formula $\theta=\eta$ need to be define to make a corresponding differential invariant proof rule sound? That is, for what premise is the following a sound proof rule?

$$\dfrac{\vdash\ ???}{\theta=\eta\vdash [x'=\theta]\theta=\eta}$$

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

Defining the total derivative of an equation $\theta = \eta$ as

$$(\theta = \eta)' \equiv ((\theta)' = (\eta)')$$

results in a sound proof rule by a simple variation of the soundness proof for $\text{DI}_{=0}$ as sketched above. The resulting proof rule

$$(\text{DI}_{=})\ \frac{\vdash (\kappa' = \eta')_{x'}^{\theta}}{\kappa = \eta \vdash [x' = \theta]\kappa = \eta}$$

for equational differential invariants captures the basic intuition that κ always stays equal to η if it has been initially (antecedent of conclusion) and the derivative of κ is the same as the derivative of η with respect to the differential equation $x' = \theta$. This intuition is made precise by Lemma 3 and Lemma 4. Instead of going through a proper soundness proof for $\text{DI}_{=}$, however, let's directly generalize the proof principles further and see if differential invariants can prove even more formulas for us. We will later prove soundness for the general differential invariant rule, from which $\text{DI}_{=}$ derives as a special case.

Example 6 (Rotational dynamics). The rotational dynamics $d' = e, e' = -d$ is complicated in that the solution involves trigonometric functions, which are generally outside decidable classes of arithmetic. Yet, we can easily prove interesting properties about it using DI and decidable polynomial arithmetic. For instance, $\text{DI}_{=}$ can directly prove formula (3), i.e. that $d^2 + e^2 = r^2$ is a differential invariant of the dynamics, using the following proof:

$$\to\text{r}\ \dfrac{\text{DI}\ \dfrac{\dfrac{\mathbb{R}\ \dfrac{*}{\vdash 2de + 2e(-d) = 0}}{\vdash (2dd' + 2ee' = 0)_{d'}^{e}{}_{e'}^{-d}}}{d^2 + e^2 = r^2 \vdash [d' = e, e' = -d]d^2 + e^2 = r^2}}{\vdash d^2 + e^2 = r^2 \to [d' = e, e' = -d]d^2 + e^2 = r^2}$$

This proof is certainly much easier and more direct than the previous proof based on $[]gen'$.

6 Differential Invariant Inequalities

The differential invariant proof rules considered so far give a good (initial) understanding of how to prove equational invariants. What about inequalities? How can they be proved?

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

The primary question is again how to define the total derivative

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

Example 7 (Cubic dynamics). Similarly, differential induction can easily prove that $\frac{1}{3} \leq 5x^2$ is an invariant of the cubic dynamics $x' = x^3$; see the proof in Fig. 7 for the dynamics in Fig. 1. To apply the differential induction rule DI, we again form the total derivative of the differential invariant $F \equiv \frac{1}{3} \leq 5x^2$, which gives the differential expression $F' \equiv (\frac{1}{3} \leq 5x^2)' \equiv 0 \leq 5 \cdot 2xx'$. Now, the differential induction rule DI takes into account that the derivative of state variable x along the dynamics is known. Substituting the differential equation $x' = x^3$ into the inequality yields $F'^{x^3}_{x'} \equiv 0 \leq 5 \cdot 2xx^3$, which is a valid formula and closes by quantifier elimination with $\mathbb{R}$.

$$\text{DI}\;\dfrac{\dfrac{\mathbb{R}\;\dfrac{*}{\vdash 0 \leq 5 \cdot 2x(x^3)}}{\vdash (0 \leq 5 \cdot 2xx')^{x^3}_{x'}}}{\frac{1}{3} \leq 5x^2 \vdash [x' = x^3]\frac{1}{3} \leq 5x^2}$$

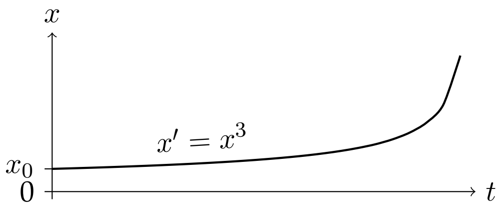


Figure 1: **a** Cubic dynamics proof 1**b**: Cubic dynamics

Differential invariants that are inequalities are not just a minor variation of equational differential invariants, because they can prove more. That is, it can be shown [Pla12b] that there are valid formulas that can be proved using differential invariant inequalities but cannot be proved just using equations as differential invariants ($\text{DI}_=$). So sometimes, you need to be prepared to look for inequalities that you can use as differential invariants. The converse is not true. Everything that is provable using $\text{DI}_=$ is also provable using differential invariant inequalities [Pla12b], but you should still look for equational differential invariants if they give easier proofs.

Strict inequalities can also be used as differential invariants when defining their total derivatives as:

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

It is easy to see (Exercise 1) that the following slightly relaxed definition would also be sound:

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

Understanding that differential substitution is sound for formulas, i.e. replacing the left-hand side of the differential equation by its right-hand side, requires a few more thoughts now, because the equational differential substitution principle Lemma 4 does not apply directly. The differential substitution principle not only works for terms, however, but also for differential first-order formulas, i.e. first-order formulas in which differential symbols occur:

Lemma 8 (Differential substitution property for differential formulas)**.** *If $\varphi : [0, r] \to \mathcal{S}$ solves the differential equation $x' = \theta$, i.e. $\varphi \models x' = \theta$, then $\varphi \models \mathcal{D} \leftrightarrow \mathcal{D}^{\theta}_{x'}$ for all differential first-order formulas $\mathcal{D}$, i.e. first-order formulas over $\Sigma \cup \Sigma'$.*

Proof. The proof is by using the Substitution Lemma [Pla10b, Lemma 2.2] for first-order logic on the basis of $[\![x']\!]_{\varphi(\zeta)} = [\![\theta]\!]_{\varphi(\zeta)}$ at each time ζ in the domain of φ by Def. 2. □

By Lemma 8, differential equations can always be substituted in along their solutions. Hence, the focus on developing differential invariant proof rules is in defining appropriate total derivatives, since Lemma 8 shows how to handle differential symbols by substitution.

Where do differential first-order formulas come from? They come from the analogue of the total derivation operator on formulas. On formulas, the total derivation operator applies the total derivation operator from Def. 1 to all terms in a first-order formula, yet it also flips disjunctions into conjunctions and existential quantifiers into universal quantifiers.

7 Disequational Differential Invariants

The case that is missing in differential invariant proof rules are for postconditions that are disequalities $\theta \neq \eta$? How can they be proved?

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

By analogy to the previous cases, one might expect the following definition:

$$(\theta \neq \eta)' \overset{?}{\equiv} ((\theta)' \neq (\eta)') \quad ???$$

It is crucial for soundness of differential invariants tha $(\theta \neq \eta)'$ is *not* defined that way! In the following counterexample, variable x can reach $x = 0$ without its derivative ever being 0; again, see Fig. 2 for the dynamics. Of course, just because θ and η start out different, does not mean they would always stay different if they evolve with different derivatives.

$$\lightning\;\cfrac{\cfrac{*\,\text{(unsound)}}{\vdash 1 \neq 0}}{x \neq 5 \vdash [x' = 1]x \neq 5}$$

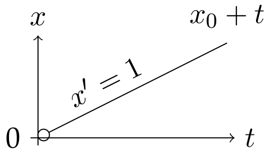


Figure 2: **a** Unsound attempt of using disequalities **2b**: Linear dynamics

Instead, if θ and η start out differently and evolve with the same derivatives, they will always stay different. So the sound definition is slightly unexpected:

$$(\theta \neq \eta)' \equiv ((\theta)' = (\eta)')$$

8 Conjunctive Differential Invariants

The next case to consider is where the invariant that we want to prove is a conjunction $F \wedge G$. Lemma 8 takes care of how to handle differential substitution for the differential equations, if only we define the correct total derivative of $(F \wedge G)'$.

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

To show that a conjunction $F \wedge G$ is invariant it is perfectly sufficient to prove that both are invariant. This can be justified separately, but is more obvious when recalling the following equivalence from Lecture 7:

$$([]\wedge)\ [\alpha](\phi \wedge \psi) \leftrightarrow [\alpha]\phi \wedge [\alpha]\psi$$

which is valid for all hybrid programs α, also when α is just a differential equation. Consequently, the total derivative of a conjunction is the conjunction of the total derivatives (i.e. $(\cdot)'$ is a homomorphism for $\wedge$):

$$(F \wedge G)' \equiv (F)' \wedge (G)'$$

Again, we will not develop a proper soundness argument, because it will follow from the general differential invariant proof rule.

With a corresponding proof rule that enables us to do the following proof:

$$\text{DI}\,\dfrac{\dfrac{\mathbb{R}\,\dfrac{*}{\vdash 2de + 2e(-d) \le 0 \wedge 2de + 2e(-d) \ge 0}}{\vdash (2dd' + 2ee' \le 0 \wedge 2dd' + 2ee' \ge 0)_{d'}^{e}{}_{e'}^{-d}}}{d^2 + e^2 \le r^2 \wedge d^2 + e^2 \ge r^2 \vdash [d' = e, e' = -d](d^2 + e^2 \le r^2 \wedge d^2 + e^2 \ge r^2)}$$

Since the invariant $d^2 + e^2 \le r^2 \wedge d^2 + e^2 \ge r^2$ is easily proved to be equivalent to $d^2 + e^2 = r^2$, the above proof gives yet another proof of (3) when combined with a corresponding use of $[]gen'$.

9 Disjunctive Differential Invariants

The next case to consider is where the invariant that we want to prove is a disjunction $F \vee G$. Lemma 8 takes care of how to handle differential substitution for the differential equations, if only we define the correct total derivative of $(F \vee G)'$. How?

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

The total derivative of a conjunction is the conjunction of the total derivatives. So, by analogy, it might stand to reason to define the total derivative of a disjunction as the disjunction of the total derivatives.

$$(F \lor G)' \overset{?}{\equiv} (F)' \lor (G)' \quad ???$$

Let's try it:

$$\mathbb{R}\,\dfrac{\dfrac{\text{unsound}}{\vdash 2de + 2e(-d) = 0 \lor 5d + re \geq 0}}{\dfrac{\vdash (2dd' + 2ee' = 0 \lor r'd + rd' \geq 0)_{d'}^{e}{}_{e'}^{-d}}{\lightning\, d^2 + e^2 = r^2 \lor rd \geq 0 \vdash [d' = e, e' = -d, r' = 5](d^2 + e^2 = r^2 \lor rd \geq 0)}}$$

That would be spectacularly wrong, however, because the formula at the bottom is not actually valid. We have no business of proving formulas that are not valid and if we ever could, we would have found a serious unsoundness in the proof rules.

For soundness of differential induction, it is crucial that Def. 1 defines the total derivative $(F \lor G)'$ of a disjunction conjunctively as $(F)' \land (G)'$ instead of as $(F)' \lor (G)'$. From an initial state ν which satisfies $\nu \models F$, and hence $\nu \models F \lor G$, the formula $F \lor G$ only is sustained differentially if F itself is a differential invariant, not if G is. For instance, $d^2 + e^2 = r^2 \lor rd \geq 0$ is no invariant of the above differential equation, because $rd \geq 0$ will be invalidated if we just follow the circle dynamics long enough. So if the disjunction was true because $rd \geq 0$ was true in the beginning, it does not stay invariant.

In practice, splitting differential induction proofs over disjunctions can be useful if a direct proof with a single differential invariant does not succeed:

$$\text{→r}\,\dfrac{\text{∨l}\,\dfrac{[]gen'\,\dfrac{\text{DI}\,\dfrac{\vdash A'^{\theta}_{x'}}{A \vdash [x' = \theta]A} \qquad \text{∨r}\,\dfrac{ax\,\dfrac{*}{A \vdash A, B}}{A \vdash A \lor B}}{A \vdash [x' = \theta](A \lor B)} \qquad []gen'\,\dfrac{\text{DI}\,\dfrac{\vdash B'^{\theta}_{x'}}{B \vdash [x' = \theta]B} \qquad \text{∨r}\,\dfrac{ax\,\dfrac{*}{B \vdash A, B}}{B \vdash A \lor B}}{B \vdash [x' = \theta](A \lor B)}}{A \lor B \vdash [x' = \theta](A \lor B)}}{\vdash A \lor B \to [x' = \theta](A \lor B)}$$

10 Differential Invariants

Differential invariants are a general proof principles for proving invariants of formulas. Summarizing what this lecture has discovered so far leads to a single proof rule for differential invariants. That is why all previous proofs just indicated DI when using the various special cases of the differential invariant proof rule to be developed next.

All previous arguments remain valid when the differential equation has an evolution domain constraint H that it cannot leave by definition. In that case, the inductive proof step can even assume the evolution domain constraint to hold, because the system, by definition, is not allowed to leave it.

Definition 9 (Derivation)**.** The operator $(\cdot)'$ that is defined as follows on first-order real-arithmetic formulas is called *syntactic (total) derivation*:

$$(F \land G)' \equiv (F)' \land (G)' \tag{4a}$$
$$(F \lor G)' \equiv (F)' \land (G)' \tag{4b}$$
$$(\forall x\, F)' \equiv \forall x\,(F)' \tag{4c}$$
$$(\exists x\, F)' \equiv \forall x\,(F)' \tag{4d}$$
$$(a \geq b)' \equiv (a)' \geq (b)' \qquad \text{accordingly for } <, >, \leq, =, \text{but not } \neq \tag{4e}$$

Furthermore, $F'^{\theta}_{x'}$ is defined to be the result of substituting θ for x' in F'. The operation mapping F to $(F)'^{\theta}_{x'}$ is called *Lie-derivative* of F with respect to $x' = \theta$.

That is, to replace the left-hand side of a differential equation by the right-hand side.

Lemma 10 (Differential invariants)**.** *The differential invariant rule is sound:*

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

The version DI′ can be derived easily from the more fundamental, essential form DI.

The basic idea behind rule DI is that the premise of DI shows that the total derivative F' holds within evolution domain H when substituting the differential equations $x' = \theta$ into F'. If F holds initially (antecedent of conclusion), then F itself always stays true (succedent of conclusion). Intuitively, the premise gives a condition showing that, within H, the total derivative F' along the differential constraints is pointing inwards or transversally to F but never outwards to $\neg F$; see Fig. 3 for an illustration. Hence, if we start in F and, as indicated by F', the local dynamics never points outside F, then the system always stays in F when following the dynamics. Observe that, unlike F', the premise of DI is a well-formed formula, because all differential expressions are replaced by non-differential terms when forming $F'^{\theta}_{x'}$.

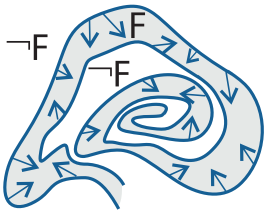


Figure 3: Differential invariant F for safety

Proof. Assume the premise $F'^{\theta}_{x'} = 0$ to be valid, i.e. true in all states. In order to prove that the conclusion $F \vdash [x' = \theta]F$ is valid, consider any state ν. Assume that $\nu \models F$, as there is otherwise nothing to show (sequent is trivially *true* since antecedent evaluates to *false*). If $\zeta \in [0, r]$ is any time during any solution $\varphi : [0, r] \to \mathcal{S}$ of any duration $r \in \mathbb{R}$ of $x' = \theta$ beginning in initial state $\varphi(0) = \nu$, then it remains to be shown that $\varphi(r) \models F$. By antecedent, $\nu \models F$, in the initial state $\nu = \varphi(0)$.

If the duration of φ is $r = 0$, we have $\varphi(0) \models F$ immediately, because $\nu \models F$. For duration $r > 0$, we show that F holds all along φ, i.e., $\varphi(\zeta) \models F$ for all $\zeta \in [0, r]$.

We have to show that $\nu \models F \to [x' = \theta \,\&\, H]F$ for all states ν. Let ν satisfy $\nu \models F$ as, otherwise, there is nothing to show. We can assume F to be in disjunctive normal form and consider any disjunct G of F that is true at ν. In order to show that F remains true during the continuous evolution, it is sufficient to show that each conjunct of G is. We can assume these conjuncts to be of the form $\eta \geq 0$ (or $\eta > 0$ where the proof is accordingly). Finally, using vectorial notation, we write $x' = \theta$ for the differential equation system. Now let $\varphi : [0, r] \to (V \to \mathbb{R})$ be any solution of $x' = \theta \,\&\, H$ beginning in $\varphi(0) = \nu$. If the duration of φ is $r = 0$, we have $\varphi(0) \models \eta \geq 0$ immediately, because $\nu \models \eta \geq 0$. For duration $r > 0$, we show that $\eta \geq 0$ holds all along the solution φ, i.e., $\varphi(\zeta) \models \eta \geq 0$ for all $\zeta \in [0, r]$.

Suppose there was a $\zeta \in [0, r]$ with $\varphi(\zeta) \models \eta < 0$, which will lead to a contradiction. The function $h : [0, r] \to \mathbb{R}$ defined as $h(t) = [\![\eta]\!]_{\varphi(\zeta)}$ satisfies the relation $h(0) \geq 0 > h(\zeta)$, because $h(0) = [\![\eta]\!]_{\varphi(0)} = [\![\eta]\!]_{\nu}$ and $\nu \models \eta \geq 0$ by antecedent of the conclusion. By Lemma 3, h is continuous on $[0, r]$ and differentiable at every $\xi \in (0, r)$. By mean value theorem, there is a $\xi \in (0, \zeta)$ such that $\frac{\mathrm{d}h(t)}{\mathrm{d}t}(\xi) \cdot (\zeta - 0) = h(\zeta) - h(0) < 0$. In particular, since $\zeta \geq 0$, we can conclude that $\frac{\mathrm{d}h(t)}{\mathrm{d}t}(\xi) < 0$. Now Lemma 3 implies that $\frac{\mathrm{d}h(t)}{\mathrm{d}t}(\xi) = [\![(\eta)']\!]_{\varphi(\xi)} < 0$. This, however, is a contradiction, because the premise implies that the formula $H \to (\eta \geq 0)'$ is true in all states along φ, including $\varphi(\xi) \models H \to (\eta \geq 0)'$. In particular, as φ is a solution for $x' = \theta \,\&\, H$, we know that $\varphi(\xi) \models H$ holds, and we have $\varphi(\xi) \models (\eta \geq 0)'$, which contradicts $[\![(\eta)']\!] < 0$. □

This proof rule enables us to prove (2) easily in d$\mathcal{L}$'s sequent calculus and all previous proofs as well:

$$\dfrac{\text{DI}\;\dfrac{\dfrac{\mathbb{R}\;\dfrac{*}{\vdash 2de + 2e(-d) \leq 0}}{\vdash (2dd' + 2ee' \leq 2rr')_{d'}^{e}\,{}_{e'}^{-d}\,{}_{r'}^{-0}}}{d^2 + e^2 \leq r^2 \vdash [d' = e, e' = -d]d^2 + e^2 \leq r^2}}{\vdash d^2 + e^2 \leq r^2 \to [d' = e, e' = -d]d^2 + e^2 \leq r^2}\;{\to}\text{r}$$

11 Example Proofs

Example 11 (Quartic dynamics). The following simple d$\mathcal{L}$ proof uses DI to prove an invariant of a quartic dynamics.

$$
{\scriptstyle\text{DI}}\dfrac{\dfrac{{\scriptstyle\mathbb{R}}\dfrac{*}{a \geq 0 \vdash 3x^2((x-3)^4+a) \geq 0}}{a \geq 0 \vdash (3x^2x' \geq 0)_{x'}^{(x-3)^4+a}}}{x^3 \geq -1 \vdash [x' = (x-3)^4 + a \,\&\, a \geq 0]x^3 \geq -1}
$$

Observe that rule DI directly makes the evolution domain constraint $a \geq 0$ available as an assumption in the premise, because the continuous evolution is never allowed to leave it.

Example 12. Consider the dynamics $x' = y, y' = -\omega^2 x - 2d\omega y$ of the damped oscillator with the undamped angular frequency ω and the damping ratio d. See Fig. 4 for one example of an evolution along this continuous dynamics. Figure 4 shows a trajectory in the x, y space on the left, and an evolution of x over time t on the right. General symbolic solutions of symbolic initial-value problems for this differential equation can become surprisingly difficult. Mathematica, for instance, produces a long equation of exponentials that spans 6 lines of terms just for one solution. A differential invariant proof, instead, is very simple:

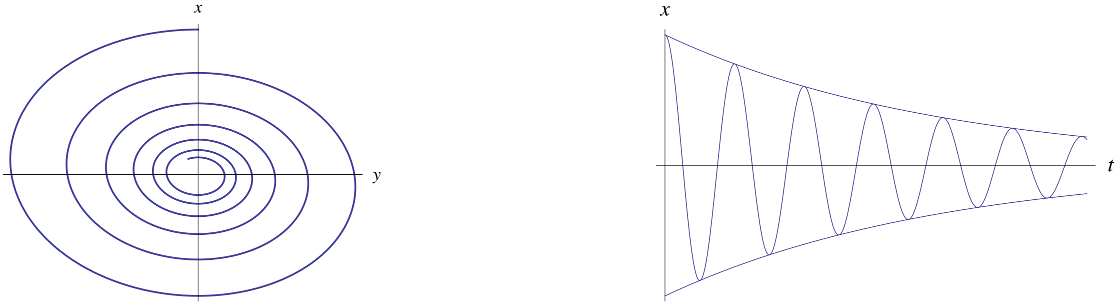


Figure 4: Trajectory and evolution of a damped oscillator

$$
{\scriptstyle\text{DI}}\dfrac{\dfrac{{\scriptstyle\mathbb{R}}\dfrac{*}{\omega \geq 0 \land d \geq 0 \vdash 2\omega^2 xy - 2\omega^2 xy - 4d\omega y^2 \leq 0}}{\omega \geq 0 \land d \geq 0 \vdash (2\omega^2 x x' + 2yy' \leq 0)_{x'}^{y}\,{}_{y'}^{-\omega^2 x - 2d\omega y}}}{\omega^2 x^2 + y^2 \leq c^2 \vdash [x' = y, y' = -\omega^2 x - 2d\omega y \,\&\, (\omega \geq 0 \land d \geq 0)]\, \omega^2 x^2 + y^2 \leq c^2}
$$

Observe that rule DI directly makes the evolution domain constraint $\omega \geq 0 \land d \geq 0$ available as an assumption in the premise, because the continuous evolution is never allowed to leave it.

12 Assuming Invariants

Let's make the dynamics more interesting and see what happens. Suppose there is a robot at a point with coordinates (x, y) that is facing in direction (d, e). Suppose the robot moves with constant (linear) velocity into direction (d, e), which is rotating as before. Then the corresponding dynamics is:

$$x' = d, y' = e, d' = e, e' = -d$$

because the derivative of the x coordinate is the component d of the direction and the derivative of the y coordinate is the component e of the direction. If the rotation of the direction (d, e) is faster or slower, the differential equation would be formed correspondingly. Consider the following conjecture:

$$(x-1)^2 + (y-2)^2 \geq p^2 \to [x' = d, y' = e, d' = e, e' = -d](x-1)^2 + (y-2)^2 \geq p^2 \tag{5}$$

This conjecture expresses that the robot at position (x, y) will always stay at distance p from the point $(1, 2)$ if it started there. Let's try to prove conjecture (5):

$$\mathrm{DI}\,\dfrac{\dfrac{\vdash 2(x-1)d + 2(y-2)e \geq 0}{\vdash (2(x-1)x' + 2(y-2)y' \geq 0)_{x'}^{d}{}_{y'}^{e}}}{(x-1)^2 + (y-2)^2 \geq p^2 \vdash [x' = d, y' = e, d' = e, e' = -d](x-1)^2 + (y-2)^2 \geq p^2}$$

Unfortunately, this differential invariant proof does not work. As a matter of fact, *fortunately* it does not work out, because conjecture (5) is not valid, so we will, fortunately, not be able to prove it with a sound proof technique. Conjecture (5) is too optimistic. Starting from some directions far far away, the robot will most certainly get too close to the point (1,2). Other directions may be fine.

Inspecting the above failed proof attempt, we could prove (5) if we knew something about the directions (d, e) that would make the remaining premise prove. What could that be?

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

Certainly, if we knew $d = e = 0$, the resulting premise would prove. Yet, that case is pretty boring because it corresponds to the point (x, y) being stuck forever. A more interesting case in which the premise would easily prove is if we knew $x - 1 = -e$ and $y - 2 = d$. In what sense could we "know" $x - 1 = -e \land y - 2 = d$? Certainly, we would have to assume this compatibility condition for directions versus position is true in the initial state, otherwise we would not necessarily know the condition holds true where we need it. So let's modify (5) to include this assumption:

$$x - 1 = -e \land y - 2 = d \land (x-1)^2 + (y-2)^2 \geq p^2 \rightarrow \\ [x' = d, y' = e, d' = e, e' = -d](x-1)^2 + (y-2)^2 \geq p^2 \quad (6)$$

Yet, where we need to know $x - 1 = -e \land y - 2 = d$ for the above sequent prove to continue is in the middle of the inductive step. How could we make that happen?

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

One step in the right direction is to convince ourselves that $x-1=-e\wedge y-2=d$ is a differential invariant of the dynamics, so it holds always if it held in the beginning:

$$
\begin{array}{rc}
 & * \\ \cline{2-2}
\mathbb{R} & \vdash d=-(-d)\wedge e=e \\ \cline{2-2}
 & \vdash (x'=-e'\wedge y'=d')^{d}_{x'}{}^{e}_{y'}{}^{e}_{d'}{}^{-d}_{e'} \\ \cline{2-2}
\text{DI} & x-1=-e\wedge y-2=d \vdash [x'=d,y'=e,d'=e,e'=-d](x-1=-e\wedge y-2=d)
\end{array}
$$

This proves easily using differential invariants.

Now, how can this freshly proved invariant $x-1=-e\wedge y-2=d$ be made available in the previous proof? Perhaps we could consider the conjunction of the invariant we want with the invariant we need:

$$(x-1)^2+(y-2)^2\geq p^2\wedge x-1=-e\wedge y-2=d$$

That does not work (eliding the antecedent in the conclusion just for space reasons)

$$
\begin{array}{rc}
 & \vdash 2(x-1)d+2(y-2)e\geq 0\wedge d=-(-d)\wedge e=e \\ \cline{2-2}
 & \vdash (2(x-1)x'+2(y-2)y'\geq 0\wedge x'=-e'\wedge y'=d')^{d}_{x'}{}^{e}_{y'}{}^{e}_{d'}{}^{-d}_{e'} \\ \cline{2-2}
\text{DI} & x-1=-e\ldots\vdash [x'=d,y'=e,d'=e,e'=-d]((x-1)^2+(y-2)^2\geq p^2\wedge x-1=-e\wedge y-2=d)
\end{array}
$$

because the differential invariant proof rule DI does not make the invariant F available in the antecedent of the premise.

In the case of loops, invariants can be assumed to hold before the loop body in the induction step.

$$(ind)\ \frac{F\vdash[\alpha]F}{F\vdash[\alpha^*]F}$$

By analogy, we could augment the differential invariant proof rule DI similarly to include F in the assumptions. Is that a good idea?

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

It looks tempting to suspect that rule DI could be improved by assuming the differential invariant F in the antecedent of the premise:

$$(DI_{??})\ \frac{H \land F \vdash F'^{\theta}_{x'}}{F \vdash [x' = \theta \,\&\, H]F} \quad \text{sound?}$$

After all, we really only care about staying safe when we are still safe. But implicit properties of differential equations are a subtle business. Assuming F like in rule $DI_{??}$ would, in fact, be unsound, as the following simple counterexample shows, which "proves" an invalid property using the unsound proof rule $DI_{??}$:

$$\text{↯}\ \frac{\dfrac{\dfrac{*\ \text{(unsound)}}{\vdash -(x-y)^2 \geq 0 \to -2(x-y)(1-y) \geq 0}}{\vdash -(x-y)^2 \geq 0 \to (-2(x-y)(x'-y') \geq 0)^{1}_{x'}{}^{y}_{y'}}}{-(x-y)^2 \geq 0 \vdash [x' = 1, y' = y](-(x-y)^2 \geq 0)}$$

Assuming an invariant of a differential equation during its own proof is, thus, incorrect, even though it has been suggested numerous times in the literature. There are some cases for which rule $DI_{??}$ would be sound, but these are nontrivial [Pla10a, Pla12b, Pla12a].

13 Differential Cuts

Instead, there is a complementary proof rule for *differential cuts* [Pla10a, Pla08, Pla12b, Pla12a] that can be used to strengthen assumptions in a sound way:

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

The differential cut rule works like a cut, but for differential equations. In the right premise, rule DC restricts the system evolution to the subdomain $H \land C$ of H, which changes the system dynamics but is a pseudo-restriction, because the left premise proves that C is an invariant anyhow (e.g. using rule DI). Note that rule DC is special in that it changes the dynamics of the system (it adds a constraint to the system evolution domain region), but it is still sound, because this change does not reduce the reachable set. The benefit of rule DC is that C will (soundly) be available as an extra assumption for all subsequent DI uses on the right premise (see, e.g., the use of the evolution domain constraint in Example 12). In particular, the differential cut rule DC can be used to strengthen the right premise with more and more auxiliary differential invariants C that will be available as extra assumptions on the right premise, once they have been proven to be differential invariants in the left premise.

Proving (6) in a sound way is now easy using a differential cut DC by $x - 1 = -e \land y - 2 = d$:

$$\text{DC}\,\dfrac{\text{DI}\,\dfrac{\dfrac{\mathbb{R}\,\dfrac{*}{\vdash d=-(-d)\land e=e}}{\vdash (x'=-e'\land y'=d')_{x'}^{d}{}_{y'}^{e}{}_{d'}^{e}{}_{e'}^{-d}}}{x{-}1{=}..\vdash [x'=d,\ldots](x{-}1{=}{-}e{\land}y{-}2{=}d)} \qquad \text{DI}\,\dfrac{\dfrac{\mathbb{R}\,\dfrac{*}{x{-}1{=}{-}e\land y{-}2{=}d\vdash 2(x-1)d+2(y-2)e\geq 0}}{x{-}1{=}{-}e\land y{-}2{=}d\vdash (2(x-1)x'+2(y-2)y'\geq 0)_{x'}^{d}{}_{y'}^{e}}}{(x{-}1)^2{+}(y{-}2)^2{\geq}p^2\vdash [x'=d,y'=e,d'=e,e'=-d\,\&\,x{-}1{=}{-}e{\land}y{-}2{=}d](x{-}1)^2{+}(y{-}2)^2{\geq}p^2}}{(x{-}1)^2{+}(y{-}2)^2{\geq}p^2{\land}x{-}1{=}{-}e{\land}y{-}2{=}d\vdash [x'=d,y'=e,d'=e,e'=-d](x-1)^2+(y-2)^2\geq p^2}$$

Using this differential cut process repeatedly has turned out to be extremely useful in practice and even simplifies the invariant search, because it leads to several simpler properties to find and prove instead of a single complex property [PC08, PC09, Pla10b].

Proof of Soundness of DC. For simplicity, consider only the case where $H \equiv true$. Rule DC is sound using the fact that the left premise implies that every solution φ that satisfies $x' = \theta$ also satisfies C *all along* the solution. Thus, if solution φ satisfies $x' = \theta$, it also satisfies $x' = \theta\,\&\,C$, so that the right premise entails the conclusion. The proof is accordingly for the case □

14 Differential Weakening

One simple but computable proof rule is *differential weakening*:

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

This rule is obviously sound, because the system $x' = \theta\,\&\,H$, by definition, can never leave H, hence, if H implies F (i.e. the region H is contained in the region F), then F is an invariant, no matter what $x' = \theta$ does. Unfortunately, this simple proof rule cannot prove very interesting properties, because it only works when H is very informative. It can, however, be useful in combination with stronger proof rules (e.g., differential cuts).

15 Summary

This lecture introduced very powerful proof rules for differential invariants, with which you can prove even complicated properties of differential equations in easy ways. Just like in the case of loops, where the search for invariants is nontrivial, differential invariants also require some smarts (or good automatic procedures) to be found. Yet, once a differential invariant has been identified, the proof follows easily.

Note 10 (Proof rules for differential equations)**.**

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

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

Exercises

Exercise 1. We have chosen to define

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

Prove that the following slightly relaxed definition would also give a sound proof rule for differential invariants:

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

Exercise 2. We have defined

$$(\theta \neq \eta)' \equiv ((\theta)' = (\eta)')$$

Suppose you remove this definition so that you can no longer use the differential invariant proof rule for formulas involving $\neq$. Can you derive a proof rule to prove such differential invariants regardless? If so, how? If not, why not?

References

[PC08] André Platzer and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. In Aarti Gupta and Sharad Malik, editors, *CAV*, volume 5123 of *LNCS*, pages 176–189. Springer, 2008. doi:10.1007/978-3-540-70545-1_17.

[PC09] André Platzer and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. *Form. Methods Syst. Des.*, 35(1):98–120, 2009. Special issue for selected papers from CAV'08. doi:10.1007/s10703-009-0079-8.

[Pla08] André Platzer. *Differential Dynamic Logics: Automated Theorem Proving for Hybrid Systems*. PhD thesis, Department of Computing Science, University of Oldenburg, Dec 2008. Appeared with Springer.

[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.

[Pla12a] André Platzer. A differential operator approach to equational differential invariants. In Lennart Beringer and Amy Felty, editors, *ITP*, volume 7406 of *LNCS*, pages 28–48. Springer, 2012. doi:10.1007/978-3-642-32347-8_3.

[Pla12b] 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.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Differential Invariants & Proof Theory

André Platzer

Carnegie Mellon University

Lecture 14

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] 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] made it possible to prove another property C of a differential equation and then change the dynamics of the system around so that it can never leave region C. Differential cuts turned out to be very useful when stacking inductive properties of differential equations on top of each other, so that easier properties are proved first and then assumed during the proof of the more complicated properties. Differential weakening (DW) [Pla10a] proves simple properties that are entailed by the evolution domain, which becomes especially useful after the evolution domain constraint has been augmented sufficiently by way of a differential cut.

Just like in the case of loops, where the search for invariants is nontrivial, differential invariants also require some smarts (or good automatic procedures) to be found. Once a differential invariant has been identified, the proof follows easily, which is a computationally attractive property.

Finding invariants of loops is very challenging. It can be shown to be the only fundamental challenge in proving safety properties of conventional discrete programs [HMP77]. Likewise, finding invariants and differential invariants is the only fundamental challenge in proving safety properties of hybrid systems [Pla08, Pla10b, Pla12a]. A more careful analysis even shows that just finding differential invariants is the only fundamental challenge for hybrid systems safety verification [Pla12a].

That is reassuring, because we know that the proofs will work[1] as soon as we find the right differential invariants. But it also tells us that we can expect the search for differential invariants (and invariants) to be challenging, because cyber-physical systems are extremely challenging, albeit very important.

Since, at the latest after this revelation, we fully realize the importance of studying and understanding differential invariants, we subscribe to developing a deeper understanding of differential invariants right away. The part of their understanding that today's lecture develops is how various classes of differential invariants relate to each other in terms of what they can prove. That is, are there properties that only differential invariants of the form $\mathcal{A}$ can prove, because differential invariants of the form $\mathcal{B}$ cannot prove them. Or are all properties provable by differential invariants of the form $\mathcal{A}$ also provable by differential invariants of the form $\mathcal{B}$.

These relations between classes of differential invariants tell us which forms of differential invariants we need to search for. A secondary goal of today's lecture besides this theoretical understanding is the practical understanding of developing more intuition about differential invariants and seeing them in action more thoroughly.

This lecture is based on [Pla12b]. In this lecture, we try to strike a balance between comprehensive handling of the subject matter and core intuition. This lecture will mostly focus on the core intuition of the heart of the proofs and leaves a more comprehensive argument and further study for articles [Pla12b]. Many proofs in this lecture are simplified and only prove the core argument, while leaving out other aspects. Those very important further details are beyond the scope of this course and can be found elsewhere [Pla12b]. For example, this lecture will not study whether indirect proofs could conclude the same properties. With a more careful analysis [Pla12b], it turns out that indirect proofs do not change the results reported in this lecture, but the proofs become significantly more complicated and require a more precise choice of the sequent calculus formulation. In this lecture, we will also not always prove all statements conjectured in a theorem. The remaining proofs can be found in the literature [Pla12b].

Note 1 (Proof theory of differential equations)**.** *The results in this lecture are part of the* proof theory *of differential equations. They are proofs about proofs, because they prove relations between the provability of logical formulas with different (sequent) proof calculi.*

2 Recap

Recall the following proof rules for differential equations from Lecture 11 on Differential Equations & Proofs:

[1] Although it may still be a lot of work in practice to make the proofs work. At least they become possible.

Note 2 (Proof rules for differential equations)**.**

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

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

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

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

3 Comparative Deductive Study

We study the relations of classes of differential invariants in terms of their relative deductive power. That is, we study whether some properties are only provable using differential invariant from the class $\mathcal{A}$, not using differential invariants from the class $\mathcal{B}$, or whether all properties provable with differential invariants from class $\mathcal{A}$ are also provable with class $\mathcal{B}$.

As a basis, we consider a propositional sequent calculus with logical cuts (which simplify glueing derivations together) and real-closed field arithmetic (we denote all uses by proof rule $\mathbb{R}$); see [Pla12b]. By $\mathcal{DI}$ we denote the proof calculus that, in addition, has general differential invariants (rule DI with arbitrary quantifier-free first-order formula F) but no differential cuts (rule DC). For a set $\Omega \subseteq \{\geq, >, =, \land, \lor\}$ of operators, we denote by $\mathcal{DI}_\Omega$ the proof calculus where the differential invariant F in rule DI is further restricted to the set of formulas that uses only the operators in Ω. For example, $\mathcal{DI}_{=,\land,\lor}$ is the proof calculus that allows only and/or-combinations of equations to be used as differential invariants. Likewise, $\mathcal{DI}_{\geq}$ is the proof calculus that only allows atomic weak inequalities $p \geq q$ to be used as differential invariants.

We consider classes of differential invariants and study their relations. If $\mathcal{A}$ and $\mathcal{B}$ are two classes of differential invariants, we write $\mathcal{A} \leq \mathcal{B}$ if all properties provable using differential invariants from $\mathcal{A}$ are also provable using differential invariants from $\mathcal{B}$. We write $\mathcal{A} \not\leq \mathcal{B}$ otherwise, i.e., when there is a valid property that can only be proven using differential invariants of $\mathcal{A} \setminus \mathcal{B}$. We write $\mathcal{A} \equiv \mathcal{B}$ if $\mathcal{A} \leq \mathcal{B}$ and $\mathcal{B} \leq \mathcal{A}$. We write $\mathcal{A} < \mathcal{B}$ if $\mathcal{A} \leq \mathcal{B}$ and $\mathcal{B} \not\leq \mathcal{A}$. Classes $\mathcal{A}$ and $\mathcal{B}$ are incomparable if $\mathcal{A} \not\leq \mathcal{B}$ and $\mathcal{B} \not\leq \mathcal{A}$.

4 Equivalences of Differential Invariants

First, we study whether there are equivalence transformations that preserve differential invariance. Every equivalence transformation that we have for differential invariant properties helps us with structuring the proof search space and also helps simplifying meta-proofs.

Lemma 1 (Differential invariants and propositional logic)**.** *Differential invariants are invariant under propositional equivalences. That is, if $F \leftrightarrow G$ is an instance of a propositional tautology then F is a differential invariant of $x' = \theta \,\&\, H$ if and only if G is.*

Proof. Let F be a differential invariant of a differential equation system $x' = \theta \,\&\, H$ and let G be a formula such that $F \leftrightarrow G$ is an instance of a propositional tautology. Then G is a differential invariant of $x' = \theta \,\&\, H$, because of the following formal proof:

$$\dfrac{\text{DI}\,\dfrac{\dfrac{*}{H \vdash {G'}^{\theta}_{x'}}}{G \vdash [x' = \theta \,\&\, H]G}}{F \vdash [x' = \theta \,\&\, H]F}$$

The bottom proof step is easy to see using (1), because precondition F implies the new precondition G and postcondition F is implied by the new postcondition G propositionally. Subgoal $H \vdash {G'}^{\theta}_{x'}$ is provable, because $H \vdash {F'}^{\theta}_{x'}$ is provable and G' is defined as a conjunction over all literals of G. The set of literals of G is identical to the set of literals of F, because the literals do not change by using propositional tautologies. Furthermore, we assumed a propositionally complete base calculus [Pla12b]. □

In subsequent proofs, we can use propositional equivalence transformations by Lemma 1. In the following, we will also implicitly use equivalence reasoning for pre- and postconditions as we have done in Lemma 1. Because of Lemma 1, we can, without loss of generality, work with arbitrary propositional normal forms for proof search.

5 Differential Invariants & Arithmetic

Not all logical equivalence transformations carry over to differential invariants. Differential invariance is not necessarily preserved under real arithmetic equivalence transformations.

Lemma 2 (Differential invariants and arithmetic)**.** *Differential invariants are* not *invariant under equivalences of real arithmetic. That is, if $F \leftrightarrow G$ is an instance of a first-order real arithmetic tautology then F may be a differential invariant of $x' = \theta \,\&\, H$ yet G may not.*

Proof. There are two formulas that are equivalent over first-order real arithmetic but, for the same differential equation, one of them is a differential invariant, the other one is not (because their differential structures differ). Since $5 \geq 0$, the formula $x^2 \leq 5^2$ is equivalent to $-5 \le x \wedge x \le 5$ in first-order real arithmetic. Nevertheless, $x^2 \le 5^2$ is a differential invariant of $x' = -x$ by the following formal proof:

$$\text{DI}\,\dfrac{\dfrac{\mathbb{R}\,\dfrac{*}{\vdash -2x^2 \le 0}}{\vdash (2xx' \le 0)_{x'}^{-x}}}{x^2 \le 5^2 \vdash [x' = -x]x^2 \le 5^2}$$

but $-5 \le x \wedge x \le 5$ is not a differential invariant of $x' = -x$:

$$\text{DI}\,\dfrac{\dfrac{\dfrac{\text{not valid}}{\vdash 0 \le -x \wedge -x \le 0}}{\vdash (0 \le x' \wedge x' \le 0)_{x'}^{-x}}}{-5 \le x \wedge x \le 5 \vdash [x' = -x](-5 \le x \wedge x \le 5)}$$

□

When we want to prove the property in the proof of Lemma 2, we need to use the principle (1) with the differential invariant $F \equiv x^2 \le 5^2$ and cannot use $-5 \le x \wedge x \le 5$.

By Lemma 2, we cannot just use arbitrary equivalences when investigating differential invariance, but have to be more careful. Not just the *elementary real arithmetical equivalence* of having the same set of satisfying assignments matters, but also the differential structures need to be compatible. Some equivalence transformations that preserve the solutions still destroy the differential structure. It is the equivalence of *real differential structures* that matters. Recall that differential structures are defined locally in terms of the behavior in neighborhoods of a point, not the point itself.

Lemma 2 illustrates a notable point about differential equations. Many different formulas characterize the same set of satisfying assignments. But not all of them have the same differential structure. Quadratic polynomials have inherently different differential structure than linear polynomials even when they have the same set of solutions over the reals. The differential structure is a more fine-grained information. This is similar to the fact that two elementary equivalent models of first-order logic can still be non-isomorphic. Both the set of satisfying assignments and the differential structure matter for differential invariance. In particular, there are many formulas with the same solutions but different differential structures. The formulas $x^2 \ge 0$ and $x^6 + x^4 - 16x^3 + 97x^2 - 252x + 262 \ge 0$ have the same solutions (all of $\mathbb{R}$), but very different differential structure; see Fig. 1.

The first two rows in Fig. 1 correspond to the polynomials from the latter two cases. The third row is a structurally different degree 6 polynomial with again the same set of solutions ($\mathbb{R}$) but a rather different differential structure. The differential structure also depends on what value x' assumes according to the differential equation. Fig. 1 illustrates that p' alone can already have a very different characteristic even if the respective sets of satisfying assignments of $p \ge 0$ are identical.

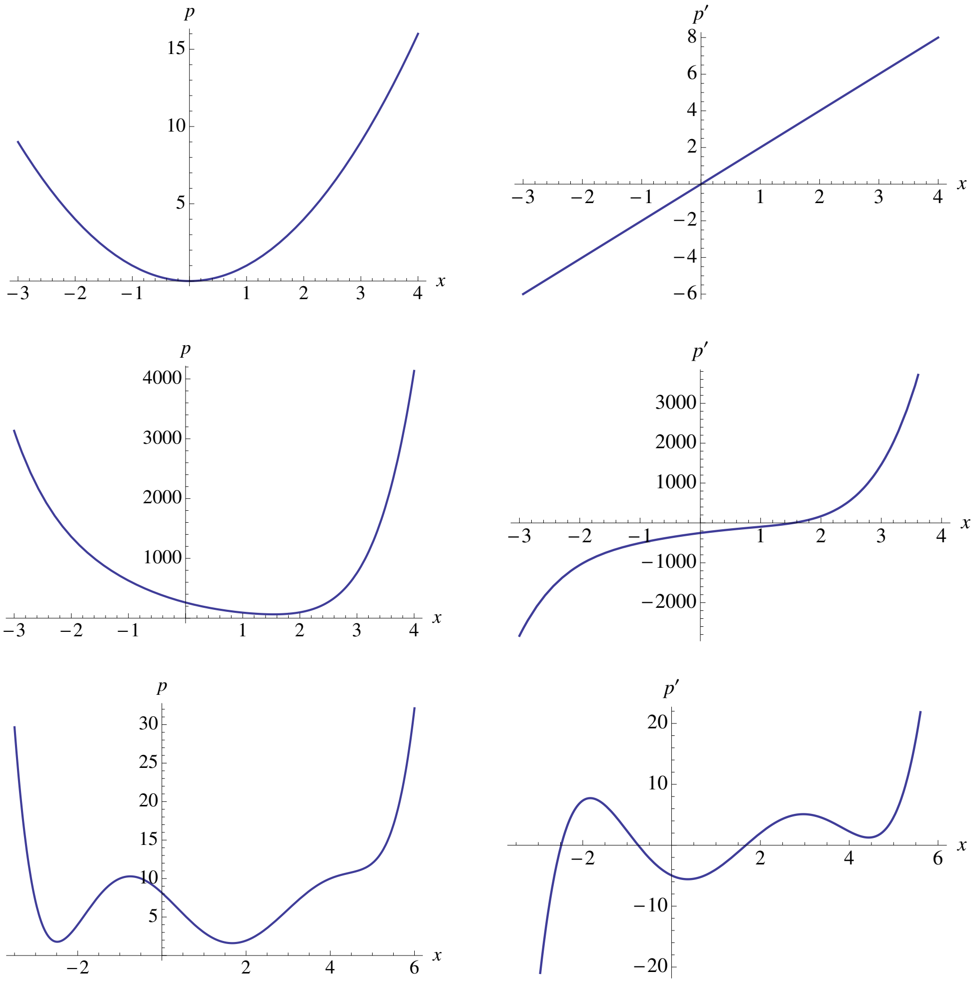


Figure 1: Equivalent solutions ($p \geq 0$ on the left) with different differential structure (p' plotted on the right)

We can, however, always normalize all atomic subformulas to have right-hand side 0, that is, of the form $p = 0, p \geq 0$, or $p > 0$. For instance, $p \leq q$ is a differential invariant if and only if $q - p \geq 0$ is, because $p \leq q$ is equivalent (in first-order real arithmetic) to $q - p \geq 0$ and, moreover, for any variable x and term θ, $(p' \leq q')^{\theta}_{x'}$ is equivalent to $(q' - p' \geq 0)^{\theta}_{x'}$ in first-order real arithmetic.

6 Differential Invariant Equations

For equational differential invariants, a.k.a. differential invariant equations, propositional operators do not add to the deductive power.

Proposition 3 (Equational deductive power [Pla10a, Pla12b])**.** *The deductive power of differential induction with atomic equations is identical to the deductive power of differential induction with propositional combinations of polynomial equations: That is, each formula is provable with propositional combinations of equations as differential invariants iff it is provable with only atomic equations as differential invariants:*

$$\mathcal{DI}_{=} \equiv \mathcal{DI}_{=,\wedge,\vee}$$

How could we prove that?

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

One direction is simple. Proving $\mathcal{DI}_{=} \leq \mathcal{DI}_{=,\wedge,\vee}$ is obvious, because every proof using a differential invariant equation $p_1 = p_2$ also is a proof using a propositional combination of differential invariant equations. The propositional combination that just consists of the only conjunct $p_1 = p_2$.

The other way around $\mathcal{DI}_{=} \geq \mathcal{DI}_{=,\wedge,\vee}$ is more difficult. If a formula can be proved using a differential invariant that is a propositional combination of equations, such as $p_1 = p_2 \wedge q_1 = q_2$, how could it possibly be proved using just a single equation?

Note 6 (Proofs of equal provability)**.** *A proof of Proposition 3 needs to show that every such provable property is also provable with a structurally simpler differential invariant. It effectively needs to transform proofs with propositional combinations of equations as differential invariants into proofs with just differential invariant equations. And, of course, the proof of Proposition 3 needs to prove that the resulting equations are actually provably differential invariants and prove the same properties as before.*

Proof of Proposition 3. Let $x' = \theta$ be the (vectorial) differential equation to consider. We show that every differential invariant that is a propositional combination F of polynomial equations is expressible as a single atomic polynomial equation (the converse inclusion is obvious). We can assume F to be in negation normal form by Lemma 1 (recall that negations are resolved and $\neq$ can be assumed not to appear). Then we reduce F inductively to a single equation using the following transformations:

- If F is of the form $p_1 = p_2 \vee q_1 = q_2$, then F is equivalent to the single equation $(p_1 - p_2)(q_1 - q_2) = 0$. Furthermore, $F'^{\theta}_{x'} \equiv (p_1' = p_2' \wedge q_1' = q_2')^{\theta}_{x'}$ directly implies
$$\big(((p_1 - p_2)(q_1 - q_2))' = 0\big)^{\theta}_{x'} \equiv \big((p_1' - p_2')(q_1 - q_2) + (p_1 - p_2)(q_1' - q_2') = 0\big)^{\theta}_{x'}$$

- If F is of the form $p_1 = p_2 \wedge q_1 = q_2$, then F is equivalent to the single equation $(p_1 - p_2)^2 + (q_1 - q_2)^2 = 0$. Furthermore, $F'^{\theta}_{x'} \equiv \big(p_1' = p_2' \wedge q_1' = q_2'\big)^{\theta}_{x'}$ implies
$$\Big(\big((p_1 - p_2)^2 + (q_1 - q_2)^2\big)' = 0\Big)^{\theta}_{x'} \equiv \big(2(p_1 - p_2)(p_1' - p_2') + 2(q_1 - q_2)(q_1' - q_2') = 0\big)^{\theta}_{x'}$$
□

Note that the polynomial degree increases quadratically by the reduction in Proposition 3, but, as a trade-off, the propositional structure simplifies. Consequently, differential invariant search for the equational case can either exploit propositional structure with lower degree polynomials or suppress the propositional structure at the expense of higher degrees.

7 Equational Incompleteness

Focusing exclusively on differential invariants with equations, however, reduces the deductive power, because sometimes only differential invariant inequalities can prove properties.

Proposition 4 (Equational incompleteness). *The deductive power of differential induction with equational formulas is strictly less than the deductive power of general differential induction, because some inequalities cannot be proven with equations.*

$$\begin{aligned} \mathcal{DI}_{=} \equiv \mathcal{DI}_{=,\land,\lor} &< \mathcal{DI} \\ \mathcal{DI}_{\geq} &\not\leq \mathcal{DI}_{=} \equiv \mathcal{DI}_{=,\land,\lor} \\ \mathcal{DI}_{>} &\not\leq \mathcal{DI}_{=} \equiv \mathcal{DI}_{=,\land,\lor} \end{aligned}$$

How could such a proposition be proved?

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

The proof strategy in Proposition 3 involved transforming proofs into proofs to prove the inclusion $\mathcal{DI}_{=} \geq \mathcal{DI}_{=,\wedge,\vee}$. Could the same strategy prove Proposition 4? No, because we need to show the opposite! Proposition 4 conjectures $\mathcal{DI}_{\geq} \not\leq \mathcal{DI}_{=,\wedge,\vee}$, which means that there are true properties that are only provable using a differential invariant inequality $p_1 \geq p_2$ and not using any differential invariant equations or propositional combinations thereof.

For one thing, this means that we ought to find a property that a differential invariant inequality can prove. That ought to be easy enough, because Lecture 11 on Differential Equations & Proofs showed us how useful differential invariants are. But then a proof of Proposition 4 also requires a proof why that very same formula cannot possibly ever be proved with any way of using differential invariant equations or their propositional combinations. That is a proof about nonprovability. Proving provability in proof theory amounts to producing a proof (in sequent calculus). Proving nonprovability most certainly does not mean it would be enough to write something down that is not a proof. After all, just because one proof attempt fails does not mean that others would not be successful. You have experienced this while you were working on proving your labs for this course. The first proof attempt might have failed miserably and was impossible to ever work out. But, come next day, you had a better idea with a different proof, and suddenly the same property turned out to be provable even if the first proof attempt failed.

How could we prove that all proof attempts do not work?

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

One way of showing that a logical formula cannot be proved is by giving a counterexample, i.e. a state which assigns values to the variables that falsify the formula. That is, of course, not what can help us proving Proposition 4, because a proof of Proposition 4 requires us to find a formula that can be proved with $\mathcal{DI}_{\geq}$ (so it cannot have a counterexample, since it is valid), just cannot be proved with $\mathcal{DI}_{=,\wedge,\vee}$. Proving that a valid formula cannot be proved with $\mathcal{DI}_{=,\wedge,\vee}$ requires us to show that all proofs in $\mathcal{DI}_{=,\wedge,\vee}$ do not prove that formula.

By analogy, recall sets. The way to prove that two sets M, N have the same "number" of elements is to come up with a pair of functions $\Phi : M \to N$ and $\Psi : N \to M$ between the sets and then prove that Φ, Ψ are inverses of each other, i.e. $\Phi(\Psi(y)) = y$ and $\Psi(\Phi(x)) = x$ for all $x \in M, y \in N$. Proving that two sets M, N do not have the same "number" of elements works entirely differently, because that has to prove for all pairs of functions $\Phi : M \to N$ and $\Psi : N \to M$ that there is is an $x \in M$ such that $\Psi(\Phi(x)) \neq x$ or an $y \in N$ such that $\Phi(\Psi(y)) \neq y$. Since that is a lot of work, indirect criteria such as cardinality or countability are often used instead, e.g. for proving that the reals $\mathbb{R}$ and rationals $\mathbb{Q}$ do not have the same number of elements, because $\mathbb{Q}$ are countable but $\mathbb{R}$ are not (by Cantor's diagonal argument).

By analogy, recall vector spaces from linear algebra. The way to prove that two vector spaces V, W are isomorphic is to think hard and construct a function $\Phi : V \to W$ and a function $\Psi : W \to V$ and then prove that Φ, Ψ are linear functions and inverses of each other. Proving that two vector spaces V, W are *not* isomorphic works entirely differently, because that has to prove that all pairs of functions $\Phi : V \to W$ and $\Psi : W \to V$ are either not linear or not inverses of each other. Proving the latter literally is a lot of work. So instead, indirect criteria are being used. One proof that V, W are not isomorphic could show that both have different dimensions and then prove that isomorphic vector spaces always have the same dimension, so V and W cannot possibly be isomorphic.

Consequently, proving non-provability leads to a study of indirect criteria about proofs of differential equations.

Note 8 (Proofs of different provability)**.** *Proving non-reducibility $\mathcal{A} \not\leq \mathcal{B}$ for classes of differential invariants requires an example formula ϕ that is provable in $\mathcal{A}$ plus a proof that no proof using $\mathcal{B}$ proves ϕ. The preferred way of doing that is finding an indirect criterion that all proofs in $\mathcal{B}$ possess but that ϕ does not have.*

Proof of Proposition 4. Consider any term $a > 0$ (e.g., 5 or $x^2 + 1$ or $x^2 + x^4 + 2$). The following formula is provable by differential induction with the weak inequality $x \geq 0$:

$$\dfrac{\mathbb{R}\dfrac{*}{\vdash a \geq 0}}{x \geq 0 \vdash [x' = a]x \geq 0}\text{DI}$$

It is not provable with an equational differential invariant. Any univariate polynomial p that is zero on $x \geq 0$ is the zero polynomial and, thus, $p = 0$ cannot be equivalent to the half space $x \geq 0$. By the equational deductive power theorem 3, the above formula then is not provable with any Boolean combination of equations as differential invariant either.

The other parts of the theorem are proved elsewhere [Pla12b]. □

It might be tempting to think that at least equational postconditions only need equational differential invariants for proving them. But that is not the case either [Pla12b].

8 Strict Differential Invariant Inequalities

We show that, conversely, focusing on strict inequalities also reduces the deductive power, because equations are obviously missing and there is at least one proof where this matters. That is, strict barrier certificates do not prove (nontrivial) closed invariants.

Formal definitions of open and closed sets come from real analysis (or topology). Roughly: A closed set is one whose boundary belongs to the set. For example the solid unit disk. An open set is one whose boundary does not belong to the set, for example the unit disk without the circle of radius 1.

Proposition 5 (Strict barrier incompleteness)**.** *The deductive power of differential induction with strict barrier certificates (formulas of the form $p > 0$) is strictly less than the deductive power of general differential induction.*

$$\mathcal{DI}_{>} < \mathcal{DI}$$
$$\mathcal{DI}_{=} \not\leq \mathcal{DI}_{>}$$

Proof. The following formula is provable by equational differential induction:

$$\text{DI}\ \dfrac{\mathbb{R}\ \dfrac{*}{\vdash 2xy + 2y(-x) = 0}}{x^2 + y^2 = c^2 \vdash [x' = y, y' = -x]x^2 + y^2 = c^2}$$

But it is not provable with a differential invariant of the form $p > 0$. An invariant of the form $p > 0$ describes an open set and, thus, cannot be equivalent to the (nontrivial) closed domain where $x^2 + y^2 = c^2$. The only sets that are both open and closed in $\mathbb{R}^n$ are $\emptyset$ and $\mathbb{R}^n$.

The other parts of the theorem are proved elsewhere [Pla12b]. □

9 Differential Invariant Equations as Differential Invariant Inequalities

Weak inequalities, however, do subsume the deductive power of equational differential invariants. This is obvious on the algebraic level but we will see that it also does carry over to the differential structure.

Proposition 6 (Equational definability). *The deductive power of differential induction with equations is subsumed by the deductive power of differential induction with weak inequalities:*

$$\mathcal{DI}_{=,\wedge,\vee} \leq \mathcal{DI}_{\geq}$$

Proof. By Proposition 3, we only need to show that $\mathcal{DI}_{=} \leq \mathcal{DI}_{\geq}$. Let $p = 0$ be an equational differential invariant of a differential equation $x' = \theta \,\&\, H$. Then we can prove the following:

$$\dfrac{\dfrac{*}{H \vdash (p' = 0)^{\theta}_{x'}}}{p = 0 \vdash [x' = \theta \,\&\, H]p = 0}\,{\scriptstyle\text{DI}}$$

Then, the inequality $-p^2 \geq 0$, which is equivalent to $p = 0$ in real arithmetic, also is a differential invariant of the same dynamics by the following formal proof:

$$\dfrac{\dfrac{*}{H \vdash (-2pp' \geq 0)^{\theta}_{x'}}}{-p^2 \geq 0 \vdash [x' = \theta \,\&\, H](-p^2 \geq 0)}\,{\scriptstyle\text{DI}}$$

The subgoal for the differential induction step is provable: if we can prove that H implies $(p' = 0)^{\theta}_{x'}$, then we can also prove that H implies $(-2pp' \geq 0)^{\theta}_{x'}$, because $(p' = 0)^{\theta}_{x'}$ implies $(-2pp' \geq 0)^{\theta}_{x'}$ in first-order real arithmetic. □

Note that the local state-based view of differential invariants is crucial to make the last proof work. By Proposition 6, differential invariant search with weak inequalities can suppress equations. Note, however, that the polynomial degree increases quadratically with the reduction in Proposition 6. In particular, the polynomial degree increases quartically when using the reductions in Proposition 3 and Proposition 6 one after another to turn propositional equational formulas into single inequalities. This quartic increase of the polynomial degree is likely a too serious computational burden for practical purposes even if it is a valid reduction in theory.

10 Differential Invariant Atoms

Next we see that, with the notable exception of pure equations (Proposition 3), propositional operators increase the deductive power.

Theorem 7 (Atomic incompleteness)**.** *The deductive power of differential induction with propositional combinations of inequalities exceeds the deductive power of differential induction with atomic inequalities.*

$$\mathcal{DI}_{\geq} < \mathcal{DI}_{\geq,\land,\lor}$$
$$\mathcal{DI}_{>} < \mathcal{DI}_{>,\land,\lor}$$

Proof. Consider any term $a \geq 0$ (e.g., 1 or x^2+1 or x^2+x^4+1 or $(x-y)^2+2$). Then the formula $x \geq 0 \land y \geq 0 \rightarrow [x' = a, y' = y^2](x \geq 0 \land y \geq 0)$ is provable using a conjunction in the differential invariant:

$$\text{DI}\,\dfrac{\dfrac{\mathbb{R}\,\dfrac{*}{\vdash a \geq 0 \land y^2 \geq 0}}{\vdash (x' \geq 0 \land y' \geq 0)_{x'\;y'}^{a\;\,y^2}}}{x \geq 0 \land y \geq 0 \vdash [x' = a, y' = y^2](x \geq 0 \land y \geq 0)}$$

By a sign argument similar to that in the proof of [Pla10a, Theorem 2] no atomic formula is equivalent to $x \geq 0 \land y \geq 0$. Thus, the above property cannot be proven using a single differential induction. The proof for a postcondition $x > 0 \land y > 0$ is similar.

The other—quite substantial—parts of the proof are proved elsewhere [Pla12b]. □

Note that the formula in the proof of Theorem 7 is provable, e.g., using differential cuts (DC) with two atomic differential induction steps, one for $x \geq 0$ and one for $y \geq 0$. Yet, a similar argument can be made to show that the deductive power of differential induction with atomic formulas (even when using differential cuts) is strictly less than the deductive power of general differential induction; see [Pla10a, Theorem 2].

11 Summary

Fig. 2 summarizes the findings of this lecture and others reported in the literature [Pla12b]. We have considered the differential invariance problem, which, by a relative completeness argument [Pla12a], is at the heart of hybrid systems verification. To better understand structural properties of hybrid systems, we have identified and analyzed more than a dozen (16) relations between the deductive power of several (9) classes of differential invariants, including subclasses that correspond to related approaches.

Our results require a symbiosis of elements of logic with real arithmetical, differential, semialgebraic, and geometrical properties. Future work includes investigating this new field further that we call *real differential semialgebraic geometry*, whose development has only just begun.

$$\begin{tikzcd}
\mathcal{DI}_{\geq} \arrow[rrr, hook] \arrow[dd, squiggly, no head] & & & \mathcal{DI}_{\geq,\wedge,\vee} \arrow[r, equal] \arrow[dd, squiggly, no head] & \mathcal{DI}_{\geq,=,\wedge,\vee} \arrow[dr, hook] \arrow[dd, squiggly, no head] & \\
& \mathcal{DI}_{=} \arrow[ul, hook] \arrow[r, equal] \arrow[dl, no head] & \mathcal{DI}_{=,\wedge,\vee} \arrow[ur, hook] \arrow[rrr, hook] \arrow[drr, hook, bend left] \arrow[dr, no head] & & & \mathcal{DI} \\
\mathcal{DI}_{>} \arrow[rrr, hook] & & & \mathcal{DI}_{>,\wedge,\vee} \arrow[r, hook] & \mathcal{DI}_{>,=,\wedge,\vee} \arrow[ur, hook] &
\end{tikzcd}$$

Figure 2: Differential invariance chart

Exercises

Exercise 1. Prove the relation $\mathcal{DI}_{>} \leq \mathcal{DI}_{>,\wedge,\vee}$.

Exercise 2. Prove the relation $\mathcal{DI}_{\geq} \equiv \mathcal{DI}_{\leq,\wedge,\vee}$.

Exercise 3. Prove the relation $\mathcal{DI}_{\geq,\wedge,\vee} \equiv \mathcal{DI}_{\geq,=,\wedge,\vee}$.

Exercise 4. Prove the relation $\mathcal{DI}_{=,\wedge,\vee} < \mathcal{DI}_{\geq,\wedge,\vee}$.

Exercise 5. Prove the relation $\mathcal{DI}_{>,\wedge,\vee} < \mathcal{DI}_{>,=,\wedge,\vee}$.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Ghosts & Differential Ghosts

André Platzer

Carnegie Mellon University
Lecture 15

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] 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] made it possible to prove another property C of a differential equation and then change the dynamics of the system around so that it can never leave region C. Lecture 14 on Differential Invariants & Proof Theory studied some part of the proof theory of differential equations and proved the differential invariance chart that compares the deductive power of classes of differential invariants; see Fig. 1.

$$\begin{tikzcd}
\mathcal{DI}_{\geq} \arrow[rrr, hook] \arrow[dd, squiggly, no head] & & & \mathcal{DI}_{\geq,\land,\lor} \arrow[r, equal] \arrow[dd, squiggly, no head] & \mathcal{DI}_{\geq,=,\land,\lor} \arrow[dr, hook] \arrow[dd, squiggly, no head] & \\
& \mathcal{DI}_{=} \arrow[ul] \arrow[r, equal] \arrow[dl, no head] & \mathcal{DI}_{=,\land,\lor} \arrow[ur] \arrow[rrr, hook] \arrow[drr, bend right] \arrow[dr, no head] & & & \mathcal{DI} \\
\mathcal{DI}_{>} \arrow[rrr, hook] & & & \mathcal{DI}_{>,\land,\lor} \arrow[r, hook] & \mathcal{DI}_{>,=,\land,\lor} \arrow[ur, hook] &
\end{tikzcd}$$

Figure 1: Differential invariance chart

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.

Yet, it can also be shown that there are properties where 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.

This lecture is based on [Pla12, Pla10b].

2 Recap

Recall the following proof rules for differential equations from Lecture 11 on Differential Equations & Proofs:

Note 1 (Proof rules for differential equations).

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

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

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

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

3 Arithmetic Ghosts

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

where $q := *$ is the nondeterministic assignment that assigns an arbitrary real number to q.

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

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

4 Nondeterministic Assignments & Ghosts of Choice

The HP statement $x := *$ 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 add the nondeterministic assignment $x := *$ as a statement to the syntax of hybrid programs.

$$\alpha, \beta ::= x := \theta \mid \ldots \mid x := *$$

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 finally need proof rules so that they can be handled in proofs.

$$(\langle{:}*\rangle)\ \frac{\exists x\, \phi}{\langle x := *\rangle \phi} \qquad ([{:}*])\ \frac{\forall x\, \phi}{[x := *]\phi}$$

Another approach for adding nondeterministic assignments $x := *$ to hybrid programs is to consider whether we even have to do that. 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.

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 [Pla13]:

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

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, it becomes clear that both sides of (3) mean the same. Hence, the above definition (3) capture the right concept. 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 less so 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 $x := *$ (indicated by $\not\equiv$):

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

5 Differential-algebraic Ghosts

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

See [Pla10b, Chapter 3] for the meaning of the nondeterministic differential equation $q' = *$.[1]

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

[1] 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.

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 $\frac{b}{c}$.

Together with the reduction of divisions in discrete assignments from Sect. 3, plus the inside that divisions in tests and evolution domain constraints can always be rewritten to division-free form, is a (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 singularities.

6 Discrete Ghosts

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

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

where y is a new program variable.

That proof rule IA is sound can be argued 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.

7 Remember the Bouncing Ball

Recall the following sequent for the bouncing ball from Lecture 7 on Control Loops & Invariants, which was based on an argument in Lecture 4 on Safety & Contracts.

$$2gh = 2gH - v^2 \land h \geq 0 \rightarrow [h' = v, v' = -g \,\&\, h \geq 0](2gh = 2gH - v^2 \land h \geq 0) \tag{4}$$

The d$\mathcal{L}$ formula (4) can be proved using the solutions of the differential equation with proof rule [′]. d$\mathcal{L}$ formula (4) can also be proved using differential invariants, with a differential cut and a use of differential weakening:

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

Note that differential weakening (DW) works for proving the postcondition $h \geq 0$, but DI would not work, because the derivative of $h \geq 0$ is $v \geq 0$, which is not an invariant of the bouncing ball since its velocity ultimately becomes negative when it is falling according to gravity. Note that this proofs is very elegant and has notably easier arithmetic than the arithmetic we ran into when working with solutions of the bouncing ball in earlier lectures.

The reason why this proof worked so elegantly is that the invariant $2gh = 2gH - v^2 \land h \geq 0$ was a very good choice that we came up with in a clever way in Lecture 4. 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 do use solutions for differential equations without invoking rule [$'$]?

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

8 Differential Ghosts

$$2gh = 2gH - v^2 \vdash [h'' = -g \,\&\, h \geq 0](2gh = 2gH - v^2 \land h \geq 0)$$

Use the usual abbreviations:

$$A_{h,v} \overset{\text{def}}{\equiv} 2gh = 2gH - v^2$$
$$B_{h,v} \overset{\text{def}}{\equiv} 2gh = 2gH - v^2 \land h \geq 0$$
$$(h'' = -g) \overset{\text{def}}{\equiv} (h' = v, v' = -g)$$

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

where the proof step marked DA omits the (here trivial) left premise of rule DA, which proves because $B_{h,v} \leftrightarrow \exists t\, B_{h,v}$ is trivially valid in first-order logic, as the fresh t does even occur in $B_{h,v}$ here.

The right premise in the above proof proves as follows

$$\text{IA}\,\dfrac{\text{DC}\,\dfrac{\text{DI}\,\dfrac{\dfrac{ax\,\dfrac{*}{h \geq 0 \land v = v_0 - tg \vdash v = v_0 - 2\frac{g}{2}t}}{h \geq 0 \land v = v_0 - tg \vdash (h' = v_0t' - 2\frac{g}{2}tt')_{h'}^{v}{}_{t'}^{1}}}{A_{h,v} \vdash \{h_0 := h, v_0 := v\}[h'' = -g, t' = 1 \,\&\, h \geq 0 \land v = v_0 - tg]h = h_0 + v_0t - \frac{g}{2}t^2}\,\triangleright}{A_{h,v} \vdash \{h_0 := h, v_0 := v\}[h'' = -g, t' = 1 \,\&\, h \geq 0 \land v = v_0 - tg]B_{h,v}}}{A_{h,v} \vdash \{v_0 := v\}[h'' = -g, t' = 1 \,\&\, h \geq 0 \land v = v_0 - tg]B_{h,v}}$$

The proof step marked DC has a second premise which is elided (marked by ▷) and proves as follows:

$$\text{DW}\,\dfrac{\mathbb{R}\,\dfrac{*}{h \geq 0 \land v = v_0 - tg \land h = h_0 + v_0t - \frac{g}{2}t^2 \vdash B_{h,v}}}{A_{h,v} \vdash \{h_0 := h, v_0 := v\}[h'' = -g, t' = 1 \,\&\, h \geq 0 \land v = v_0 - tg \land h = h_0 + v_0t - \frac{g}{2}t^2]B_{h,v}}$$

The arithmetic (marked $\mathbb{R}$) can be proved with enough care, but it has a twist! First of all, the arithmetic can be simplified substantially using the equality substitution rule =r and subsequent weakening.

$$\text{=r}\,\dfrac{\land\text{l,Wr}\,\dfrac{\land\text{r}\,\dfrac{\text{Wl}\,\dfrac{\vdash 2g(h_0 + v_0t - \frac{g}{2}t^2) = 2gH - (v_0 - tg)^2}{h \geq 0 \vdash 2g(h_0 + v_0t - \frac{g}{2}t^2) = 2gH - (v_0 - tg)^2} \quad ax\,\dfrac{*}{h \geq 0 \vdash h \geq 0}}{h \geq 0 \vdash 2g(h_0 + v_0t - \frac{g}{2}t^2) = 2gH - (v_0 - tg)^2 \land h \geq 0}}{h \geq 0 \land v = v_0 - tg \land h = h_0 + v_0t - \frac{g}{2}t^2 \vdash 2g(h_0 + v_0t - \frac{g}{2}t^2) = 2gH - (v_0 - tg)^2 \land h \geq 0}}{h \geq 0 \land v = v_0 - tg \land h = h_0 + v_0t - \frac{g}{2}t^2 \vdash 2gh = 2gH - v^2 \land h \geq 0}$$

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. The arithmetic in the left branch

$$2g(h_0 + v_0t - \frac{g}{2}t^2) = 2gH - (v_0 - tg)^2$$

expands by polynomial arithmetic and cancels as follows

$$2g(h_0 + \cancel{v_0t} - \cancel{\tfrac{g}{2}t^2}) = 2gH - v_0^2 + \cancel{2v_0tg} + \cancel{t^2g^2}$$

That leaves the remaining condition

$$2gh_0 = 2gH - v_0^2$$

Indeed, this relation characterizes exactly how H, which turns out to have been the maximal height, relates to the initial height h_0 and initial velocity v_0. In the case of initial velocity $v_0 = 0$, this relation collapses to $h_0 = H$.

For the case of the bouncing ball, this proof was unnecessarily complicated, because the solution rule [′] could have been used instead. But the same proof technique 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.

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

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

proves

where y new and $y' = \eta, y(0) = y_0$ has a solution $y : [0, \infty) \to \mathbb{R}^n$ 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 ϕ implies ψ for some choice of y (left premise). Yet, for any y, ψ is an invariant of the extended dynamics (right premise). Thus, ψ always holds after the evolution for some y (its value can be different than in the initial state), which still implies ϕ (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 variable y with a suitable differential equation ($y' = \eta$).

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

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

9 Axiomatic Ghosts

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

$$x_1' = v\cos\vartheta \qquad x_2' = v\sin\vartheta \qquad \vartheta' = \omega. \tag{5}$$

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.

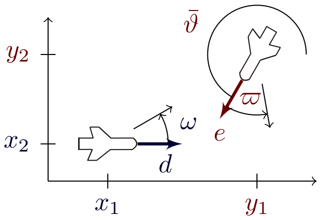


Figure 2: Aircraft dynamics

Unlike for straight-line flight ($\omega = 0$), the nonlinear dynamics in (5) is difficult to analyse [TPS98] for curved flight ($\omega \neq 0$), especially due to the trigonometric expressions which are generally undecidable. Solving (5) 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. 2 and 3. Substituting this coordinate change into differential equations (5), 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 (5) by simple symbolic differentiation:

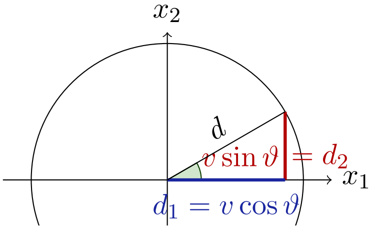


Figure 3: Reparametrize for differential axiomatization

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

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 flight [TPS98, LLL00] so that angular velocity ω is the primary control parameter in air traffic control. Hence, equations (5) can be restated as the following differential equation $\mathcal{F}(\omega)$:

$$x_1' = d_1 \,,\; x_2' = d_2 \,,\; d_1' = -\omega d_2 \,,\; d_2' = \omega d_1 \qquad (\mathcal{F}(\omega))$$

$$y_1' = e_1 \,,\; y_2' = e_2 \,,\; e_1' = -\varpi e_2 \,,\; e_2' = \varpi e_1 \qquad (\mathcal{G}(\varpi))$$

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. 2) and angular velocity ϖ corresponds to the differential equation $\mathcal{F}(\omega), \mathcal{G}(\varpi)$. 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]. Our proof calculus in this chapter works with the differential equations themselves and not with their solutions, so that differential axiomatization helps.

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

10 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. Ghosts, in their various forms, let us achieve that by adding auxiliary variables into the system dynamics. Sometimes such ghosts are even necessary to prove properties. Although, as a workaround, it is also sometimes possible to rewrite the original model so that it already includes the ghost variables. The phenomenon that relations between state and ghost variables are sometimes easier to prove than just properties of state variables applies in either case. A secondary goal of today's lecture is, again, developing more intuition and deeper understandings of differential invariants and differential cuts.

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.

[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.

[Pla13] André Platzer. A complete axiomatization of differential game logic for hybrid games. Technical Report CMU-CS-13-100R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January, Revised and extended in July 2013.

[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.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Differential & Temporal Logics

André Platzer

Carnegie Mellon University

Lecture 16

1 Introduction

This course is devoted to the study of the Foundations of Cyber-Physical Systems [Pla12c, Pla12b]. Lecture 3 on Choice & Control explained hybrid programs, a program notation for hybrid systems [Pla08, Pla10, Pla12c, Pla12a]. Lecture 4 on Safety & Contracts defined differential dynamic logic [Pla08, Pla10, Pla12c, Pla12a] as a specification and verification logic for hybrid programs. Lecture 5 on Dynamical Systems & Dynamic Axioms and subsequent lectures studied proof principles for differential dynamic logic with which we can prove correctness properties of hybrid systems. In your labs, you have demonstrated aptly how you can model, specify, and verify quite sophisticated and challenging robots.

Yet, there was one rather puzzling phenomenon that we noticed in Lecture 4 only then did not have a chance to consider any further. For a hybrid program α and differential dynamic logic formula ϕ, the modal formula

$$[\alpha]\phi$$

expresses that all *final* states reached by all runs of α satisfy the logical formula ϕ. The modal formula $[\alpha]\phi$ is, consequently, false exactly in those states from which α can reach a final state that violates the safety condition ϕ. Yet, what about states from which the final state reached by running α is safe but some intermediate state along the execution of α was not safe?

Shouldn't systems that violate safety conditino ϕ at an intermediate state be considered unsafe as well?

The short answer is: that depends.

Does it even make a difference whether we study intermediate states as well or only worry about final states?

The short answer is again: that depends.

What exactly it depends on and how to systematically approach the general case of safety *throughout* the system execution is what today's lecture studies. The key to the answer will be understanding the temporal behavior of hybrid programs. The hybrid trace semantics of hybrid programs will also give us a deeper understanding of the hybrid aspect of time in hybrid systems.

This lecture is based on [Pla10, Chapter 4], which is a significant extension of [Pla07].

2 Temporalizing Hybrid Systems

In order to be able to distinguish whether a CPS is safe at the end of its run or safe always throughout its run, differential dynamic logic d$\mathcal{L}$ will be extended with additional temporal modalities. The resulting logic extends d$\mathcal{L}$ and is called *differential temporal dynamic logic* (dTL) [Pla10, Chapter 4]. The modal formula

$$[\alpha]\phi$$

of d$\mathcal{L}$ [Pla08, Pla12c] expresses that all *final* states reached by all runs of α satisfy the logical formula ϕ. The same d$\mathcal{L}$ formula $[\alpha]\phi$ is allowed in the logic dTL and has the same semantics [Pla10, Chapter 4]. The new temporal modal dTL formula

$$[\alpha]\Box\phi$$

instead, expresses that all states reached *all along* all traces of α satisfy ϕ. Those two modalities can be be used to distinguish systems that are always throughout from those that are only safe in final states. For example, if the dTL formula

$$[\alpha]\phi \land \neg[\alpha]\Box\phi$$

is true in an initial state ν, then the system α will be safe (in the sense of satisfying ϕ) in all final states reached after running α from ν, but is not safe always throughout all traces of all runs of α from ν. Can that happen?

You should try to answer this question before it is discussed in a later part of these lecture notes.

3 Syntax of Differential Temporal Dynamic Logic

The *differential temporal dynamic logic* dTL extends differential dynamic logic [Pla08, Pla10, Pla12c] with temporal modalities for verifying temporal specifications of hybrid systems. Hence, dTL has two kinds of modalities:

Modal operators. Modalities of dynamic logic express statements about all possible behaviour ($[\alpha]\pi$) of a system α, or about the existence of a trace ($\langle\alpha\rangle\pi$), satisfying condition π. Unlike in standard dynamic logic, α is a model of a hybrid system.

The logic dTL uses hybrid programs to describe α as in previous lectures. Yet, unlike in standard dynamic logic [HKT00] or d$\mathcal{L}$, π is a *trace formula* in dTL, and π can refer to all states that occur *during* a trace using temporal operators.

Temporal operators. For dTL, the temporal trace formula $\Box\phi$ expresses that the formula ϕ holds all along a trace selected by $[\alpha]$ or $\langle\alpha\rangle$. For instance, the state formula $\langle\alpha\rangle\Box\phi$ says that the state formula ϕ holds at every state along at least one trace of α. Dually, the trace formula $\Diamond\phi$ expresses that ϕ holds at some point during such a trace. It can occur in a state formula $\langle\alpha\rangle\Diamond\phi$ to express that there is such a state in some trace of α, or as $[\alpha]\Diamond\phi$ to say that along each trace there is a state satisfying ϕ. The primary focus of attention in today's lecture is on homogeneous combinations of path and trace quantifiers like $[\alpha]\Box\phi$ or $\langle\alpha\rangle\Diamond\phi$.

The formulas of dTL are defined similarly to differential dynamic logic. However, the modalities $[\alpha]$ and $\langle\alpha\rangle$ accept trace formulas that refer to the temporal behavior of *all* states along a trace. Inspired by CTL and CTL* [EC82, EH86], dTL distinguishes between state formulas, which are true or false in states, and trace formulas, which are true or false for system traces. The sets Fml of state formulas and Fml_T of trace formulas with variables in Σ are simultaneously inductively defined in Def. 1.

Definition 1 (dTL formula)**.** The *(state) formulas of differential temporal dynamic logic* (dTL) are defined by the grammar (where ϕ, ψ are dTL state formulas, π is a dTL trace formula, θ_1, θ_2 (polynomial) terms, x a variable, α a HP):

$$\phi, \psi \ ::= \ \theta_1 = \theta_2 \mid \theta_1 \geq \theta_2 \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \forall x\, \phi \mid \exists x\, \phi \mid [\alpha]\pi \mid \langle\alpha\rangle\pi$$

The *trace formulas of* dTL are defined by the grammar (where ϕ is a dTL state formula):

$$\pi ::= \phi \mid \Box\phi \mid \Diamond\phi$$

Operators $>, \leq, <, \leftrightarrow$ *can be defined as usual, e.g.,* $\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$.

Formulas without $\Box$ and $\Diamond$ are *nontemporal formulas* and have the same semantics as the corresponding d$\mathcal{L}$ formulas. Unlike in CTL, dTL state formulas are true on a trace if they hold for the *last* state of a trace, not for the first. Thus, dTL formula $[\alpha]\phi$ expresses that ϕ is true at the end of each trace of α, which is the same as the d$\mathcal{L}$ formula $[\alpha]\phi$. In contrast, $[\alpha]\Box\phi$ expresses that ϕ is true *all along* all states of every trace of α. This combination gives a smooth embedding of nontemporal d$\mathcal{L}$ into dTL and makes it possible to define a compositional calculus. Like CTL, dTL allows nesting with a branching time semantics [EC82], e.g., $[\alpha]\Box(x \geq 2 \rightarrow \langle\beta\rangle\Diamond x \leq 0)$.

4 Trace Semantics of Hybrid Programs

In differential dynamic d$\mathcal{L}$ [Pla08, Pla12c] from Lecture 4, modalities only refer to the final states of system runs and the semantics is a reachability relation on states: State ω

is reachable from state ν using system α if there is a run of α which terminates in ω when started in ν. For dTL, however, formulas can refer to intermediate states of runs as well. To capture this, we change the semantics of a hybrid system α to be the set of its possible *traces*, i.e., successions of states that occur during the evolution of α. The relation between the initial and the final state alone is not sufficient.

States define the values of system variables during a hybrid evolution. A *state* is a map $\nu : \Sigma \to \mathbb{R}$. In addition, we distinguish a separate state Λ to denote the failure of a system run when it is *aborted* due to a test $?\chi$ that yields *false*. In particular, Λ can only occur at the end of an aborted system run and marks that no further extension of that trace is possible because of a failed test. The set of all states is denoted by $\mathcal{S}$.

Hybrid systems evolve along piecewise continuous traces in multi-dimensional space as time passes. Continuous phases are governed by differential equations, whereas discontinuities are caused by discrete jumps in state space. Unlike in discrete cases [Pra79, BS01], traces are not just sequences of states, since hybrid systems pass through uncountably many states even in bounded time. Beyond that, continuous changes are more involved than in pure real time [ACD90, HNSY92], because all variables can evolve along differential equations with different slopes. Generalizing the real-time traces of [HNSY92], the following definition captures hybrid behaviour by splitting the uncountable succession of states into periods σ_i that are regulated by the same control law. For discrete jumps, some of those periods are point flows of duration 0.

The (trace) semantics of hybrid programs is compositional, that is, the semantics of a complex program is defined as a simple function of the trace semantics of its parts. What a hybrid trace captures is the full temporal evolution of a hybrid system. Hybrid systems can behave in different ways, so their trace semantics will be a set of hybrid traces, each of which describes one particular temporal evolution over time. Time, however, is hybridized to a pair (i, ζ) of a discrete time index $i \in \mathbb{N}$ and a real time point $\zeta \in \mathbb{R}$. A single time component $\zeta \in \mathbb{R}$ itself would an inadequate model of time for hybrid systems, because hybrid systems can make progress by a discrete transition without continuous time passing. That happens whenever discrete controls take action. Continuous time only passes during continuous evolutions along differential equations. Discrete actions only make discrete time index i pass.

Definition 2 (Hybrid trace)**.** A *trace* is a (nonempty) finite or infinite sequence $\sigma = (\sigma_0, \sigma_1, \sigma_2, \dots)$ of functions $\sigma_i : [0, r_i] \to \mathcal{S}$ with their respective durations $r_i \in \mathbb{R}$ (for $i \in \mathbb{N}$). A *position* of σ is a pair (i, ζ) with $i \in \mathbb{N}$ and ζ in the interval $[0, r_i]$; the state of σ at (i, ζ) is $\sigma_i(\zeta)$. Positions of σ are ordered lexicographically by $(i, \zeta) \prec (j, \xi)$ iff either $i < j$, or $i = j$ and $\zeta < \xi$. Further, for a state $\nu \in \mathcal{S}$, $\hat{\nu} : 0 \mapsto \nu$ is the *point flow* at ν with duration 0. A trace *terminates* if it is a finite sequence $(\sigma_0, \sigma_1, \dots, \sigma_n)$ and $\sigma_n(r_n) \neq \Lambda$. In that case, the last state $\sigma_n(r_n)$ is denoted by $\operatorname{last} \sigma$. The first state $\sigma_0(0)$ is denoted by $\operatorname{first} \sigma$.

Unlike in [ACD90, HNSY92], the definition of traces also admits finite traces of bounded duration, which is necessary for compositionality of traces in $\alpha; \beta$. The semantics of hybrid programs α as the set $\tau(\alpha)$ of its possible traces depends on valuations $[\![\cdot]\!]_\nu$ of formulas and terms at intermediate states ν. The valuation of terms and interpretations of function and predicate symbols are as for real arithmetic (Lecture 4). The valuation of formulas will be defined in Def. 6. Again, we use ν_x^d to denote the *modification* that agrees with state ν on all variables except for the symbol x, which is changed to $d \in \mathbb{R}$.

Definition 3 (Trace semantics of hybrid programs)**.** The *trace semantics, $\tau(\alpha)$, of a hybrid program α*, is the set of all its possible hybrid traces and is defined inductively as follows:

1. $\tau(x := \theta) = \{(\hat{\nu}, \hat{\omega}) \ : \ \omega = \nu$ except that $[\![x]\!]_\omega = [\![\theta]\!]_\nu$ for $\nu \in \mathcal{S}\}$
2. $\tau(x' = \theta \,\&\, H) = \{(\varphi) \ : \ \varphi(t) \models x' = \theta$ and $\varphi(t) \models H$ for all $0 \leq t \leq r$ for a solution $\varphi : [0, r] \to \mathcal{S}$ of any duration $r\}$; i.e., with $\varphi(t)(x') \overset{\text{def}}{=} \frac{\mathsf{d}\varphi(\zeta)(x)}{\mathsf{d}\zeta}(t)$, φ solves the differential equation and satisfies H at all times, see Lecture 2.
3. $\tau(?\chi) = \{(\hat{\nu}) \ : \ [\![\chi]\!]_\nu = \mathit{true}\} \cup \{(\hat{\nu}, \hat{\Lambda}) \ : \ [\![\chi]\!]_\nu = \mathit{false}\}$
4. $\tau(\alpha \cup \beta) = \tau(\alpha) \cup \tau(\beta)$
5. $\tau(\alpha; \beta) = \{\sigma \circ \varsigma \ : \ \sigma \in \tau(\alpha)\,,\, \varsigma \in \tau(\beta)$ when $\sigma \circ \varsigma$ is defined$\}$;
   the composition of $\sigma = (\sigma_0, \sigma_1, \sigma_2, \dots)$ and $\varsigma = (\varsigma_0, \varsigma_1, \varsigma_2, \dots)$ is
   $$\sigma \circ \varsigma := \begin{cases} (\sigma_0, \dots, \sigma_n, \varsigma_0, \varsigma_1, \dots) & \text{if } \sigma \text{ terminates at } \sigma_n \text{ and } \mathrm{last}\,\sigma = \mathrm{first}\,\varsigma \\ \sigma & \text{if } \sigma \text{ does not terminate} \\ \text{not defined} & \text{otherwise} \end{cases}$$
6. $\tau(\alpha^*) = \bigcup_{n \in \mathbb{N}} \tau(\alpha^n)$, where $\alpha^{n+1} := (\alpha^n; \alpha)$ for $n \geq 1$, as well as $\alpha^1 := \alpha$ and $\alpha^0 := (?\mathit{true})$.

Time passes differently during discrete and continuous change. During continuous evolution, the discrete step index i of positions (i, ζ) remains constant, whereas the continuous duration ζ remains 0 during discrete point flows. This permits multiple discrete state changes to happen at the same (super-dense) continuous time, unlike in other approaches [ACD90].

Example 4. For comparing the transition semantics of hybrid programs for $\mathsf{d}\mathcal{L}$ from Lecture 3 and the trace semantics of hybrid programs for dTL from Def. 3, consider the following simple hybrid program α:

$$a := -2a;\ a := a^2.$$

The transition semantics is just the relation between initial and final states:

$$\rho(\alpha) \equiv \{(\nu, \omega) \ : \ \omega \text{ is like } \nu \text{ except that } \omega(a) = 4\nu(a)^2\}.$$

In particular, the d$\mathcal{L}$ formula $[\alpha]a \geq 0$ is valid, because all final states have a square as the value of a. In contrast, the trace semantics of α retains all intermediate states:

$$\begin{aligned}\tau(\alpha) \equiv \{(\hat{\nu}, \hat{s}, \hat{\omega}) \ : \ & s \text{ is like } \nu \text{ except } s(a) = -2\nu(a) \\ & \text{and } \omega \text{ is like } s \text{ except } \omega(a) = s(a)^2 = 4\nu(a)^2\}.\end{aligned}$$

During these traces, $a \geq 0$ does not hold at all states. If the trace starts with a positive value ($\nu \models a > 0$), then it will become negative at the point flow s (where $s \models a < 0$), yet recover to a positive value ($\omega \models a > 0$) at the end.

Example 5. The previous example only had discrete jumps, and, thus, the traces only involved point flows. Now consider the hybrid program β from the train context:

$$a := -b;\ z' = v, v' = a;\ ?v \geq 0;\ a := A;\ z' = v, v' = a.$$

The transition semantics of this program only considers successful runs to completion. In particular, if $A > 0$, the velocity v will always be nonnegative at the end (otherwise the test $?v \geq 0$ in the middle fails and the program aborts), because the last differential equation will accelerate and increase the velocity again. Thus, the position z at the end of the program run will never be smaller than at the beginning.

If, instead, we consider the trace semantics of β, all intermediate states are in the set of traces:

$$\begin{aligned}\tau(\beta) \ \equiv \ & \{(\hat{\mu_0}, \hat{\mu_1}, \varphi_1, \hat{\mu_2}, \hat{\mu_3}, \varphi_2) \ : \ \mu_1 = \mu_0[a \mapsto -\mu_0(b)] \text{ and} \\ & \quad \varphi_1 \text{ is a state flow of some duration } r_1 \geq 0 \text{ with } \varphi_1 \models z' = v \wedge v' = a \\ & \quad \text{starting in } \varphi_1(0) = \mu_1 \text{ and ending in a state with } \varphi_1(r_1)(v) \geq 0 \\ & \quad \text{and } \mu_2 = \varphi_1(r_1), \mu_3 = \varphi_1(r_1)[a \mapsto \varphi_1(r_1)(A)] \text{ and} \\ & \quad \varphi_2 \text{ is a state flow of some duration } r_2 \geq 0 \text{ with } \varphi_2 \models z' = v \wedge v' = a \\ & \quad \text{starting in } \varphi_2(0) = \mu_3 \text{ and ending in state } \varphi_2(r_2)\} \\ & \cup \{(\hat{\mu_0}, \hat{\mu_1}, \varphi_1, \hat{\mu_2}, \hat{\Lambda}) \ : \ \mu_1 = \mu_0[a \mapsto -\mu_0(b)] \text{ and} \\ & \quad \varphi_1 \text{ is a state flow of some duration } r \geq 0 \text{ with } \varphi_1 \models z' = v \wedge v' = a \\ & \quad \text{starting in } \varphi_1(0) = \mu_1 \text{ and ending in a state with } \varphi_1(r)(v) < 0 \\ & \quad \text{further } \mu_2 = \varphi_1(r)\}.\end{aligned}$$

The first set is the set of traces where the test $?v \geq 0$ in the middle succeeds and the system continues. The second set (after the union) is the set of traces that are aborted with $\hat{\Lambda}$ during their execution, because the middle test fails. Note that the traces in the first set have two continuous flows φ_1, φ_2 and four point flows $\hat{\mu_0}, \hat{\mu_1}, \hat{\mu_2}, \hat{\mu_3}$ in each trace. The traces in the second set have only one continuous flow φ_1 and three point flows $\hat{\mu_0}, \hat{\mu_1}, \hat{\mu_2}$, because the subsequent aborting point flow $\hat{\Lambda}$ does not terminate and aborts all further execution. In the trace semantics, $v < 0$ is possible in the middle of some traces, which is a fact that the transition semantics does not notice. Combining traces for $\alpha \cup \beta$, that is, for

$$(a := -2a;\ a := a^2) \cup (a := -b;\ z' = v, v' = a;\ ?v \geq 0;\ a := A;\ z' = v, v' = a)$$

is just the union $\tau(\alpha) \cup \tau(\beta)$ of the traces $\tau(\alpha)$ and $\tau(\beta)$ from Examples 4 and 5. Note that $a \leq 0$ will hold at least once during every trace of $\alpha \cup \beta$, either in the beginning, or after setting $a := -2a$ or $a := -b$, respectively, when we assume $b > 0$.

5 Semantics of State and Trace Formulas

In the semantics of dTL formulas, the dynamic modalities determine the set of traces according to the trace semantics of hybrid programs, and, independently, the temporal modalities determine at which points in time the respective postcondition needs to hold. The semantics of formulas is compositional and denotational, that is, the semantics of a complex formula is defined as a simple function of the semantics of its subformulas.

Definition 6 (dTL semantics)**.** The *satisfaction relation*$\nu \models \phi$ for a dTL (state) formula ϕ in state ν is defined inductively:

- $\nu \models (\theta_1 = \theta_2)$ iff $[\![\theta_1]\!]_\nu = [\![\theta_2]\!]_\nu$.
- $\nu \models (\theta_1 \geq \theta_2)$ iff $[\![\theta_1]\!]_\nu \geq [\![\theta_2]\!]_\nu$.
- $\nu \models \neg\phi$ iff $\nu \not\models \phi$, i.e. if it is not the case that $\nu \models \phi$.
- $\nu \models \phi \wedge \psi$ iff $\nu \models \phi$ and $\nu \models \psi$.
- $\nu \models \phi \vee \psi$ iff $\nu \models \phi$ or $\nu \models \psi$.
- $\nu \models \phi \rightarrow \psi$ iff $\nu \not\models \phi$ or $\nu \models \psi$.
- $\nu \models \phi \leftrightarrow \psi$ iff ($\nu \models \phi$ and $\nu \models \psi$) or ($\nu \not\models \phi$ and $\nu \not\models \psi$).
- $\nu \models \forall x\, \phi$ iff $\nu_x^d \models \phi$ for all $d \in \mathbb{R}$.
- $\nu \models \exists x\, \phi$ iff $\nu_x^d \models \phi$ for some $d \in \mathbb{R}$.
- $\nu \models [\alpha]\pi$ iff for each trace $\sigma \in \tau(\alpha)$that starts in $\mathrm{first}\, \sigma = \nu$, if $[\![\pi]\!]_\sigma$ is defined, then $[\![\pi]\!]_\sigma = \mathit{true}$.
- $\nu \models \langle\alpha\rangle\pi$ iff there is a trace $\sigma \in \tau(\alpha)$ starting in $\mathrm{first}\, \sigma = \nu$ such that $[\![\pi]\!]_\sigma$ is defined and $[\![\pi]\!]_\sigma = \mathit{true}$.

For trace formulas, the *valuation* $[\![\cdot]\!]_\sigma$ with respect to trace σ is defined inductively as:

1. If ϕ is a state formula, then $[\![\phi]\!]_\sigma = [\![\phi]\!]_{\mathrm{last}\, \sigma}$ if σ terminates, whereas $[\![\phi]\!]_\sigma$ is *not defined* if σ does not terminate.
2. $[\![\Box\phi]\!]_\sigma = \mathit{true}$ iff $\sigma_i(\zeta) \models \phi$ holds for all positions (i, ζ) of σ with $\sigma_i(\zeta) \neq \Lambda$.
3. $[\![\Diamond\phi]\!]_\sigma = \mathit{true}$ iff $\sigma_i(\zeta) \models \phi$ holds for some position (i, ζ) of σ with $\sigma_i(\zeta) \neq \Lambda$.

As usual, a (state) formula is *valid* if it is true in all states. If $\nu \models \phi$, then we say that dTL state formula ϕ is true at ν or that ν is a model of ϕ. A (state) formula ϕ is *valid*, written $\vDash \phi$, iff $\nu \models \phi$ for all states ν. A formula ϕ is a *consequence* of a set of formulas Γ, written $\Gamma \vDash \phi$, iff, for each ν: ($\nu \models \psi$ for all $\psi \in \Gamma$) implies that $\nu \models \phi$. Likewise, for trace formula π and trace σ we write $\sigma \models \pi$ iff $[\![\pi]\!]_\sigma = \mathit{true}$ and $\sigma \not\models \pi$ iff $[\![\pi]\!]_\sigma = \mathit{false}$. In particular, we only write $\sigma \models \pi$ or $\sigma \not\models \pi$ if $[\![\pi]\!]_\sigma$ is defined, which it is not the case if π is a state formula and σ does not terminate. The points where a dTL property ϕ has to hold for the various combinations of temporal and dynamic modalities are illustrated in Fig. 1.

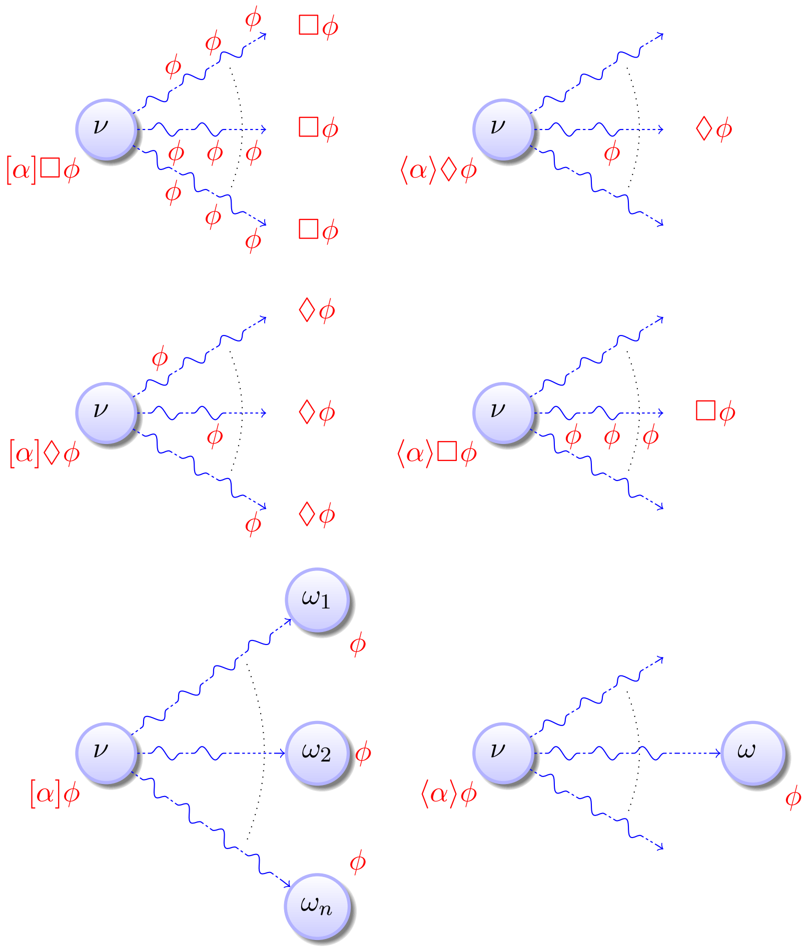


Figure 1: Trace semantics of dTL formulas

6 Conservative Temporal Extension

The following result shows that the extension by temporal operators that dTL provides does not change the meaning of nontemporal $\mathsf{d}\mathcal{L}$ formulas. The trace semantics given in Def. 6 is equivalent to the final state reachability relation semantics given in Lecture 4 for the sublogic $\mathsf{d}\mathcal{L}$ of dTL.

Proposition 7 (Conservative temporal extension [Pla10, Proposition 4.1]). *The logic* dTL *is a* conservative extension *of nontemporal* $\mathsf{d}\mathcal{L}$*, i.e., the set of valid* $\mathsf{d}\mathcal{L}$ *formulas is the same with respect to transition reachability semantics of* $\mathsf{d}\mathcal{L}$ *(Lecture 4) as with respect to the trace semantics of* dTL *(Def. 6).*

The proof is by induction using that the reachability relation fits to the trace semantics. That is, the reachability relation semantics of hybrid programs agrees with the first and last states of the traces in the trace semantics.

Lemma 8 (Trace relation [Pla10, Lemma 4.1]). *For hybrid programs α:*

$$\rho(\alpha) \;=\; \{(\text{first}\,\sigma, \text{last}\,\sigma) \;:\; \sigma \in \tau(\alpha) \textit{ terminates}\}.$$

In particular, the trace semantics from today's lecture fits seamlessly to the original reachability semantics that was the basis for the previous lectures. The trace semantics exactly satisfies the objective of characterizing the same reachability relation between initial and final states, while, in addition, keeping a trace of all intermediate states around. For nontemporal dTL formulas and for $\mathsf{d}\mathcal{L}$ formulas, this full trace with intermediate states is not needed, because the reachability relation between initial and final states is sufficient to define the meaning For temporal dTL formulas, instead, the trace is crucial to give a meaning to $\Box$ and $\Diamond$.

7 Summary

This lecture introduced a temporal extension of the logic $\mathsf{d}\mathcal{L}$ and a trace semantics of hybrid programs. This extends the syntax and semantics to the presence of temporal modalities. The next lecture investigates how to prove temporal properties of hybrid systems.

Exercises

Exercise 1. Can you give a formula of the following form that is valid?

$$[\alpha]\Box\phi \land \neg[\alpha]\phi$$

Exercise 2. In which case does the temporal $[\alpha]\Box\phi$ differ from the nontemporal $[\alpha]\phi$.

References

[ACD90] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking for real-time systems. In *LICS*, pages 414–425. IEEE Computer Society, 1990.

[BS01] Bernhard Beckert and Steffen Schlager. A sequent calculus for first-order dynamic logic with trace modalities. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *IJCAR*, volume 2083 of *LNCS*, pages 626–641. Springer, 2001.

[DBL12] *Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25–28, 2012*. IEEE, 2012.

[EC82] E. Allen Emerson and Edmund M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Program.*, 2(3):241–266, 1982.

[EH86] E. Allen Emerson and Joseph Y. Halpern. "Sometimes" and "Not Never" revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.

[HKT00] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic logic*. MIT Press, 2000.

[HNSY92] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. In *LICS*, pages 394–406. IEEE Computer Society, 1992.

[Pla07] André Platzer. A temporal dynamic logic for verifying hybrid system invariants. In Sergei N. Artëmov and Anil Nerode, editors, *LFCS*, volume 4514 of *LNCS*, pages 457–471. Springer, 2007. doi:10.1007/978-3-540-72734-7_32.

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

[Pla10] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14509-4.

[Pla12a] André Platzer. The complete proof theory of hybrid systems. In *LICS* [DBL12], pages 541–550. doi:10.1109/LICS.2012.64.

[Pla12b] André Platzer. Dynamic logics of dynamical systems. *CoRR*, abs/1205.4788, 2012. arXiv:1205.4788.

[Pla12c] André Platzer. Logics of dynamical systems. In *LICS* [DBL12], pages 13–24. doi:10.1109/LICS.2012.13.

[Pra79] Vaughan R. Pratt. Process logic. In *POPL*, pages 93–100, 1979.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Differential & Temporal Proofs

André Platzer

Carnegie Mellon University
Lecture 17

1 Introduction

This lecture continues the study of temporal aspects of cyber-physical systems that Lecture 5 on Differential & Temporal Logics started. The trace semantics of hybrid programs as well as the semantics of differential temporal dynamic logic (dTL) [Pla10], a temporal extension of differential dynamic logic d$\mathcal{L}$ [Pla08, Pla12], have been discussed in said lecture.

This lecture is based on [Pla10, Chapter 4], which extends [Pla07].

2 Temporal Proof Rules

When extending a logic, it is not enough to extend just the syntax (Lecture 5) and semantics (Lecture 5). The proof rules also need to be extended to handle the new concepts, that is the temporal modalities of dTL.

This section shows a sequent calculus for verifying temporal specifications of hybrid systems in differential temporal dynamic logic dTL. With the basic idea being to perform a symbolic decomposition, the calculus transforms hybrid programs successively into simpler logical formulas describing their effects. Statements about the temporal behaviour of a hybrid program are successively reduced to corresponding nontemporal statements about the intermediate states. This lecture shows a proof calculus for differential temporal dynamic logic dTL that inherits the proof rules of d$\mathcal{L}$ from previous lectures and adds new proof rules for temporal modalities.

Inherited Nontemporal Rules The dTL calculus is presented in Fig. 1 and inherits the (nontemporal) d$\mathcal{L}$ proof rules, i.e., the propositional, first-order, dynamic, and global rules from d$\mathcal{L}$. That is, it includes the propositional and quantifier rules from Lecture 6. The dynamic rules ($\langle;\rangle$–$[']$) and global rules ($[]gen$,$\langle\rangle gen$,ind,con) for handling nontemporal dynamic modalities are also inherited directly from Lecture 6. The only possible exception is that $[\cup]$,$\langle\cup\rangle$ can be generalised to apply to formulas of the form $[\alpha\cup\beta]\pi$ where π is an arbitrary trace formula, and not just a state formula as in d$\mathcal{L}$. Thus, π may begin with $\Box$ or $\Diamond$, which is why the rules are repeated in this generalised form as $[\cup]\Box$ and $\langle\cup\rangle\Diamond$ in Fig. 1.

Note 1.

$$([\cup]\Box)\ \frac{[\alpha]\pi\land[\beta]\pi}{[\alpha\cup\beta]\pi}\text{[1]} \qquad (\langle\cup\rangle\Diamond)\ \frac{\langle\alpha\rangle\pi\lor\langle\beta\rangle\pi}{\langle\alpha\cup\beta\rangle\pi}\text{[1]}$$

$$([;]\Box)\ \frac{[\alpha]\Box\phi\land[\alpha][\beta]\Box\phi}{[\alpha;\beta]\Box\phi} \qquad (\langle;\rangle\Diamond)\ \frac{\langle\alpha\rangle\Diamond\phi\lor\langle\alpha\rangle\langle\beta\rangle\Diamond\phi}{\langle\alpha;\beta\rangle\Diamond\phi}$$

$$([?]\Box)\ \frac{\phi}{[?\chi]\Box\phi} \qquad (\langle?\rangle\Diamond)\ \frac{\phi}{\langle?\chi\rangle\Diamond\phi}$$

$$([:=]\Box)\ \frac{\phi\land[x:=\theta]\phi}{[x:=\theta]\Box\phi} \qquad (\langle:=\rangle\Diamond)\ \frac{\phi\lor\langle x:=\theta\rangle\phi}{\langle x:=\theta\rangle\Diamond\phi}$$

$$([']\Box)\ \frac{[x'=\theta]\phi}{[x'=\theta]\Box\phi} \qquad (\langle'\rangle\Diamond)\ \frac{\langle x'=\theta\rangle\phi}{\langle x'=\theta\rangle\Diamond\phi}$$

$$([^{*n}]\Box)\ \frac{[\alpha;\alpha^*]\Box\phi}{[\alpha^*]\Box\phi} \qquad (\langle^{*n}\rangle\Diamond)\ \frac{\langle\alpha;\alpha^*\rangle\Diamond\phi}{\langle\alpha^*\rangle\Diamond\phi}$$

$$([^*]\Box)\ \frac{[\alpha^*][\alpha]\Box\phi}{[\alpha^*]\Box\phi} \qquad (\langle^*\rangle\Diamond)\ \frac{\langle\alpha^*\rangle\langle\alpha\rangle\Diamond\phi}{\langle\alpha^*\rangle\Diamond\phi}$$

[1] π is a trace formula and—unlike the state formulas ϕ and ψ—may thus begin with a temporal modality $\Box$ or $\Diamond$.

Figure 1: Axiomatization of differential temporal dynamic logic dTL

Temporal Rules The new temporal rules in Fig. 1 for the dTL calculus successively transform temporal specifications of hybrid programs into nontemporal d$\mathcal{L}$ formulas. The idea underlying this transformation is to decompose hybrid programs and recursively augment intermediate state transitions with appropriate specifications. Also see Fig. 2 for an illustration of the correspondence of a representative set of proof rules for temporal modalities to the trace semantics of hybrid programs (Def. **??**).

Rule $[;]\Box$ decomposes invariants of $\alpha;\beta$ (i.e., $[\alpha;\beta]\Box\phi$ holds) into an invariant of α (i.e., $[\alpha]\Box\phi$) and an invariant of β that holds when β is started in *any* final state of α (i.e., $[\alpha]([\beta]\Box\phi)$). Its difference with the d$\mathcal{L}$ rule $[;]$ thus is that the dTL rule $[;]\Box$ also checks safety invariant ϕ at the symbolic states in between the execution of α and β, and recursively so because of the temporal modality $\Box$. Again, see Fig. 2 for an illustration

$$\frac{\phi \land [x := \theta]\phi}{[x := \theta]\Box\phi}$$

$$\frac{[x' = \theta]\phi}{[x' = \theta]\Box\phi}$$

$$\frac{[\alpha]\Box\phi \land [\alpha][\beta]\Box\phi}{[\alpha;\beta]\Box\phi}$$

$$\frac{[\alpha]\Box\phi \land [\beta]\Box\phi}{[\alpha \cup \beta]\Box\phi}$$

$$\frac{[\alpha^*][\alpha]\Box\phi}{[\alpha^*]\Box\phi}$$

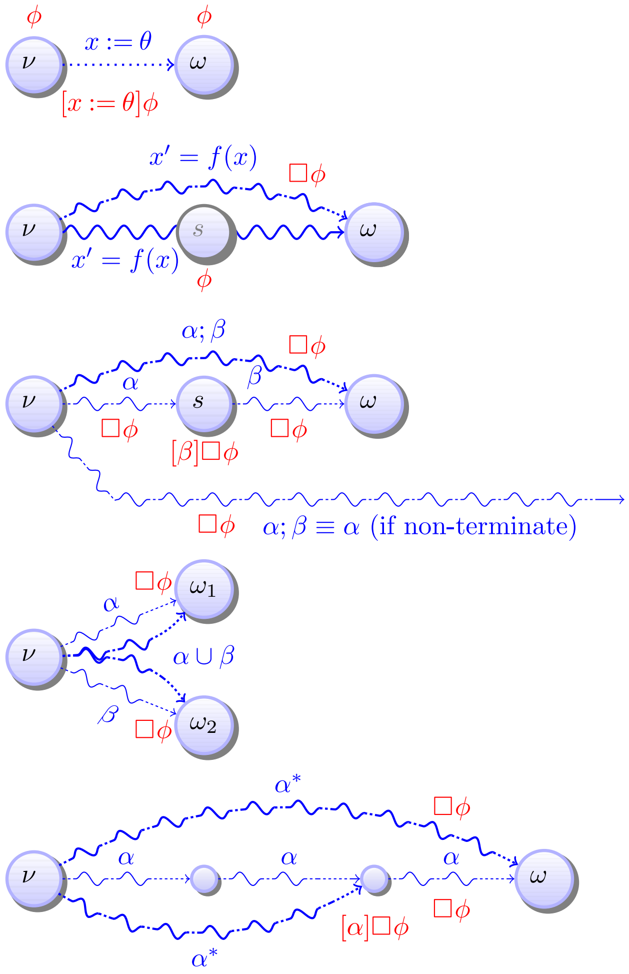


Figure 2: Correspondence of temporal proof rules and trace semantics

of this proof principle.

Rule $[:=]\Box$ expresses that invariants of assignments need to hold before and after the discrete change (similarly for $[?]\Box$, except that tests do not lead to a state change, so ϕ holding before the test is all there is to it). Rule $[']\Box$ can directly reduce invariants of continuous evolutions to nontemporal formulas as restrictions of solutions of differential equations are themselves solutions of different duration and thus already included in the evolutions of $x' = \theta$. In particular, observe that the handling of differential equations within hybrid systems is fully encapsulated within the fragment of dynamic rules from d$\mathcal{L}$.

The (optional) iteration rule $[^{*n}]\Box$ can partially unwind loops. It relies on rule $[;]\Box$ and is simpler than d$\mathcal{L}$ rule $[^{*n}]$, because the other rules will inductively produce a premise that ϕ holds in the current state, because of the temporal modality $\Box\phi$. The dual rules $\langle\cup\rangle\Diamond$,$\langle;\rangle\Diamond$,$\langle?\rangle\Diamond$,$\langle:=\rangle\Diamond$,$\langle'\rangle\Diamond$,$\langle^{*n}\rangle\Diamond$ work similarly.

In d$\mathcal{L}$ (Lecture 7 on Control Loops & Invariants), the primary means for handling loops are the invariant induction (*ind*) and variant convergence (*con*) rules. The logic dTL takes a different, completely modular approach for verifying temporal properties of loops based on the d$\mathcal{L}$ capabilities for verifying nontemporal properties of loops. Rules $[^*]\Box$ and $\langle^*\rangle\Diamond$ actually *define* temporal properties of loops inductively. Rule $[^*]\Box$ expresses that ϕ holds at all times during repetitions of α (i.e., $[\alpha^*]\Box\phi$) iff, *after* repeating α any number of times, ϕ holds at all times *during* one execution of α (i.e., $[\alpha^*]([\alpha]\Box\phi)$). See Fig. 2 for an illustration. Dually, $\langle^*\rangle\Diamond$ expresses that α holds at some time during repetitions of α (i.e., $\langle\alpha^*\rangle\Diamond\phi$) iff, after some number of repetitions of α, formula ϕ holds at some point during one execution of α (i.e., $\langle\alpha^*\rangle(\langle\alpha\rangle\Diamond\phi)$). In this context, the nontemporal modality $\langle\alpha^*\rangle$ can be thought of as skipping over to the iteration of α during which ϕ actually occurs, as expressed by the nested dTL formula $\langle\alpha\rangle\Diamond\phi$. The inductive definition rules $[^*]\Box$ and $\langle^*\rangle\Diamond$ completely reduce temporal properties of loops to dTL properties of standard nontemporal d$\mathcal{L}$-modalities such that standard induction (*ind*) or convergence rules (*con*) can be used for the outer nontemporal modality of the loop. Hence, after applying the inductive loop definition rules $[^*]\Box$ and $\langle^*\rangle\Diamond$, the standard d$\mathcal{L}$ loop invariant and variant rules can be used for verifying temporal properties of loops without change, except that the postcondition contains temporal modalities.

Rules for handling $[\alpha]\Diamond\phi$ and $\langle\alpha\rangle\Box\phi$ are discussed in [Pla10].

3 Temporal Bouncing Ball

Recall the bouncing ball that has served us so well in previous lectures.

$$(v^2 \leq 2g(H-h) \wedge h \geq 0 \wedge g > 0 \wedge H \geq 0 \wedge 1 > c \geq 0) \rightarrow [ball](0 \leq h \leq H). \tag{1}$$

Use the abbreviations

$$\begin{aligned} \mathit{ball} &\equiv h' = v, v' = -g \,\&\, h \geq 0; (?h > 0 \cup (?h = 0; v := -cv)) \\ \psi &\equiv g > 0 \land H \geq 0 \land 1 > c \geq 0, \\ \varphi &\equiv v^2 \leq 2g(H - h) \land h \geq 0 \\ \langle h := ..(t)\rangle F &\equiv \langle h := h + vt - \frac{g}{2}t^2; v := v - tg\rangle F. \end{aligned}$$

When simplifying the *ball* dynamics to remove evolution domain constraints:

$$h' = v, v' = -g; (?h > 0 \cup (?h = 0; v := -cv))$$

the proof for the simplified bouncing ball property without evolution domain constraint is shown in Fig. 3. The dℒ proof for the original bouncing ball property (1) with an evolution domain constraint is shown in Fig. 4.

4 Verification Example

Recall the bouncing ball. The proofs from previous lectures or Fig. 4 can be generalized easily to a proof of the temporal property

$$\begin{aligned} &v^2 \leq 2g(H - h) \land h \geq 0 \land g > 0 \land H \geq 0 \land 1 > c \geq 0 \\ &\quad \to [(h'' = -g \,\&\, h \geq 0; (?h > 0 \cup (?h = 0; v := -cv)))^*]\Box(0 \leq h \leq H). \end{aligned} \tag{2}$$

The only aspect of the proof that changes is that the temporal proof rules in Fig. 1 are used instead of the dynamic proof rules for dℒ, and that the resulting extra proof goals for the invariance property at intermediate steps have to be proven.

In contrast, the proof in Fig. 3 for the simplified dynamics without evolution domain restriction $h \geq 0$ cannot be generalized to a proof of the temporal property

$$\begin{aligned} &v^2 \leq 2g(H - h) \land h \geq 0 \land g > 0 \land H \geq 0 \land 1 > c \geq 0 \\ &\quad \to [(h'' = -g; (?h > 0 \cup (?h = 0; v := -cv)))^*]\Box(0 \leq h \leq H). \end{aligned} \tag{3}$$

This difference in provability is for good reasons. The property in (2) is valid, but the property in (3) is not! While there was no noticeable semantical difference between the nontemporal dℒ counterparts of the properties (2) versus (3), there is a decisive difference between the corresponding temporal properties (3) and (2). Because there is no evolution domain restriction in (3), its hybrid program does not prevent continuous evolution to a negative height under the floor ($h < 0$), for which $0 \leq h \leq H$ does not hold.

The reason for this discrepancy of the temporal version compared to the nontemporal versions thus is that the nontemporal modalities do not "see" the temporary violation of $0 \leq h \leq H$. Such a temporary violation of $0 \leq h$ during the continuous evolution does not produce a successful run of the hybrid program, because it is blocked by

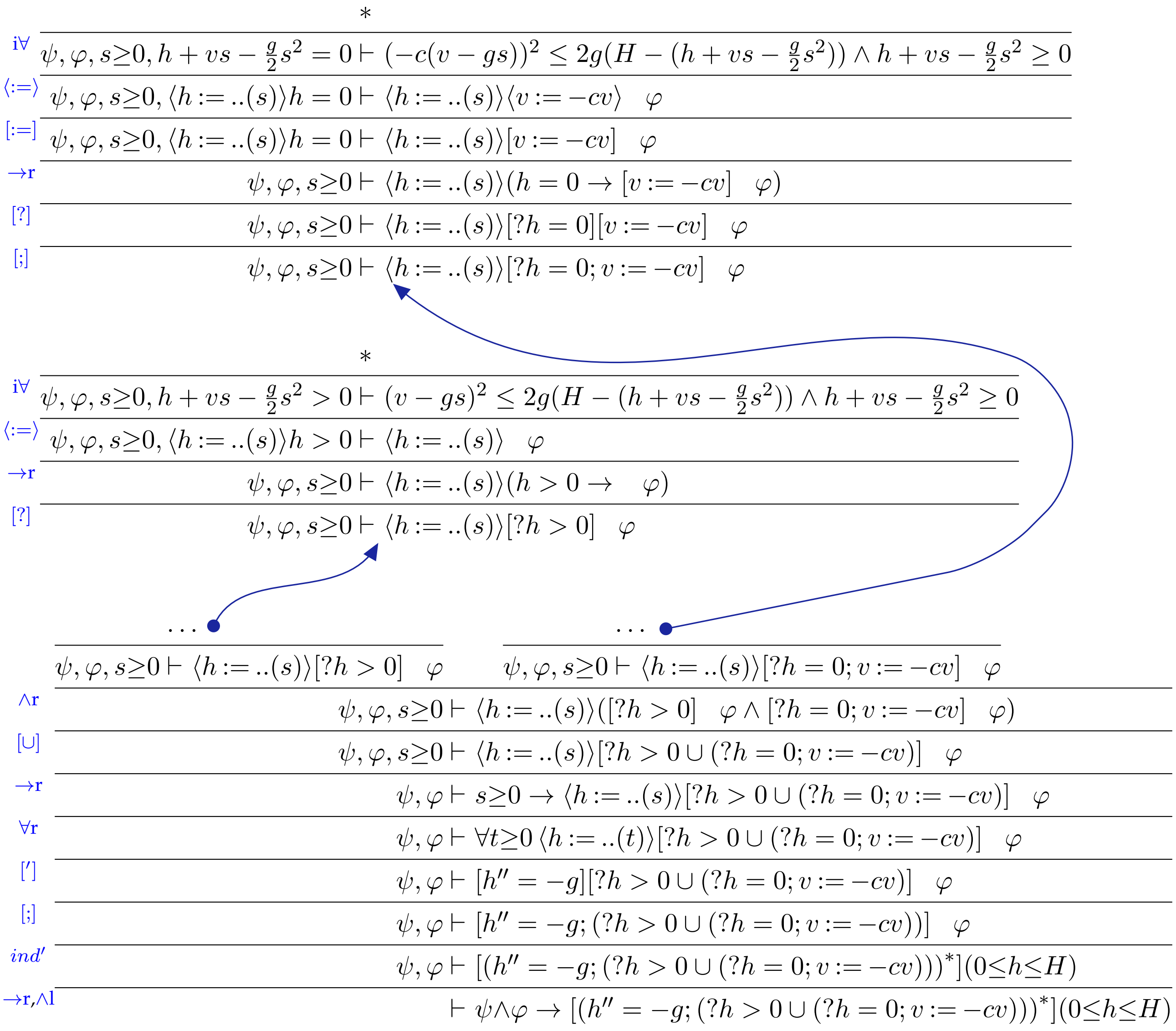

$*$
i∀ $\psi,\varphi,s{\geq}0,h+vs-\frac{g}{2}s^2=0\vdash(-c(v-gs))^2\leq 2g(H-(h+vs-\frac{g}{2}s^2))\wedge h+vs-\frac{g}{2}s^2\geq 0$
⟨:=⟩ $\psi,\varphi,s{\geq}0,\langle h:=..(s)\rangle h=0\vdash\langle h:=..(s)\rangle\langle v:=-cv\rangle\quad\varphi$
[:=] $\psi,\varphi,s{\geq}0,\langle h:=..(s)\rangle h=0\vdash\langle h:=..(s)\rangle[v:=-cv]\quad\varphi$
→r $\psi,\varphi,s{\geq}0\vdash\langle h:=..(s)\rangle(h=0\to[v:=-cv]\quad\varphi)$
[?] $\psi,\varphi,s{\geq}0\vdash\langle h:=..(s)\rangle[?h=0][v:=-cv]\quad\varphi$
[;] $\psi,\varphi,s{\geq}0\vdash\langle h:=..(s)\rangle[?h=0;v:=-cv]\quad\varphi$
$*$
i∀ $\psi,\varphi,s{\geq}0,h+vs-\frac{g}{2}s^2>0\vdash(v-gs)^2\leq 2g(H-(h+vs-\frac{g}{2}s^2))\wedge h+vs-\frac{g}{2}s^2\geq 0$
⟨:=⟩ $\psi,\varphi,s{\geq}0,\langle h:=..(s)\rangle h>0\vdash\langle h:=..(s)\rangle\quad\varphi$
→r $\psi,\varphi,s{\geq}0\vdash\langle h:=..(s)\rangle(h>0\to\quad\varphi)$
[?] $\psi,\varphi,s{\geq}0\vdash\langle h:=..(s)\rangle[?h>0]\quad\varphi$
$\ldots$
$\psi,\varphi,s{\geq}0\vdash\langle h:=..(s)\rangle[?h>0]\quad\varphi$
$\ldots$
$\psi,\varphi,s{\geq}0\vdash\langle h:=..(s)\rangle[?h=0;v:=-cv]\quad\varphi$
∧r $\psi,\varphi,s{\geq}0\vdash\langle h:=..(s)\rangle([?h>0]\quad\varphi\wedge[?h=0;v:=-cv]\quad\varphi)$
[∪] $\psi,\varphi,s{\geq}0\vdash\langle h:=..(s)\rangle[?h>0\cup(?h=0;v:=-cv)]\quad\varphi$
→r $\psi,\varphi\vdash s{\geq}0\to\langle h:=..(s)\rangle[?h>0\cup(?h=0;v:=-cv)]\quad\varphi$
∀r $\psi,\varphi\vdash\forall t{\geq}0\,\langle h:=..(t)\rangle[?h>0\cup(?h=0;v:=-cv)]\quad\varphi$
[′] $\psi,\varphi\vdash[h''=-g][?h>0\cup(?h=0;v:=-cv)]\quad\varphi$
[;] $\psi,\varphi\vdash[h''=-g;(?h>0\cup(?h=0;v:=-cv))]\quad\varphi$
ind′ $\psi,\varphi\vdash[(h''=-g;(?h>0\cup(?h=0;v:=-cv)))^*](0{\leq}h{\leq}H)$
→r,∧l $\vdash\psi\wedge\varphi\to[(h''=-g;(?h>0\cup(?h=0;v:=-cv)))^*](0{\leq}h{\leq}H)$

Figure 3: Bouncing ball proof (no evolution domain)

$$\begin{array}{ll}
 & * \\ \hline
\mathrm{i}\forall & \psi \land \varphi, s{\geq}0, , h + vs - \frac{g}{2}s^2 = 0 \vdash (-c(v - gs))^2 \leq 2g(H - (h + vs - \frac{g}{2}s^2)) \land h + vs - \frac{g}{2}s^2 \geq 0 \\ \hline
\langle:=\rangle & \psi \land \varphi, s{\geq}0, , \langle h := ..(s)\rangle h = 0 \vdash \langle h := ..(s)\rangle\langle v := -cv\rangle \quad \varphi \\ \hline
[:=] & \psi \land \varphi, s{\geq}0, , \langle h := ..(s)\rangle h = 0 \vdash \langle h := ..(s)\rangle[v := -cv] \quad \varphi \\ \hline
\rightarrow\mathrm{r} & \psi \land \varphi, s{\geq}0, \langle h := ..(s)\rangle h \geq 0 \vdash \langle h := ..(s)\rangle(h = 0 \rightarrow [v := -cv] \quad \varphi) \\ \hline
[?] & \psi \land \varphi, s{\geq}0, \langle h := ..(s)\rangle h \geq 0 \vdash \langle h := ..(s)\rangle[?h = 0][v := -cv] \quad \varphi \\ \hline
[;] & \psi \land \varphi, s{\geq}0, \langle h := ..(s)\rangle h \geq 0 \vdash \langle h := ..(s)\rangle[?h = 0; v := -cv] \quad \varphi
\end{array}$$

$$\begin{array}{ll}
 & * \\ \hline
\mathrm{i}\forall & \psi \land \varphi, s{\geq}0, , h + vs - \frac{g}{2}s^2 > 0 \vdash (v - gs)^2 \leq 2g(H - (h + vs - \frac{g}{2}s^2)) \land h + vs - \frac{g}{2}s^2 \geq 0 \\ \hline
\langle:=\rangle & \psi \land \varphi, s{\geq}0, , \langle h := ..(s)\rangle h > 0 \vdash \langle h := ..(s)\rangle \quad \varphi \\ \hline
\rightarrow\mathrm{r} & \psi \land \varphi, s{\geq}0, \langle h := ..(s)\rangle h \geq 0 \vdash \langle h := ..(s)\rangle(h > 0 \rightarrow \quad \varphi) \\ \hline
[?] & \psi \land \varphi, s{\geq}0, \langle h := ..(s)\rangle h \geq 0 \vdash \langle h := ..(s)\rangle[?h > 0] \quad \varphi
\end{array}$$

$$\begin{array}{ll}
 & \dfrac{\dots}{\dots \vdash \langle h := ..(s)\rangle[?h > 0] \quad \varphi} \qquad \dfrac{\dots}{\psi \land \varphi, s{\geq}0, \langle h := ..(s)\rangle h \geq 0 \vdash \langle h := ..(s)\rangle[?h = 0; v := -cv] \quad \varphi} \\ \hline
\land\mathrm{r} & \psi \land \varphi, s{\geq}0, \langle h := ..(s)\rangle h \geq 0 \vdash \langle h := ..(s)\rangle([?h > 0] \quad \varphi \land [?h = 0; v := -cv] \quad \varphi) \\ \hline
[\cup] & \psi \land \varphi, s{\geq}0, \langle h := ..(s)\rangle h \geq 0 \vdash \langle h := ..(s)\rangle[?h > 0 \cup (?h = 0; v := -cv)] \quad \varphi \\ \hline
\rightarrow\mathrm{r} & \psi \land \varphi, s{\geq}0 \vdash \langle h := ..(s)\rangle h \geq 0 \rightarrow \langle h := ..(s)\rangle[?h > 0 \cup (?h = 0; v := -cv)] \quad \varphi \\ \hline
\rightarrow\mathrm{r} & \psi \land \varphi \vdash s{\geq}0 \rightarrow (\langle h := ..(s)\rangle h \geq 0 \rightarrow \langle h := ..(s)\rangle[?h > 0 \cup (?h = 0; v := -cv)] \quad \varphi) \\ \hline
\forall\mathrm{r} & \psi \land \varphi \vdash \forall t{\geq}0\, (\langle h := ..(t)\rangle h \geq 0 \rightarrow \langle h := ..(t)\rangle[?h > 0 \cup (?h = 0; v := -cv)] \quad \varphi) \\ \hline
['] & \psi \land \varphi \vdash [h'' = -g \,\&\, h \geq 0][?h > 0 \cup (?h = 0; v := -cv)] \quad \varphi \\ \hline
[;] & \psi \land \varphi \vdash [h'' = -g \,\&\, h \geq 0; (?h > 0 \cup (?h = 0; v := -cv))] \quad \varphi \\ \hline
\mathit{ind}' & \psi \land \varphi \vdash [(h'' = -g \,\&\, h \geq 0; (?h > 0 \cup (?h = 0; v := -cv)))^*](0{\leq}h{\leq}H) \\ \hline
\rightarrow\mathrm{r} & \vdash \psi \land \varphi \rightarrow [(h'' = -g \,\&\, h \geq 0; (?h > 0 \cup (?h = 0; v := -cv)))^*](0{\leq}h{\leq}H)
\end{array}$$

Figure 4: Bouncing ball proof (with evolution domain)

the subsequent tests $?h = 0$ and $?h > 0$. A state with negative height fails both tests. While this behaviour does not give a successful program transition of $(\nu, \omega) \in \rho(\mathit{ball})$ by Lecture 3 so that the proof in Fig. 3 is correct, the behaviour still gives a valid trace $\sigma \in \tau(\mathit{ball})$ by Def. **??**. This trace σ is a partial trace, because it ends in a failure state Λ, but it is still one of the traces that $[\mathit{ball}]\Box(0 \leq h \leq H)$ quantifies over (quite unlike $[\mathit{ball}](0 \leq h \leq H)$, which only considers final states of successful traces).

5 Summary

This lecture showed a systematic way of specifying and verifying temporal properties of hybrid systems. The focus was on safety properties that hold always throughout the evolution of the system and are specified as $[\alpha]\Box\phi$ with a mix of a temporal and a dynamic modality instead of just a dynamic modality as in $[\alpha]\phi$. The difference is that $[\alpha]\Box\phi$ includes that safety condition ϕ holds at all intermediate states during all traces of α, whereas $[\alpha]\phi$ only specifies that ϕ holds at the end of each trace of α. This difference matters in systems that have more intermediate states than final states. The difference is insignificant for systems that can "stop anytime", because those will already include all intermediate states of longer system runs as the final state of a corresponding shorter system run. This has been the case in almost all systems studied in this course and is frequently the case in practice.

The systematic way of ensuring safety always throughout the execution of hybrid systems is the use of the dynamic and temporal modality $[\alpha]\Box\phi$, which works whether or not the system has the special structure that allows it to stop anytime. In a nutshell, the temporal proof rules for $[\alpha]\Box\phi$ properties lead to additional branches that correspond to the safety conditions at the respective intermediate state. It can be shown that temporal dTL properties reduce to nontemporal d$\mathcal{L}$ properties completely [Pla10, Chapter 4], justifying the intimate relation of temporal and nontemporal properties That completeness result made crucial use of the clever form of the $[^*]\Box$ proof rule.

Other temporal modalities are more complicated but can either be handled directly (in the case of $\langle\alpha\rangle\Diamond\phi$) or by transformation [Pla10].

Exercises

Exercise 1. Can you give a formula of the following form that is valid?

$$[\alpha]\Box\phi \land \neg[\alpha]\phi$$

Exercise 2. In which case does the temporal $[\alpha]\Box\phi$ differ from the nontemporal $[\alpha]\phi$.

Exercise 3. Can you give a temporal box version of the differential invariant proof rule?

References

[Pla07] André Platzer. A temporal dynamic logic for verifying hybrid system invariants. In Sergei N. Artëmov and Anil Nerode, editors, *LFCS*, volume 4514 of *LNCS*, pages 457–471. Springer, 2007. doi:10.1007/978-3-540-72734-7_32.

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

[Pla10] 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. Logics of dynamical systems. In *LICS*, pages 13–24. IEEE, 2012. doi:10.1109/LICS.2012.13.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on
Virtual Substitution & Real Equations

André Platzer

Carnegie Mellon University

Lecture 18

1 Introduction

Cyber-physical systems are important technical concepts for building better systems around us. Their safe design requires careful specification and verification, which this course provides using differential dynamic logic and its proof calculus [Pla08, Pla10, Pla12b]. The proof calculus for differential dynamic logic has a number of powerful axioms and proof rules (especially in Lecture 5, Lecture 6, Lecture 11, and Lecture 15). In theory, the *only* difficult problem in proving hybrid systems safety is finding their invariants or differential invariants [Pla08, Pla12a] (Lecture 14 on Differential Invariants & Proof Theory). In practice, however, the handling of real arithmetic is another challenge that you have faced, even though the problem is easier in theory. How arithmetic interfaces with proofs has already been discussed in Lecture 9 on Proofs & Arithmetic. Today's lecture shows one technique for deciding interesting formulas of first-order real arithmetic. Understanding how such techniques for real arithmetic work is interesting for at least two reasons. First of all, it is important to understand why this miracle happens that something as complicated and expressive as first-order logic of real arithmetic is decidable. But it is also helpful to get an intuition about how real arithmetic decision procedures work. With such an understanding, you are better prepared to identify the limitations of these techniques, learn when they are likely not to work out in due time, and get a sense of what you can do to help arithmetic prove more complicated properties. For complex proofs, it is often very important to use your insights and intuitions about the system to help the prover along to scale more.

These lecture notes are loosely based on [Wei97, Pla10, Appendix D]. They add substantial intuition and motivation that is helpful for following the technical development. More information about virtual substitution can be found in the literature [Wei97]. See, e.g., [PQR09, Pas11] for an overview of other techniques for real arithmetic.

2 Framing the Miracle

First-order logic is an expressive logic in which many interesting properties and concepts can be expressed, analyzed, and proven. It is certainly significantly more expressive than propositional logic, which is decidable by NP-complete SAT solving.

In classical (uninterpreted) *first-order logic* (FOL), no symbol (except possibly equality) has a special meaning. There are only predicate symbols $p, q, r, \dots$ and function symbols $f, g, h, \dots$ whose meaning is subject to interpretation. And the domain that quantifiers range over is subject to interpretation. In particular, a formula of first-order logic is only valid if it holds true for all interpretations of all predicate and function symbols and all domains.

In contrast, *first-order logic of real arithmetic* ($\text{FOL}_{\mathbb{R}}$ or the theory of real-closed field arithmetic FOL_{RCF} [Pla10, Appendix D]) is interpreted, because its symbols have a special fixed interpretation. The only predicate symbols are $=, \geq, >, \leq, <, \neq$ and they mean exactly equality, greater-or-equals, greater-than, etc., and the only function symbols are $+, -, \cdot$, which mean exactly addition, subtraction, and multiplication of real numbers. Furthermore, the quantifiers quantify over the set $\mathbb{R}$ of all real numbers.[1]

The first special interpretation for symbols that comes to mind may not necessarily by the real numbers but maybe the natural numbers $\mathbb{N}$ with $+$ for addition and $\cdot$ for multiplication on natural numbers and where quantifiers range over the natural numbers. That gives the *first-order logic of natural numbers* ($\text{FOL}_{\mathbb{N}}$). Is $\text{FOL}_{\mathbb{N}}$ easier or harder than FOL? How do both compare to $\text{FOL}_{\mathbb{R}}$? What would happen compared to $\text{FOL}_{\mathbb{Q}}$, the first-order logic of rational numbers? $\text{FOL}_{\mathbb{Q}}$ is like $\text{FOL}_{\mathbb{R}}$ and $\text{FOL}_{\mathbb{N}}$, except that the rational numbers $\mathbb{Q}$ are used as the domain of quantification and interpretation of variables, rather than $\mathbb{R}$ and $\mathbb{N}$, respectively. How do those different flavors of first-order logic compare? How difficult is it to prove validity of logical formulas in each case?

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

[1] Respectively over another real-closed field, but that has been shown not to change validity [Tar51].

Uninterpreted first-order logic FOL is semidecidable, because there is a (sound and complete [Göd30]) proof procedure that is algorithmic and able to prove all true sentences of first-order logic [Her30]. The natural numbers are much more difficult. By Gödel's incompleteness theorem, first-order logic $\text{FOL}_{\mathbb{N}}$ of natural numbers does not have a sound and complete effective axiomatization. $\text{FOL}_{\mathbb{N}}$ is neither semidecidable nor cosemidecidable [Chu36]. There is neither an algorithm that can prove all valid formulas of $\text{FOL}_{\mathbb{N}}$ nor one that can disprove all formulas of $\text{FOL}_{\mathbb{N}}$ that are not valid. One way of realizing the inherent challenge of the logic of natural numbers is to use that not all questions about programs can be answered effectively (for example the halting problem of Turing machines is undecidable) [Chu36, Tur37], in fact "none" can [Ric53].

Yet, a miracle happened. Alfred Tarski proved in 1930 [Tar31, Tar51] that reals are much better behaved and that $\text{FOL}_{\mathbb{R}}$ is decidable, even though this seminal result remained unpublished for many years and only appeared in full in 1951 [Tar51].

The first-order logic $\text{FOL}_{\mathbb{Q}}$ of rational numbers, however, was shown to be undecidable [Rob49], even though rational numbers may appear to be so close to real numbers. Rationals are lacking something important: completeness (in the topological sense).

Note 1 (Overview of validity problems of first-order logics)**.**

Logic	*Validity*
FOL	*semidecidable*
$\text{FOL}_{\mathbb{N}}$	*not semidecidable nor cosemidecidable*
$\text{FOL}_{\mathbb{Q}}$	*not semidecidable nor cosemidecidable*
$\text{FOL}_{\mathbb{R}}$	*decidable*
$\text{FOL}_{\mathbb{C}}$	*decidable*

3 Quantifier Elimination

Alfred Tarski's seminal insight for deciding real arithmetic is based on quantifier elimination, i.e. the successive elimination of quantifiers from formulas so that the remaining formula is equivalent but structurally significantly easier. Why does eliminating quantifiers help? When evaluating a logical formula for whether it is true or false in a given state (i.e. an assignment of real numbers to all its free variables), arithmetic comparisons and polynomial terms are easy, because all we need to do is plug the numbers in and compute according to their semantics (recall Lecture 2). For example, for a state ν with $\nu(x) = 2$, we can easily evaluate the logical formula

$$x^2 > 2 \land 2x < 3 \lor x^3 < x^2$$

to true just by plugging in 2 for x. But quantifiers are difficult, because they require us to check for all possible values of a variable (in the case $\forall x\, F$) or to find exactly the right value for a variable that makes the formula true (in the case of $\exists x\, F$). The easiest formulas to evaluate are the ones that have no free variables (because then their value does not depend on the state) and that also have no quantifiers (because then there are no choices for the values of the quantified variables during the evaluation). Quantifier elimination can take a logical formula that is closed, i.e. has no free variables, and equivalently remove its quantifiers, so that it becomes easy to evaluate the formula to *true* or *false*. Quantifier elimination also works for formulas that still have free variables. Then it will eliminate all quantifiers in the formula but the original free variables will remain in the resulting formula, unless it simplifies in the quantifier elimination process.

Definition 1 (Quantifier elimination)**.** A first-order theory admits *quantifier elimination* if, with each formula ϕ, a quantifier-free formula $\mathrm{QE}(\phi)$ can be associated effectively that is equivalent, i.e. $\phi \leftrightarrow \mathrm{QE}(\phi)$ is valid (in that theory).

Theorem 2 (Tarski [Tar51])**.** *The first-order logic of real arithmetic admits quantifier elimination and is, thus, decidable.*

The operation QE is further assumed to evaluate ground formulas (i.e., without variables), yielding a decision procedure for closed formulas of this theory (i.e., formulas without free variables). For a closed formula ϕ, all it takes is to compute its quantifier-free equivalent $\mathrm{QE}(\phi)$ by quantifier elimination. The closed formula ϕ is closed, so has no free variables or other free symbols, and neither will $\mathrm{QE}(\phi)$. Hence, ϕ as well as its equivalent $\mathrm{QE}(\phi)$ are either equivalent to *true* or to *false*. Yet, $\mathrm{QE}(\phi)$ is quantifier-free, so which one it is can be found out simply by evaluating the (variable-free) concrete arithmetic in $\mathrm{QE}(\phi)$.

Example 3. Quantifier elimination uses the special structure of real arithmetic to express quantified arithmetic formulas equivalently without quantifiers and without using more free variables. For instance, QE yields the following equivalence:

$$\mathrm{QE}(\exists x\, (2x^2 + c \le 5)) \;\equiv\; c \le 5.$$

In particular, the formula $\exists x\, (2x^2 + c \le 5)$ is not valid, but only if $c \le 5$, as has been so aptly described by the outcome of the above quantifier elimination result.

Example 4. Quantifier elimination can be used to find out whether a first-order formula of real arithmetic is valid. Take $\exists x\, (2x^2 + c \le 5)$, for example. A formula is valid iff its universal closure is, i.e. the formula obtained by universally quantifying all free variables. After all, valid means that a formula is true for all intepretations. Hence, consider the universal closure $\forall c\, \exists x\, (2x^2 + c \le 5)$, which is a closed formula. Quantifier elimination might, for example, lead to

$$\mathrm{QE}(\forall c\, \exists x\, (2x^2{+}c \le 5)) \equiv \mathrm{QE}(\forall c\; \mathrm{QE}(\exists x\, (2x^2{+}c \le 5))) \equiv \mathrm{QE}(\forall c\, (c \le 5)) \equiv -100 \le 5 \wedge 5 \le 5 \wedge 100 \le 5$$

The resulting formula is still has no free variables but is now quantifier-free, so it can simply be evaluated arithmetically. Since the conjunct $100 \le 5$ evaluates to *false*, the universal closure $\forall c\, \exists x\, (2x^2 + c \le 5)$ is equivalent to *false* and, hence, the original formula $\exists x\, (2x^2 + c \le 5)$ is not valid (although still satisfiable for $c = 1$).

The complexity of Alfred Tarski's decision procedure is non-elementary, i.e. cannot be bounded by any tower of exponentials $2^{2^{2^{\cdots^{n}}}}$. Still, it was a seminal breakthrough because it showed reals to be decidable at all. It was not until another seminal result in 1949 by Julia Robinson, who proved the rationals to be undecidable [Rob49]. It took many further advances [Sei54, Coh69, KK71, Hör83, Eng93] and a major breakthrough by George Collins in 1975 [Col75] until more practical procedures had been found [Col75, CH91, Wei97]. The virtual substitution technique shown in this lecture has been implemented in Redlog [DS97], which has an interface for KeYmaera [PQ08].

4 Homomorphic Normalization

The first insight for defining quantifier elimination is to understand that the quantifier elimination operation commutes with almost all logical connectives, so that QE only needs to be defined for existential quantifiers. Especially, as soon as we understand how to eliminate existential quantifiers, universal quantifiers can be eliminated as well just by double negation.

$$\begin{aligned}
\mathrm{QE}(A \land B) &\equiv \mathrm{QE}(A) \land \mathrm{QE}(B)\\
\mathrm{QE}(A \lor B) &\equiv \mathrm{QE}(A) \lor \mathrm{QE}(B)\\
\mathrm{QE}(\neg A) &\equiv \neg\, \mathrm{QE}(A)\\
\mathrm{QE}(\forall x\, A) &\equiv \mathrm{QE}(\neg \exists x\, \neg A)
\end{aligned}$$

These transformations isolate existential quantifiers for quantifier elimination. In particular, it is sufficient if quantifier elimination focuses on existentially quantified variables. When using the QE operation inside out, i.e. when using it repeatedly to eliminate the inner-most quantifier to a quantifier-free equivalent and then again eliminating the inner-most quantifier, the quantifier elimination is solved if only we manage to solve it for $\exists x\, A$ with a quantifier-free formula A. If A is not quantifier-free, its quantifiers can be eliminated from inside out:

$$\mathrm{QE}(\exists x\, A) \equiv \mathrm{QE}(\exists x\ \mathrm{QE}(A)) \qquad\qquad \text{if } A \text{ not quantifier-free}$$

It is possible, although not necessary and not even necessarily helpful, to simply the form of A as well. The following transformations transform the kernel of a quantifier into negation normal form using deMorgan's equivalences.

$$\begin{aligned}
\mathrm{QE}(\exists x\, (A \lor B)) &\equiv \mathrm{QE}(\exists x\, A) \lor \mathrm{QE}(\exists x\, B)\\
\mathrm{QE}(\exists x\, \neg(A \land B)) &\equiv \mathrm{QE}(\exists x\, (\neg A \lor \neg B))\\
\mathrm{QE}(\exists x\, \neg(A \lor B)) &\equiv \mathrm{QE}(\exists x\, (\neg A \land \neg B))\\
\mathrm{QE}(\exists x\, \neg\neg A) &\equiv \mathrm{QE}(\exists x\, A)
\end{aligned}$$

Distributivity can be used to simplify the form of the quantifier-free *kernel* A to disjunctive normal form and split existential quantifiers over disjuncts:

$$\begin{aligned}
\mathrm{QE}(\exists x\,(A \land (B \lor C))) &\equiv \mathrm{QE}(\exists x\,((A \land B) \lor (A \land C))) \\
\mathrm{QE}(\exists x\,((A \lor B) \land C)) &\equiv \mathrm{QE}(\exists x\,((A \land C) \lor (B \land C))) \\
\mathrm{QE}(\exists x\,(A \lor B)) &\equiv \mathrm{QE}((\exists x\,A) \lor (\exists x\,B))
\end{aligned}$$

The remaining case to address is the case $\mathrm{QE}(\exists x\,(A \land B))$ where $A \land B$ is a purely conjunctive formula (yet it can have any number of conjuncts, not just two). Using the following normalizing equivalences,

$$\begin{aligned}
p = q &\equiv p - q = 0 \\
p \geq q &\equiv p - q \geq 0 \\
p > q &\equiv p - q > 0 \\
p \neq q &\equiv p - q \neq 0 \\
p \leq q &\equiv q - p \geq 0 \\
p < q &\equiv q - p > 0 \\
\neg(p \geq q) &\equiv p < q \\
\neg(p > q) &\equiv p \leq q \\
\neg(p = q) &\equiv p \neq q \\
\neg(p \neq q) &\equiv p = q
\end{aligned}$$

it is further possible to normalize all atomic formulas equivalently to one of the forms $p = 0, p > 0, p \geq 0, p \neq 0$. Since $p \neq 0 \equiv p > 0 \lor p < 0$, disequations $\neq$ are unnecessary *in theory* as well (although they are useful in practice).

5 Substitution Base

Virtual substitution is a quantifier elimination technique that is based on substituting extended terms into formulas virtually, i.e. without the extended terms[2] actually occurring in the resulting constraints.

Note 4. *Virtual substitution essentially leads to an equivalence of the form*

$$\exists x\,F \leftrightarrow \bigvee_{t \in T} A_t \land F_x^t \tag{1}$$

for a suitable finite *set T of extended terms that depends on the formula F and that gets substituted into F virtually, i.e. in a way that results in standard real arithmetic terms, not extended terms.*

[2] Being an *extended real term* really means it is not a real term, but somehow closely related. We will see more concrete extended real terms and how to get rid of them again later.

Such an equivalence is how quantifier elimination can work. Certainly if the right-hand side of (1) is true, then t is a witness for $\exists x\, F$. The key to establishing an equivalence of the form (1) is to ensure that if F has a solution (in the sense of $\exists x\, F$ being true), then F must hold for one of the cases in T. That is, T must cover all representative cases. If we were to choose all real numbers $T \overset{\text{def}}{=} \mathbb{R}$, then (1) would be trivially valid, but then the right-hand side is not a formula because it is uncountably infinitely long, which is even worse than the quantified form on the left-hand side. But if a finite set T is sufficient for the equivalence (1) and the extra formulas A_t are quantifier-free, then the right-hand side of (1) is structurally simpler than the left-hand side, even if it may be (sometimes significantly) less compact.

The various ways of virtually substituting various extended reals e into logical formulas equivalently without having to mention the actual extended reals is the secret of virtual substitution. The first step is to see that it is enough to define substitutions only on atomic formulas of the form $p = 0, p < 0, p \leq 0$ (or, just as well, on $p = 0, p > 0, p \geq 0$). If σ denotes such an extended substitution of θ for x, then σ lifts to arbitrary first-order formulas homomorphically[3] as follows

$$\begin{aligned}
\sigma(A \land B) &\equiv \sigma A \land \sigma B \\
\sigma(A \lor B) &\equiv \sigma A \lor \sigma B \\
\sigma(\neg A) &\equiv \neg \sigma A \\
\sigma(\forall y\, A) &\equiv \forall y\, \sigma A && \text{if } x \neq y \text{ and } x \notin \theta \\
\sigma(\exists y\, A) &\equiv \exists y\, \sigma A && \text{if } x \neq y \text{ and } x \notin \theta \\
\sigma(p = q) &\equiv \sigma(p - q = 0) \\
\sigma(p < q) &\equiv \sigma(p - q < 0) \\
\sigma(p \leq q) &\equiv \sigma(p - q \leq 0) \\
\sigma(p > q) &\equiv \sigma(q - p < 0) \\
\sigma(p \geq q) &\equiv \sigma(q - p \leq 0) \\
\sigma(p \neq q) &\equiv \sigma(\neg(p - q = 0))
\end{aligned}$$

This lifting applies the substitution σ to all subformulas, with minor twists on quantifiers for admissibility and normalization of atomic formulas into the forms $p = 0, p < 0, p \leq 0$ for which σ has been assumed to already have been defined.

6 Term Substitutions

Consider a formula of the form

$$\exists x\, (bx + c = 0 \land F) \tag{2}$$

where x does not occur in the terms b, c. Let's consider how a first mathematical solution to this formula might look like. The only solution that the conjunct $bx + c = 0$ has

[3] With a caveat on admissibility for quantifiers to avoid capture of variables.

is $x = -c/b$. Hence, the left conjunct in (2) only holds for $x = -c/b$, so formula (2) can only be true if F also holds for that single solution $-c/b$ in place of x. That is, formula (2) holds only if $F_x^{-c/b}$ does. Hence, (2) is equivalent to the formula $F_x^{-c/b}$, which is quantifier-free.

So, how can we eliminate the quantifier in (2) equivalently?

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

Most certainly, $F_x^{-c/b}$ is quantifier-free. But it is not exactly always equivalent to (2) and, thus, does not necessarily qualify as its quantifier eliminate form. Oh no! What we wrote down is a good intuitive start, but does not make any sense at all if $b = 0$, for then $-c/b$ would have been a rather ill-devised division by zero. Performing such divisions by zero sounds like a fairly shaky start for an equivalence transformation such as quantifier elimination. And certainly like a shaky start for anything that is supposed to turn into a proof.

Let's start over. The first conjunct in (2) has the only solution $x = -c/b$ **if** $b \neq 0$. In that case, indeed, (2) is equivalent to $F_x^{-c/b}$, because the only way for (2) to be true then is exactly when the second conjunct F holds for the solution of the first conjunct, i.e. when $F_x^{-c/b}$ holds. But there is, in general, no way of knowing whether evaluation could yield $b \neq 0$ or not, because b might be a complicated polynomial term that is only zero under some interpretations, not under all. Certainly if b is the zero polynomial, we know for sure. Or if b is a polynomial that is never zero, such as a sum of squares plus a positive constant. In general, if $b = 0$, then, the first conjunct in (2) has all numbers for x as solutions if $c = 0$ and, otherwise, has no solution at all if $c \neq 0$. In the latter case, $b = 0, c \neq 0$, (2) is false, because its first conjunct is already false. In the former case, $b = c = 0$, however, the first conjunct $bx + c = 0$ is trivial and does not impose any constraints on x, nor does it help for finding out a quantifier-free equivalent of (2). In that case $b = c = 0$, the trivial constraint will be dropped and the remaining formula will be considered recursively instead.

Note 5. *In the non-degenerate case $b \neq 0$, (2) can be rephrased into a quantifier-free equivalent over $\mathbb{R}$ as follows:*

$$b \neq 0 \rightarrow \big(\exists x\,(bx + c = 0 \land F) \leftrightarrow b \neq 0 \land F_x^{-c/b}\big) \tag{3}$$

All it takes is, thus, the ability to substitute the term $-c/b$ for x in the formula F. The division $-c/b$ that will occur in $F_x^{-c/b}$ for ordinary term substitutions can cause technical annoyances but at least it is well-defined, because $b \neq 0$ holds in that context.

7 Square Root $\sqrt{\cdot}$ Substitutions for Quadratics

Consider a formula of the form

$$\exists x\,(ax^2 + bx + c = 0 \land F) \tag{4}$$

where x does not occur in the terms a, b, c. The generic solution of its first conjunct is $x = (-b \pm \sqrt{b^2 - 4ac})/(2a)$, but that, of course, again depends on whether a could evaluate to zero, in which case linear solutions may be possible and the division by $2a$ is most certainly not well-defined. Whether a could be zero may again sometimes be hard to say when a is a polynomial term that has roots, but does not always evaluate to 0 either (which only the zero polynomial would). So let's be more careful this time to find an equivalent formulation right away for all possible cases of a, b, c.

The cases to consider are where the first conjunct is either a constant equation (in which case the equation is no interesting constraint on x) or a linear equation (in which case $x = -c/b$ is the solution Sect. 6) or a proper quadratic equation with $a \neq 0$ (in which case $x = (-b \pm \sqrt{b^2 - 4ac})/(2a)$ is the solution). The trivial equation $0 = 0$ when $a = b = c = 0$ is again useless, so another part of F would have to be considered in that case, and the equation $c = 0$ for $a = b = 0, c \neq 0$ is again *false*.

When $ax^2 + bx = 0$ is either a proper linear or a proper quadratic equation, its respective solutions single out the only points that can solve (4), so the only points in which it remains to be checked whether the second conjunct F also holds.

Theorem 5 (Virtual substitution of quadratic equations)**.** *For a quantifier-free formula F, the following equivalence is valid over $\mathbb{R}$:*

$$\begin{aligned}
&a \neq 0 \lor b \neq 0 \lor c \neq 0 \rightarrow \\
&\qquad \Big(\exists x\, (ax^2 + bx + c = 0 \land F) \leftrightarrow \\
&\qquad\quad a = 0 \land b \neq 0 \land F_x^{-c/b} \\
&\qquad\quad \lor\, a \neq 0 \land b^2 - 4ac \geq 0 \land \big(F_x^{(-b+\sqrt{b^2-4ac})/(2a)} \lor F_x^{(-b-\sqrt{b^2-4ac})/(2a)}\big)\Big)
\end{aligned} \tag{5}$$

The resulting formula on the right-hand side of the biimplication is quantifier-free and, thus, sounds like it could be chosen for $\mathrm{QE}(\exists x\, (ax^2 + bx + c = 0 \land F))$ as long as it is not the case that $a = b = c = 0$.

Note 7. *The important thing to notice, though, is that $(-b \pm \sqrt{b^2 - 4ac})/(2a)$ is not exactly a polynomial term, not even a rational term, because it involves a square root $\sqrt{\cdot}$. Hence, (5) is not generally a formula of first-order real arithmetic.*

Square roots are really not part of real arithmetic. But they can be defined, still, by appropriate quadratures. For example, the positive root $x = \sqrt{y}$ can be defined as $x^2 = y \land y \geq 0$. Let's find out how square roots such as $(-b \pm \sqrt{b^2 - 4ac})/(2a)$ can be substituted into first-order formulas systematically without the need for square roots in the resulting formula.

A *square root expression* is an expression of the form

$$(a + b\sqrt{c})/d$$

with polynomials $a, b, c, d \in \mathbb{Q}[x_1, \dots, x_n]$ of rational coefficients in the variables $x_1, \dots, x_n$ and, for well-definedness, $d \neq 0$. Square roots with the same $\sqrt{d}$ can be added and multiplied as expected:

$$\begin{aligned}
(a + b\sqrt{c})/d + (a' + b'\sqrt{c'})/d' &= ((ad' + da') + (bd' + db')\sqrt{c})/(dd') \\
((a + b\sqrt{c})/d) \cdot ((a' + b'\sqrt{c'})/d') &= ((aa' + bb'c) + (ab' + ba')\sqrt{c})/(dd')
\end{aligned}$$

Substituting $(a+b\sqrt{c})/d$ for a variable x in a polynomial term p, thus, leads to a square root $p_x^{(a+b\sqrt{c})/d} = (\tilde{a}+\tilde{b}\sqrt{c})/\tilde{d}$ with the same $\sqrt{c}$, because the arithmetic resulting from evaluating the polynomial only requires addition and multiplication.

Note 8. *This explains how a square root expression can be substituted in for a variable in a polynomial. Yet, the result is still a square root expression, which cannot be written down directly in first-order real arithmetic. Yet, as soon as a square root expression, say $(a+b\sqrt{c})/d$, appears in an atomic formula of first-order real arithmetic, the square root can be rephrased equivalently to disappear.*

Assume $d \neq 0 \land c \geq 0$ for well-definedness. For square-root-free expressions ($b = 0$) with just divisions, i.e. $(a+0\sqrt{c})/d$, the following equivalences hold:

$$\begin{aligned}
(a+0\sqrt{c})/d = 0 &\equiv a = 0\\
(a+0\sqrt{c})/d \leq 0 &\equiv ad \leq 0\\
(a+0\sqrt{c})/d < 0 &\equiv ad < 0\\
(a+0\sqrt{c})/d \neq 0 &\equiv ad \neq 0
\end{aligned}$$

Assume $d \neq 0 \land c \geq 0$ for well-definedness. For square root expressions $(a+b\sqrt{c})/d$ with arbitrary b, the following equivalences hold:

$$\begin{aligned}
&(a+b\sqrt{c})/d = 0 \equiv ab \leq 0 \land a^2 - b^2c = 0\\
&(a+b\sqrt{c})/d \leq 0 \equiv ad \leq 0 \land a^2 - b^2c \geq 0 \lor bd \leq 0 \land a^2 - b^2c \leq 0\\
&(a+b\sqrt{c})/d < 0 \equiv ad < 0 \land a^2 - b^2c > 0 \lor bd \leq 0 \land (ad < 0 \lor a^2 - b^2c < 0)\\
&(a+b\sqrt{c})/d \neq 0 \equiv ab > 0 \lor a^2 - b^2c \neq 0
\end{aligned}$$

This defines the substitution of a square root $(a+b\sqrt{c})/d$ for x into atomic formulas when normalizing atomic formulas appropriately[4]. The important thing to observe is that the result of this substitution does not introduce square root expressions nor divisions even though the square root expression $(a+b\sqrt{c})/d$ had the square root $\sqrt{c}$ and the division $/d$. Substitution of a square root $(a+b\sqrt{c})/d$ for x into a (quantifier-free) first-order formula F then works as usual by substitution in all atomic formulas (as defined in Sect. 5). Denote the result of such a substitution by $F_{\bar{x}}^{(a+b\sqrt{c})/d}$.

It is crucial to note that the *virtual substitution* of square root expression $(a+b\sqrt{c})/d$ for x in F giving $F_{\bar{x}}^{(a+b\sqrt{c})/d}$ is semantically equivalent to the result $F_x^{(a+b\sqrt{c})/d}$ of the literal substitution replacing x with $(a+b\sqrt{c})/d$, but operationally different, because the virtual substitution never introduces square roots or divisions. Because of their semantical equivalence, we use the same notation by abuse of notation.

Theorem 5 continues to hold when using the so-defined square root substitutions $F_{\bar{x}}^{(-b\pm\sqrt{b^2-4ac})/(2a)}$ that make (5) a valid formula of first-order real arithmetic, without

[4] E.g. $f > g \equiv f - g > 0$ and $f \leq g \equiv g \geq f$

square root expressions. In particular, since the fraction $-c/b$ also is a (somewhat impoverished) square root expression $(-c+0\sqrt{0})/b$, $F_{x}^{-c/b}$ in (5) can be formed using the square root substitution, so the quantifier-free right-hand side of (5) neither introduces square roots nor divisions.

With this virtual substitution, the right-hand side of the biimplication (5) can be chosen as $\mathrm{QE}(\exists x\,(ax^2+bx+c=0 \land F))$ if it is not the case that $a=b=c=0$.

When using square root substitutions, divisions could, thus, also have been avoided in the quantifier elimination (3) for the linear case. Thus, the right-hand side of (3) can be chosen as $\mathrm{QE}(\exists x\,(bx+c=0 \land F))$ if it is not the case that $b=c=0$.

Before going any further, it is helpful to notice that virtual substitutions admit a number of useful optimizations that make it more practical. For example, when substituting a square root expression $(a+b\sqrt{c})/d$ for a variable x in a polynomial p, the resulting square root expression $p_{x}^{(a+b\sqrt{c})/d} = (\tilde{a}+\tilde{b}\sqrt{c})/\tilde{d}$ has a higher power $\tilde{d}=d^k$ where k is the degree of p in variable x, just by inspecting the above definitions of addition and multiplication. Such larger powers of d can be avoided. Note the equivalences $(pq^3 \sim 0) \equiv (pq \sim 0)$ and, if $q \neq 0$, even $(pq^2 \sim 0 \equiv (p \sim 0)$ for arithmetic relations $\sim \in \{=,>,\geq,\neq,<,\leq\}$. Since $d \neq 0$ for well-definedness, the degree of d in the result $F_{x}^{(a+b\sqrt{c})/d}$ of the virtual substitution can be lowered to 0 or 1 depending on whether it occurs as an even or odd power.

Example 6. Using this principle to check under which circumstance the quadratic equality from (4) evaluates to $true$ requires a nontrivial number of computations to handle the virtual substitution of the respective roots of $ax^2+bx+c=0$ into F. What would happen if we tried to apply the same virtual substitution coming from this equation to $ax^2+bx+c=0$ itself? Imagine, for example, that $ax^2+bx+c=0$ shows up again in F. Let's only consider the case of quadratic solutions, i.e. where $a \neq 0$. And let's only consider the root $(-b+\sqrt{b^2-4ac})/(2a)$. The other cases are left as an exercise. First virtually substitute $(-b+\sqrt{b^2-4ac})/(2a)$ into the polynomial ax^2+bx+c:

$$
\begin{aligned}
& (ax^2+bx+c)_{x}^{(-b+\sqrt{b^2-4ac})/(2a)} \\
= \;& a((-b+\sqrt{b^2-4ac})/(2a))^2 + b((-b+\sqrt{b^2-4ac})/(2a)) + c \\
= \;& a((b^2+b^2-4ac+(-b-b)\sqrt{b^2-4ac})/(4a^2)) + (-b^2+b\sqrt{b^2-4ac})/(2a) + c \\
= \;& (ab^2+ab^2-4a^2c+(-ab-ab)\sqrt{b^2-4ac})/(4a^2) + (-b^2+2ac+b\sqrt{b^2-4ac})/(2a) \\
= \;& ((ab^2+ab^2-4a^2c)2a + (-b^2+2ac)4a^2 + ((-ab-ab)2a+b4a^2)\sqrt{b^2-4ac})/(4a^2) \\
= \;& (\cancel{2a^2b^2} + \cancel{2a^2b^2} - \cancel{8a^3c} + -\cancel{4a^2b^2} + \cancel{8a^3c} + (-\cancel{2a^2b} - \cancel{2a^2b} + \cancel{4a^2b})\sqrt{b^2-4ac})/(4a^2) \\
= \;& (0+0\sqrt{0})/1 = 0
\end{aligned}
$$

So $(ax^2+bx+c)_{x}^{(-b+\sqrt{b^2-4ac})/(2a)}$ is the zero square root expression? That is actually exactly as expected by construction, because $(-b \pm \sqrt{b^2-4ac})/(2a)$ is supposed to be the root of ax^2+bx+c in the case where $a \neq 0 \land b^2-4ac \geq 0$. In particular, if ax^2+bx+c occurs again in F as either an equation or inequality, its virtual substitute in the various cases is

$$
\begin{aligned}
(ax^2+bx+c=0)_{x}^{(-b+\sqrt{b^2-4ac})/(2a)} &\equiv ((0+0\sqrt{0})/1=0) \equiv (0\cdot 1=0) \equiv \mathit{true}\\
(ax^2+bx+c\leq 0)_{x}^{(-b+\sqrt{b^2-4ac})/(2a)} &\equiv ((0+0\sqrt{0})/1\leq 0) \equiv (0\cdot 1\leq 0) \equiv \mathit{true}\\
(ax^2+bx+c< 0)_{x}^{(-b+\sqrt{b^2-4ac})/(2a)} &\equiv ((0+0\sqrt{0})/1< 0) \equiv (0\cdot 1< 0) \equiv \mathit{false}\\
(ax^2+bx+c\neq 0)_{x}^{(-b+\sqrt{b^2-4ac})/(2a)} &\equiv ((0+0\sqrt{0})/1\neq 0) \equiv (0\cdot 1\neq 0) \equiv \mathit{false}
\end{aligned}
$$

And that makes sense as well. After all, the roots of $ax^2+bx+c=0$ satisfy the weak inequality $ax^2+bx+c\leq 0$ but not the strict inequality $ax^2+bx+c<0$. In particular, Theorem 5 could substitute the roots of $ax^2+bx+c=0$ also into the full formula $ax^2+bx+c=0\land F$ under the quantifier, but the formula resulting from the left conjunct $ax^2+bc+c=0$ will always simplify to *true* so that only the virtual substitution into F will remain.

Exercises

Exercise 1. Example 6 showed that $ax^2+bx+c=0$ simplifies to *true* for the virtual substitution of the root $(-b+\sqrt{b^2-4ac})/(2a)$. Show that the same thing happens for the root $(-b-\sqrt{b^2-4ac})/(2a)$ and the root $(-c+0\sqrt{0})/b$.

Exercise 2. Example 6 argued that the simplification of $ax^2+bx+c=0$ to *true* for the virtual substitution of the root $(-b+\sqrt{b^2-4ac})/(2a)$ is to be expected, because $(-b+\sqrt{b^2-4ac})/(2a)$ is a root of $ax^2+bx+c=0$ in the case where $a\neq 0\land b^2-4ac\geq 0$. Yet, what happens in the case where the extra assumption $a\neq 0\land b^2-4ac\geq 0$ does not hold? What is the value of the virtual substitution in that case? Is that a problem? Discuss carefully!

References

[CH91] George E. Collins and Hoon Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *J. Symb. Comput.*, 12(3):299–328, 1991.

[Chu36] Alonzo Church. A note on the Entscheidungsproblem. *J. Symb. Log.*, 1(1):40–41, 1936.

[Coh69] Paul J. Cohen. Decision procedures for real and p-adic fields. *Communications in Pure and Applied Mathematics*, 22:131–151, 1969.

[Col75] George E. Collins. Hauptvortrag: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In H. Barkhage, editor, *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.

[DBL12] *Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25–28, 2012*. IEEE, 2012.

[DS97] Andreas Dolzmann and Thomas Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bull.*, 31:2–9, 1997.

[Eng93] E. Engeler. *Foundations of Mathematics: Questions of Analysis, Geometry and Algorithmics*. Springer, 1993.

[Göd30] Kurt Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Mon. hefte Math. Phys.*, 37:349–360, 1930.

[Her30] Jacques Herbrand. Recherches sur la théorie de la démonstration. *Travaux de la Société des Sciences et des Lettres de Varsovie, Class III, Sciences Mathématiques et Physiques*, 33:33–160, 1930.

[Hör83] L. Hörmander. *The Analysis of Linear Partial Differential Operators II*, volume 257 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1983.

[KK71] Georg Kreisel and Jean-Louis Krivine. *Elements of mathematical logic: Model Theory*. North-Holland, 2 edition, 1971.

[Pas11] Grant Olney Passmore. *Combined Decision Procedures for Nonlinear Arithmetics, Real and Complex*. PhD thesis, School of Informatics, University of Edinburgh, 2011.

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

[Pla10] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14509-4.

[Pla12a] André Platzer. The complete proof theory of hybrid systems. In *LICS* [DBL12], pages 541–550. doi:10.1109/LICS.2012.64.

[Pla12b] André Platzer. Logics of dynamical systems. In *LICS* [DBL12], pages 13–24. doi:10.1109/LICS.2012.13.

[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.

[PQR09] André Platzer, Jan-David Quesel, and Philipp Rümmer. Real world verification. In Renate A. Schmidt, editor, *CADE*, volume 5663 of *LNCS*, pages 485–501. Springer, 2009. doi:10.1007/978-3-642-02959-2_35.

[Ric53] H. Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Trans. AMS*, 89:25–59, 1953.

[Rob49] Julia Robinson. Definability and decision problems in arithmetic. *J. Symb. Log.*, 14(2):98–114, 1949.

[Sei54] Abraham Seidenberg. A new decision method for elementary algebra. *Annals of Mathematics*, 60:365–374, 1954.

[Tar31] Alfred Tarski. Sur les ensembles définissables de nombres réels I. *Fundam. Math.*, 17:210–239, 1931.

[Tar51] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, 2nd edition, 1951.

[Tur37] Alan M. Turing. Computability and lambda-definability. *J. Symb. Log.*, 2(4):153–163, 1937.

[Wei97] Volker Weispfenning. Quantifier elimination for real algebra — the quadratic case and beyond. *Appl. Algebra Eng. Commun. Comput.*, 8(2):85–101, 1997.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Virtual Substitution & Real Arithmetic

André Platzer

Carnegie Mellon University
Lecture 19

1 Introduction

Reasoning about cyber-physical systems and hybrid systems requires understanding and handling their real arithmetic, which can be challenging, because cyber-physical systems can have complex behavior. Differential dynamic logic and its proof calculus [Pla08, Pla10, Pla12] reduce the verification of hybrid systems to real arithmetic. How arithmetic interfaces with proofs has already been discussed in Lecture 9 on Proofs & Arithmetic. How real arithmetic with linear and quadratic equations can be handled by virtual substitution has been shown in Lecture 18 on Virtual Substitution & Real Equations. Today's lecture shows how virtual substitution for quantifier elimination in real arithmetic extends to the case of linear and quadratic inequalities.

These lecture notes are loosely based on [Wei97, Pla10, Appendix D]. They add substantial intuition and motivation that is helpful for following the technical development. More information about virtual substitution can be found in the literature [Wei97]. See, e.g., [PQR09, Pas11] for an overview of other techniques for real arithmetic.

2 Recall: Square Root $\sqrt{\cdot}$ Substitutions for Quadratics

Recall the way to handle quantifier elimination for linear or quadratic equations from Lecture 18 on Virtual Substitution & Real Equations:

Theorem 1 (Virtual substitution of quadratic equations)**.** *For a quantifier-free formula F, the following equivalence is valid over $\mathbb{R}$:*

$$\begin{aligned}
&a \neq 0 \lor b \neq 0 \lor c \neq 0 \rightarrow \\
&\qquad\qquad \Big(\exists x\,(ax^2 + bx + c = 0 \land F) \leftrightarrow \\
&\qquad\qquad\quad a = 0 \land b \neq 0 \land F_x^{-c/b} \\
&\qquad\qquad \lor\, a \neq 0 \land b^2 - 4ac \geq 0 \land \big(F_x^{(-b+\sqrt{b^2-4ac})/(2a)} \lor F_x^{(-b-\sqrt{b^2-4ac})/(2a)}\big)\Big)
\end{aligned} \tag{1}$$

When using virtual substitutions of square roots from Lecture 18, the resulting formula on the right-hand side of the biimplication is quantifier-free and can be chosen for $\mathrm{QE}(\exists x\,(ax^2 + bx + c = 0 \land F))$ as long as it is not the case that $a = b = c = 0$. In case $a = b = c = 0$, another formula in F needs to be considered for directing quantifier elimination, because the equation $ax^2 + bx + c = 0$ is noninformative if $a = b = c = 0$, e.g. when a, b, c are the zero polynomials or even if they just have a common root.

The formula on the right-hand side of the biimplication in (1) is a formula in the first-order logic of real arithmetic when using the virtual substitution of square root expressions defined in Lecture 18.

3 Infinity ∞ Substitution

Theorem 1 address the case where the quantified variable occurs in a linear or quadratic equation. It might only occur in inequalities, however. Consider a formula of the form

$$\exists x\,(ax^2 + bx + c \leq 0 \land F) \tag{2}$$

Under the respective conditions on a, b, c from (1), the possible solutions $-c/b, (-b + \sqrt{d})/(2a), (-b - \sqrt{d})/(2a)$ from (1) continue to be options for solutions of (2), because one way of satisfying the weak inequality $ax^2 + bx + c \leq 0$ is by satisfying the equation $ax^2 + bx + c = 0$. So if F is *true* for any of those solutions of the quadratic equation (under the auspices of the additional constraints on a, b, c), then (2) holds as well.

Yet, if those points do not work out, the weak inequality in (2) allows for more possible solutions. For example, if $a = 0, b > 0$, then sufficiently small values of x would satisfy $0x^2 + bx + c \leq 0$. Also, if $a < 0$, then sufficiently small values of x would satisfy $ax^2 + bx + c \leq 0$, because x^2 grows faster than x and, thus the negative ax^2 ultimately overcomes any contribution of bx and c to the value of $ax^2 + bx + c$. But that would quickly diverge into the principle full substitution principle for the uninsightful case of $T \stackrel{\text{def}}{=} \mathbb{R}$ from Lecture 18.

Now, one possibility of pursuing this line of thought may be to substitute smaller and smaller values for x into (2) and see if that happens to work. There is a much better way though. The only really small value that would have to be substituted into (2) to see if it happens to work is one that is so negative that it is smaller than all others: $-\infty$, which is the lower limit of all negative real numbers. Alternatively, $-\infty$ can be understood as being "always as negative as needed, i.e. more negative than anything else". Think of $-\infty$ as being built out of elastic rubber so that it always ends up being smaller when being compared to any actual real number.

Let $\infty, -\infty$ be *positive and negative infinities*, respectively, i.e. choose extra elements $\infty, -\infty \notin \mathbb{R}$ with $-\infty < r < \infty$ for all $r \in \mathbb{R}$. Formulas of real arithmetic can be substituted with $\pm\infty$ for a variable x if the compactified reals $\mathbb{R} \cup \{\infty, -\infty\}$. Yet, just like with square root expressions, $\pm\infty$ do not actually need to ever occur in the resulting formula, because substitution of infinities can be defined differently. For example, $(x + 5 > 0)_x^\infty$ simplifies to *false*, while $(x + 5 < 0)_x^\infty$ simplifies to *true*.

Note 2. *Substitution of the infinity $-\infty$ for x into an atomic formula for a polynomial $p \overset{def}{=} \sum_{i=0}^n a_i x^i$ with polynomials a_i that do not contain x is defined by the following equivalences (accordingly for substituting ∞ for x).*

$$(p = 0)_x^{-\infty} \equiv \bigwedge_{i=0}^n a_i = 0 \tag{3}$$

$$(p \le 0)_x^{-\infty} \equiv (p < 0)_x^{-\infty} \lor (p = 0)_x^{-\infty} \tag{4}$$

$$(p < 0)_x^{-\infty} \equiv @_{-\infty}(p) < 0 \tag{5}$$

$$(p \ne 0)_x^{-\infty} \equiv \bigvee_{i=0}^n a_i \ne 0 \tag{6}$$

Lines (3) and (6) use that the only equation of real arithmetic that infinities $\pm\infty$ satisfy is the trivial equation $0 = 0$. Line (4) uses the equivalence $p \le 0 \equiv p < 0 \lor p = 0$. Line (5) uses a simple inductive definition based on the degree, $\deg(p)$, in the variable x of the polynomial p to characterize whether p is ultimately negative at $-\infty$ (or for sufficiently negative numbers):

Note 3. *Let $p \overset{def}{=} \sum_{i=0}^n a_i x^i$ with polynomials a_i that do not contain x. Whether p is ultimately negative (written $@_{-\infty}(p) < 0$) at $-\infty$ is easy to characterize:*

$$@_{-\infty}(p) < 0 \overset{def}{\equiv} \begin{cases} p < 0 & \text{if } deg(p) = 0 \\ (-1)^n a_n < 0 \lor (a_n = 0 \land @_{-\infty}(\sum_{i=0}^{n-1} a_i x^i) < 0) & \text{if } deg(p) > 0 \end{cases}$$

Substitution of ∞ for x into an atomic formula is defined similarly, except that the sign factor $(-1)^n$ disappears. Substitution of ∞ or of $-\infty$ for x into first-order formulas is then defined as in Lecture 18.

Example 2. Using this principle to check under which circumstance the quadratic inequality from (2) evaluates to *true* yields the answer from our earlier ad-hoc analysis of what happens for sufficiently small values of x:

$$(ax^2 + bx + c \leq 0)_{x}^{-\infty} \equiv (-1)^2 a < 0 \lor a = 0 \land ((-1)b < 0 \lor b = 0 \land c < 0)$$

In the same way, the virtual substitution can be used to see under which circumstance F would also evaluate to *true* for sufficiently small values of x, exactly when $F_{x}^{-\infty}$ holds. Note that (at least if $a \neq 0$), the virtual substitution of ∞ for x would not make sense to check (2) at, because in that case, the inequality $ax^2 + bx + c \leq 0$ is violated. That would be different for an inequality such as $ax^2 + bx + c \geq 0$.

The crucial thing to note is again that the *virtual substitution* of infinities $\pm\infty$ for x in F giving $F_{x}^{\pm\infty}$ is semantically equivalent to the result $F_{x}^{\pm\infty}$ of the literal substitution replacing x with $\pm\infty$, but operationally different, because the virtual substitution never introduces actual infinities. Because of their semantical equivalence, we use the same notation by abuse of notation.

4 Infinitesimal ε Substitutions

Theorem 1 address the case where the quantified variable occurs in a linear or quadratic equation and the virtual substitution in Sect. 3 adds the case of sufficiently small values for x. Consider a formula of the form

$$\exists x\, (ax^2 + bx + c < 0 \land F) \tag{7}$$

In this case, the roots from Theorem 1 will not help, because they satisfy the equation $ax^2 + bx + c = 0$ but not the strict inequality $ax^2 + bx + c < 0$. The virtual substitution of $-\infty$ for x from Sect. 3 still makes sense to consider, because that one might satisfy F and $ax^2 + bx + c < 0$. If $-\infty$ does not work, however, the solution of (7) could be near one of the roots of $ax^2 + bx + c = 0$, just slightly off so that $ax^2 + bx + c < 0$ is satisfied. How far off? Well, saying that exactly by any real number is again difficult, because any particular real number might already have been too large in absolute value, depending on the constraints in the remainder of F. Again, this calls for quantities that are always as small as we need them to be.

Sect. 3 used a negative quantity that is so small that it is smaller than all negative numbers and hence infinitely small (but infinitely large in absolute value). Analyzing (7) needs positive quantities that are infinitely small and hence also infinitely small in absolute value. Infinitesimals are positive quantities that are always smaller than all positive real numbers, i.e. "always as small as needed". Think of them as built out of elastic rubber so that they always shrink as needed when compared with any actual positive real number so that the infinitesimals end up being smaller than positive reals. Another way of looking at infinitesimals is that they are the multiplicative inverses of $\pm\infty$.

A positive infinitesimal $\infty > \varepsilon > 0$ is positive and an extended real that is infinitesimal, i.e. positive but smaller than all positive real numbers ($\varepsilon < r$ for all $r \in \mathbb{R}$ with $r > 0$). For all polynomials $p \in \mathbb{R}[x] \setminus \{0\}$, $\zeta \in \mathbb{R}$, the Taylor expansion of p around ζ evaluated at $\zeta + \varepsilon$ can be used to show:

1. $p(\zeta + \varepsilon) \neq 0$
   that is, infinitesimal are positive and always so small that they never yield roots of any equation, except the trivial zero polynomial. Whenever it looks like there might be a root, the infinitesimal just became a bit smaller. And nonzero univariate polynomials only have finitely many roots, so the infinitesimals will take care to avoid them.

2. $p(\zeta) \neq 0 \Rightarrow p(\zeta)p(\zeta + \varepsilon) > 0$,
   that is, p has constant sign on infinitesimal neighborhoods of nonroots ζ. If the neighborhood around ζ is small enough (and for an infinitesimal it will be), then the polynomial will not yet have changed sign then.

3. $0 = p(\zeta) = p'(\zeta) = p''(\zeta) = \cdots = p^{(k-1)}(\zeta) \neq p^{(k)}(\zeta) \Rightarrow p^{(k)}(\zeta)p(\zeta + \varepsilon) > 0$,
   that is the first nonzero derivative of p at ζ determines the sign of p in an infinitesimal neighborhood of ζ.

Note 4. *Substitution of an infinitesimal expression $e + \varepsilon$ with a square root expression e and a positive infinitesimal ε for x into a polynomial $p = \sum_{i=0}^{n} a_i x^i$ with polynomials a_i that do not contain x is defined by the following equivalences.*

$$(p = 0)_{x}^{e+\varepsilon} \equiv \bigwedge_{i=0}^{n} a_i = 0 \tag{8}$$

$$(p \leq 0)_{x}^{e+\varepsilon} \equiv (p < 0)_{x}^{e+\varepsilon} \lor (p = 0)_{x}^{e+\varepsilon} \tag{9}$$

$$(p < 0)_{x}^{e+\varepsilon} \equiv (@_{-}(p) < 0)_{x}^{e} \tag{10}$$

$$(p \neq 0)_{x}^{e+\varepsilon} \equiv \bigvee_{i=0}^{n} a_i \neq 0 \tag{11}$$

Lines (8) and (11) use that infinitesimals offsets satisfy no equation except the trivial equation 0=0 (case 1). Line (9) again uses the equivalence $p \leq 0 \equiv p < 0 \lor p = 0$. Line (10) checks that the sign of p at e is negative (which will make p inherit the same negative sign at $e + \varepsilon$ by case 2) or will immediately become negative right away using a recursive formulation of immediately becoming negative that uses higher derivatives (which determine the sign by case 3). The lifting to arbitrary quantifier-free formulas of real arithmetic is again by substitution into all atomic subformulas and equivalences such as $(p > q) \equiv (p - q > 0)$ as defined in Lecture 18. Note that, for the case $(p < 0)_{x}^{e+\varepsilon}$, the (non-infinitesimal) square root expression e gets virtually substituted in for x into a formula $@_{-}(p) < 0$, which characterizes whether p becomes negative immediately at or after x (which will be substituted by e).

Note 5. *Whether p is immediately negative, i.e. negative itself or with a derivative p' that makes it negative on an infinitesimal interval, can be characterized recursively:*

$$@_{-}(p) < 0 \overset{\mathit{def}}{\equiv} \begin{cases} p < 0 & \text{if } deg(p) = 0 \\ p < 0 \lor (p = 0 \land @_{-}(p')) & \text{if } deg(p) > 0 \end{cases}$$

Example 3. Using this principle to check under which circumstance the quadratic strict inequality from (7) evaluates to *true* at $(-b + \sqrt{b^2 - 4ac})/(2a) + \varepsilon$, i.e. right after its root $(-b + \sqrt{b^2 - 4ac})/(2a)$, leads to the following computation.

$$@_{-}(ax^2+bx+c) \equiv ax^2+bx+c < 0 \lor ax^2+bc+c = 0 \land (2ax+b < 0 \lor 2ax+b = 0 \land 2a < 0)$$

Hence,

$$\begin{aligned}
&(ax^2 + bx + c < 0)_{x}^{(-b+\sqrt{b^2-4ac})/(2a)+\varepsilon} \equiv (@_{-}(ax^2 + bx + c))_{x}^{(-b+\sqrt{b^2-4ac})/(2a)+\varepsilon} \equiv \\
&(ax^2 + bx + c < 0 \lor ax^2 + bc + c = 0 \land (2ax + b < 0 \lor 2ax + b = 0 \land 2a < 0))_{x}^{(-b+\sqrt{b^2-4ac})/(2a)} \\
&\equiv 0{\cdot}1 < 0 \lor 0 = 0 \land (\underbrace{(0 < 0 \lor 4a^2 \le 0 \land (0 < 0 \lor -4a^2(b^2 - 4ac) < 0))}_{2ax+b<0_{x}^{(-b+\sqrt{b^2-4ac})/(2a)}} \lor \underbrace{0 = 0}_{2ax+b=0_{x}^{\cdots}} \land \underbrace{2a1 < 0}_{2a<0}) \\
&\qquad \equiv 4a^2 \le 0 \land -4a^2(b^2 - 4ac) < 0 \lor 2a < 0
\end{aligned}$$

because the square root virtual substitution gives $(ax^2 + bx + c)_{x}^{(-b+\sqrt{b^2-4ac})/(2a)} = 0$ by construction (compare example from Lecture 18). The virtual substitution into the polynomial $2ax + b$ computes as follows:

$$\begin{aligned}
(2ax + b)_{x}^{(-b\pm\sqrt{b^2-4ac})/(2a)} &\equiv 2a \cdot (-b \pm \sqrt{b^2 - 4ac})/(2a) + b \\
&\equiv (-2ab + \pm 2a\sqrt{b^2 - 4ac})/(2a) + b \\
&\equiv (\cancel{-2ab} + \cancel{2ab} + \pm 2a\sqrt{b^2 - 4ac})/(2a)
\end{aligned}$$

The resulting formula can be further simplified internally to

$$(ax^2 + bx + c < 0)_{x}^{(-b+\sqrt{b^2-4ac})/(2a)+\varepsilon} \equiv 4a^2 \le 0 \land -4a^2(b^2 - 4ac) < 0 \lor 2a < 0 \equiv 2a < 0$$

because the first conjunct $4a^2 \le 0 \equiv a = 0$ and, with $a = 0$, the second conjunct simplifies to $-4a^2(b^2 - 4ac)_{a}^{0} = -0(b^2) < 0$, which is impossible in the reals. This answer makes sense. Because, indeed, exactly if $2a < 0$ will a quadratic polynomial still evaluate to $ax^2 + bx + c < 0$ right after its second root $(-b + \sqrt{b^2 - 4ac})/(2a)$.

Formulas such as this one ($2a > 0$) are the result of a quantifier elimination procedure. If the formula after quantifier elimination is either *true* or *false*, then you know for sure that the formula is valid (*true*) or unsatisfiable (*false*), respectively. If the result of quantifier elimination is *true*, for example, KeYmaera can close proof branches (marked by proof rule $\mathbb{R}$ in our sequent proofs). Yet, quantifier elimination can also return other formulas, such as $2a > 0$, which are equivalent to the formula where quantifier elimination has been applied. In particular, they identify exactly under which circumstance that respective quantified formula is true. This can be very useful for identifying the missing assumptions to make a proof work and the corresponding statement true.

The crucial thing to note is again that the *virtual substitution* of infinitesimal expressions $e + \varepsilon$ for x in F giving $F_{x}^{e+\varepsilon}$ is semantically equivalent to the result $F_x^{e+\varepsilon}$ of the literal substitution replacing x with $e + \varepsilon$, but operationally different, because the virtual substitution never introduces actual infinitesimals. Because of their semantical equivalence, we use the same notation by abuse of notation.

Computationally more efficient substitutions of infinitesimals have been reported elsewhere [BD07].

5 Quantifier Elimination by Virtual Substitution

The following quantifier elimination technique works for formulas with a quantified variable that occurs at most quadratically.

Theorem 4 (Virtual substitution of quadratic constraints [Wei97])**.** *Let F be a quantifier-free formula in which all atomic formulas are of the form $ax^2 + bx + c \sim 0$ for x-free polynomials a, b, c and $\sim \in \{=, \leq, <, \neq\}$, with corresponding discriminant $d \overset{def}{=} b^2 - 4ac$. Then $\exists x\, F$ is equivalent over $\mathbb{R}$ to the following quantifier-free formula:*

$$
\begin{aligned}
& F_{x}^{-\infty} \\
\lor & \bigvee_{\left(ax^2+bx+c\left\{\substack{= \\ \leq}\right\}0\right)\in F} \left(a = 0 \land b \neq 0 \land F_{x}^{-c/b} \lor a \neq 0 \land d \geq 0 \land \left(F_{x}^{(-b+\sqrt{d})/(2a)} \lor F_{x}^{(-b-\sqrt{d})/(2a)}\right)\right) \\
\lor & \bigvee_{\left(ax^2+bx+c\left\{\substack{\neq \\ <}\right\}0\right)\in F} \left(a = 0 \land b \neq 0 \land F_{x}^{-c/b+\varepsilon} \lor a \neq 0 \land d \geq 0 \land \left(F_{x}^{(-b+\sqrt{d})/(2a)+\varepsilon} \lor F_{x}^{(-b-\sqrt{d})/(2a)+\varepsilon}\right)\right)
\end{aligned}
$$

Proof. The proof first considers the literal substitution of square root expressions, infinities, and infinitesimals and then, as a second step, uses that the virtual substitutions that avoid square root expressions, infinities, and infinitesimals are equivalent.

The implication from the quantifier-free formula on the right-hand side (denoted G) to $\exists x\, F$ is obvious, because each disjunct of the quantifier-free formula has a conjunct of the form F_x^t for some (extended) term t even if it may be a square root expression or infinity or term involving infinitesimals.

The converse implication from $\exists x\, F$ to the quantifier-free formula depends on showing that the quantifier-free formula covers all possible representative cases and that the accompanying constraints on a, b, c, d are actually necessary.

It is enough to prove this for the case where all variables in F except x have concrete numeric real values, because the equivalence holds iff it holds in all states ν. By a fundamental property of real arithmetic called o-minimality, the set $\mathcal{S}(F)$ of all real values for x that satisfy F forms a finite union of (pairwise disjoint) intervals, because the polynomials in F only change signs at their roots, of which there only are finitely many now that the polynomials have become univariate, i.e. with the only variable x, since all free variables are evaluated to concrete real numbers in ν. Without loss of generality (by merging overlapping or adjacent intervals), we assume all those intervals to be maximal, i.e. no bigger interval would satisfy F. So F actually changes its truth-value at the lower and upper endpoints of these intervals (unless the interval is unbounded).

The endpoints of these intervals can be seen to be of the form $-c/b, (-b+\sqrt{d})/(2a), (-b-\sqrt{d})/(2a), \infty, -\infty$ for any of the polynomials in F, because those polynomials are at most quadratic and all roots of those polynomials are contained in that set. Hence, as usual, $-c/b \in \mathcal{S}(F)$ implies $a = 0, b \neq 0$, because that is the only case where $-c/b$ satisfies F, which has only at most quadratic polynomials, while $(-b+\sqrt{d})/(2a) \in \mathcal{S}(F)$ as well as $(-b-\sqrt{d})/(2a) \in \mathcal{S}(F)$ both imply that $a \neq 0$ and discriminant $d \geq 0$. So the side conditions for the roots considered in the quantifier-free formula are necessary for quadratic polynomials.

Now consider one interval $I \subseteq \mathcal{S}(F)$ (if there is none, $\exists x\, F$ is *false*). If I has no lower bound, then $F_{x}^{-\infty}$ is true by construction (by Sect. 3, the virtual substitution $F_{x}^{-\infty}$ is equivalent to the literal substitution $F_x^{-\infty}$ in ±∞-extended real arithmetic). Otherwise, let $\alpha \in \mathbb{R}$ be the lower bound of I. If $\alpha \in I$ (I is closed at the lower bound), then α is of the form $-c/b, (-b+\sqrt{d})/(2a), (-b-\sqrt{d})/(2a)$ for some equation $(ax^2 + bx + c = 0) \in F$ or weak inequality $(ax^2 + bx + c \leq 0) \in F$. Since the respective extra conditions on a, b, c, d hold, the quantifier-free formula evaluates to true. If, otherwise, $\alpha \notin I$ (I is open at the lower bound α), then α is of the form $-c/b, (-b+\sqrt{d})/(2a), (-b-\sqrt{d})/(2a)$ for some disequation $(ax^2 + bx + c \neq 0) \in F$ or strict inequality $(ax^2 + bx + c < 0) \in F$ and the interval I cannot be a single point. Thus, one of the infinitesimal increments $-c/b+\varepsilon, (-b+\sqrt{d})/(2a)+\varepsilon, (-b-\sqrt{d})/(2a)+\varepsilon$ is in $I \subseteq \mathcal{S}(F)$. Since the respective conditions a, b, c, d hold, the quantifier-free formula is again true. Hence, in either case, the quantifier-free formula is equivalent to $\exists x\, F$ in state ν. Since the state ν giving concrete real numbers to all free variables of $\exists x\, F$ was arbitrary, the same equivalence holds for all ν, which means that the quantifier-free formula (call it G) is equivalent to $\exists x\, F$. That is $G \leftrightarrow \exists x\, F$ is valid, i.e. $\vDash G \leftrightarrow \exists x\, F$. □

Optimizations are possible [Wei97] if there is only one quadratic occurrence of x, and that occurrence is not in an equation. If that occurrence is an equation, Theorem 1 already showed what to do. If there is only one occurrence of a quadratic inequality, the following variation works.

Note 7 ([Wei97]). *Let* $\left(Ax^2 + Bx + C \left\{\begin{smallmatrix}\leq\\<\\\neq\end{smallmatrix}\right\} 0\right) \in F$ *be the only quadratic occurrence of* x. *In that case,* $\exists x\, F$ *is equivalent over* $\mathbb{R}$ *to the following quantifier-free formula:*

$$\begin{aligned}
&A = 0 \land B \neq 0 \land F_{x}^{-C/B} \lor A \neq 0 \land F_{x}^{-B/(2A)}\\
\lor\ &F_{x}^{\infty} \lor F_{x}^{-\infty}\\
\lor\ &\bigvee_{\left(0x^2+bx+c\left\{\begin{smallmatrix}=\\\leq\end{smallmatrix}\right\}0\right)\in F} \left(b \neq 0 \land F_{x}^{-c/b}\right)\\
\lor\ &\bigvee_{\left(0x^2+bx+c\left\{\begin{smallmatrix}\neq\\<\end{smallmatrix}\right\}0\right)\in F} \left(b \neq 0 \land (F_{x}^{-c/b+\varepsilon} \lor F_{x}^{-c/b-\varepsilon})\right)
\end{aligned}$$

Further optimizations are possible if some signs of a, b are known, because several cases in the quantifier-free expansion then become impossible and can be simplified to *true* or *false* immediately. This helps simplify the formula in Theorem 4, because one of the cases $a = 0$ versus $a \neq 0$ might drop. But it also reduces the number of disjuncts in $F_{x}^{-\infty}$, see Example 3, and in the virtual substitutions of square roots (Lecture 18) and of infinitesimals (Sect. 4).

Theorem 4 also applies for polynomials of higher degrees in x if all those factor to polynomials of at most quadratic degree in x [Wei97]. Degree reduction is also possible by renaming based on the greatest common divisor of all powers of x that occur in F. If a quantified variable x occurs only with degrees that are multiples of an odd number d then virtual substitution can use $\exists x\, F(x^d) \equiv \exists y\, F(y)$. If x only occurs with degrees that are multiples of an even number d then $\exists x\, F(x^d) \equiv \exists y\, (y \geq 0 \land F(y))$. The cases with infinitesimals $+\varepsilon$ are only needed if x occurs in strict inequalities. The cases $F_{x}^{(-b+\pm\sqrt{d})/(2a)}$ are only needed if x occurs in equations or weak inequalities.

6 Summary

Virtual substitution is one technique for eliminating quantifiers in real arithmetic. It works for linear and quadratic constraints and can be extended to some cubic cases [Wei94]. Virtual substitution can be applied repeatedly from inside out to eliminate quantifiers. In each case, however, virtual substitution requires the eliminated variable to occur with small enough degrees only. Even if that was the case initially, it may stop to be the case after eliminating the innermost quantifier, because the degrees of the formulas resulting from virtual substitution may increase. In that case, degree optimizations and simplifications may sometimes work. If not, then other quantifier elimination techniques need to be used, which are based on semialgebraic geometry or model theory. Virtual substitution alone always works for mixed quadratic-linear formulas, i.e. those in which all quantified variables occur linearly except for one variable that occurs quadratically. In practice, however, many other cases turn out to work well with virtual substitution.

Exercises

Exercise 1. Consider

$$\exists x\,(ax^2 + bx + c \leq 0 \wedge F) \tag{12}$$

The virtual substitution of the roots of $ax^2 + bx + c = 0$ according to Sect. 2 as well as of $-\infty$ according to Sect. 3 will lead to

$$F_{x}^{-\infty} \vee a = 0 \wedge b \neq 0 \wedge F_{x}^{-c/b} \vee a \neq 0 \wedge b^2 - 4ac \geq 0 \wedge \big(F_{x}^{(-b+\sqrt{b^2-4ac})/(2a)} \vee F_{x}^{(-b-\sqrt{b^2-4ac})/(2a)}\big)$$

But when F is $-ax^2 + bx + e < 0$, then none of those cases necessarily works. Does that mean the result of virtual substitution is not equivalent to (12)? Where is the catch in this argument?

References

[BD07] Christopher W. Brown and James H. Davenport. The complexity of quantifier elimination and cylindrical algebraic decomposition. In Dongming Wang, editor, *ISSAC*, pages 54–60. ACM, 2007.

[Pas11] Grant Olney Passmore. *Combined Decision Procedures for Nonlinear Arithmetics, Real and Complex*. PhD thesis, School of Informatics, University of Edinburgh, 2011.

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

[Pla10] 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. Logics of dynamical systems. In *LICS*, pages 13–24. IEEE, 2012. doi:10.1109/LICS.2012.13.

[PQR09] André Platzer, Jan-David Quesel, and Philipp Rümmer. Real world verification. In Renate A. Schmidt, editor, *CADE*, volume 5663 of *LNCS*, pages 485–501. Springer, 2009. doi:10.1007/978-3-642-02959-2_35.

[Wei94] Volker Weispfenning. Quantifier elimination for real algebra — the cubic case. In *ISSAC*, pages 258–263, 1994.

[Wei97] Volker Weispfenning. Quantifier elimination for real algebra — the quadratic case and beyond. *Appl. Algebra Eng. Commun. Comput.*, 8(2):85–101, 1997.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Hybrid Systems & Games

André Platzer

Carnegie Mellon University

Lecture 20

1 Introduction

Hybrid systems have so far served us well throughout this course as a model for cyber-physical systems [Pla08, Pla10b]. Most definitely, hybrid systems can also serve as models for other systems that are not cyber-physical per se, i.e. they are not built as a combination of cyber and computing capabilities with physical capabilities. Some biological systems can be understood as hybrid systems, because they combine discrete and continuous dynamics. Or physical processes in which things happen at very different speeds, so where there is a slow process about which a continuous understanding is critical as well as a very fast process in which a discrete abstraction might be sufficient. Neither of those examples are particularly cyber-physical. Yet, nevertheless, they can have natural models as hybrid systems, because their fundamental characteristics is the interaction of discrete and continuous dynamics, which is exactly what hybrid systems are good for. Hence, despite their good match, not *all* hybrid systems are cyber-physical systems.

The converse is not true either, though. Not all cyber-physical systems are hybrid systems. The reason for that is usually not that cyber-physical systems would not involve both discrete and continuous dynamics, but, rather, that, in addition to those, they involve also other dynamical aspects. It is a common phenomenon in cyber-physical systems that they involve several dynamical aspects, which is why they are best understood as *multi-dynamical systems*, i.e. systems with multiple dynamical features [Pla12c, Pla12b, Pla11, Pla13]. And this does not only happen for cyber-physical systems but also for other systems. Some applications imply require more dynamical features than just discrete and continuous dynamics.

It is not going to be feasible to understand all those multi-dynamical system aspects at once in today's lecture. But today's lecture is going to introduce one very fundamental dynamical aspect: *adversarial dynamics* [Pla13]. Adversarial dynamics comes from multiple players on a hybrid system that are allowed to make their respective choices arbitrarily. The combination of discrete, continuous, and adversarial dynamics leads to *hybrid games*. Unlike hybrid systems, hybrid games allow choices in the system dynamics to be resolved adversarially by different players with different objectives.

Hybrid games are certainly necessary in situations where multiple agents actively compete. The canonical situation of a hybrid game would, thus, be RoboCup, where two teams of robots play robot soccer, moving around physically in space, controlled according to discrete computer decisions, and in active competition for scoring goals in opposite directions on the field. It turns out that hybrid games also come up for reasons of analytic competition, that is, where possible competition is assumed for the sake of a worst-case analysis.

Consider lab 5, for example, where a robot is interacting with a roguebot. You are in control of the robot, but somebody else is controlling the roguebot. Your objective is to control your robot so that it will not collide with the roguebot. That means you need to find some way of playing your control choices for your robot so that it will be safe for all possible control choices that the roguebot might do, after all you do not exactly know how the other roguebot is implemented. That could be considered as the robot playing a hybrid game with the roguebot in which your robot is trying to safely avoid collisions. The roguebot might behave sanely and tries to stay safe as well. But if your robot causes a collision, because it chose an action that was bad for the roguebot, your robot would certainly be faulty and sent back to the design table.

Alas, when you try to understand how you need to control your robot to stay safe, it can be instructive to think about what the worst-case action of a roguebot might be to make life difficult for you. And when your friendly course instructors try to demonstrate for you under which circumstance a simulation of your robot controller exhibits a faulty behavior, so that you can learn from the cases where your control does not work, they might be playing a hybrid game with you. If your robot wins and stays safe, this can very well be an indication of a strong robot design. But if your course TAs win and show an unsafe trace, you still win, because you learn more about the corner cases in your robot control design than when staring at simulation movies where everything is just fair-weather control.

If you think carefully again about lab 2, where your robot was put on a highway and had to find some way of being controlled to stay safe for all possible choices of the robot in front of it, then you will find that a hybrid game interpretation might be in order for that lab as well.

These lecture notes are based on [Pla13], where more information can be found on logic and hybrid games.

2 Choices & Nondeterminism

Note 1. *Hybrid systems involve choices. They manifest evidently in hybrid programs as nondeterministic choices $\alpha \cup \beta$ whether to run HP α or HPβ, in nondeterministic repetitions α^* where the choice is how often to repeat α, and in differential equations $x' = \theta \,\&\, H$ where the choice is how long to follow that differential equation. All those choices, however, have still been resolved in one way, i.e. by the same entity or player.*

In which way the various choices are resolved depends on the context. In the box modality $[\alpha]$ of differential dynamic logic [Pla08, Pla10b, Pla12c], the choices are resolved in *all possible ways* so that the modal formula $[\alpha]\phi$ expresses that formula ϕ holds for all ways how the choices in HP α could resolve. In the diamond modality $\langle\alpha\rangle$, instead, the choices are resolved in *some way* so that formula $\langle\alpha\rangle\phi$ expresses that formula ϕ holds for one way of resolving the choices in HP α.

In particular, choices in α help$\langle\alpha\rangle\phi$, because what this formulas calls for is *some* way of making ϕ happen after α. If α has many possible behaviors, this is easier to satisfy. Choices in α hurt $[\alpha]\phi$, however, because this formula requires ϕ to hold for all those choices. The more choices there are, the more difficult it is to make sure that ϕ holds after every single combination of those choices.

Note 2. *Choices in α either help uniformly (when they occur in $\langle\alpha\rangle\phi$) or make things more difficult uniformly (when they occur in $[\alpha]\phi$).*

That is why these various forms of choices in hybrid programs have been called *nondeterministic*. They are "unbiased". All possible resolutions of the choices in α could happen nondeterministically when running α. Which possibilities we care about (all or some) just depends on what the modal formula around it is that we consider.

3 Control & Dual Control

Another way of looking at the choices that are to be resolved during the runs of a hybrid program α is that they can be resolved by one player. Let's call her *Angel*. Whenever a choice is about to happen (by running the statements $\alpha \cup \beta$, α^*, or $x' = \theta \,\&\,$H), Angel is called upon to see how the choice is supposed to be resolved this time.

From that perspective, it sounds easy enough to add a second player. Let's call him *Demon*. Only so far, Demon will probably be rather bored after a while, when he realizes that he never actually gets to decide anything, because Angel has all the fun in choosing how the hybrid program world unfolds. So to keep Demon entertained, we need to introduce some choices that fall under Demon's control.

One thing, we could do to keep Demon interested in playing along is to add a pair of shiny new controls especially for him. They might be called $\alpha \cap \beta$ for Demon's choice between α or β as well as $\alpha^\times$ for repetition of α under Demon's control as well as an operation, say $x' = \theta \,\&\, H^d$, for continuous evolution under Demon's reign. But that would cause a lot of attention to Demon's control, which might make him feel overly majestic. Let's not do that, because we don't want Demon to get any ideas.

Instead, we will find it sufficient to add just a single operator to hybrid programs: the dual operator ^{d}. What α^d does is to give all control that Angel had in α to Demon, and, vice versa, all control that Demon had in α to Angel. The dual operator, thus, is a little bit like what happens when you turn a chessboard around by 180° in the middle of the game. Whoever played the choices of player White before suddenly controls Black, and whoever played Black now controls White. With just this single duality operator it turns out that Demon still gets his own set of controls ($\alpha \cap \beta$, $\alpha^\times$, $x' = \theta \,\&\, H^d$) by a suitable nesting of operators, but we did not have to give him those controls specifically. Yet, now those extra controls are not special but simply an aspect of a more fundamental principle: duality.

4 Hybrid Games

Differential game logic (dG$\mathcal{L}$) is a logic for studying properties of hybrid games. The idea is to describe the game form, i.e. rules, dynamics, and choices of the particular hybrid game of interest, using a program notation and to then study its properties by proving the validity of logical formulas that refer to the existence of winning strategies for objectives of those hybrid games. Even though hybrid game forms only describe the game *form* with its dynamics and rules and choices, not the actual objective, they are still simply called hybrid games. The objective for a hybrid game is defined in the modal logical formula that refers to that hybrid game form.

Definition 1 (Hybrid games)**.** The *hybrid games of differential game logic* dG$\mathcal{L}$ are defined by the following grammar (α, β are hybrid games, x a vector of variables, θ a vector of (polynomial) terms of the same dimension, H is a dG$\mathcal{L}$ formula or just a formula of first-order real arithmetic):

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

The only syntactical difference of hybrid games compared to hybrid programs for hybrid systems from Lecture 3 on Choice & Control is that, unlike hybrid programs, hybrid games allow the dual operator α^d. This minor syntactic change also requires us to reinterpret the meaning of the other operators in a much more flexible way to make sense of the presence of subgames within the games. The basic principle is that whenever there used to be nondeterminism in the hybrid program semantics, there will now be a choice of Angel in the hybrid game semantics. But don't be fooled. The parts of a such hybrid game may still be hybrid games, in which players interact, rather than just a single system running. So *all* operators of hybrid games still need a careful understanding as games, not just $\cdot^d$, because all operators can be applied to subgames.

The *atomic games* of dG$\mathcal{L}$ are assignments, continuous evolutions, and tests. In the *deterministic assignment game* (or discrete assignment game) $x := \theta$, the value of variable

x changes instantly and deterministically to that of θ by a discrete jump without any choices to resolve. In the *continuous evolution game* (or continuous game) $x' = \theta \,\&\, H$, the system follows the differential equation $x' = \theta$ where the duration is Angel's choice, but Angel is not allowed to choose a duration that would, at any time, take the state outside the region where formula H holds. In particular, Angel is deadlocked and loses immediately if H does not hold in the current state, because she cannot even evolve for duration 0 then without going outside H.[1] The *test game* or *challenge* $?H$ has no effect on the state, except that Angel loses the game immediately if d$\mathcal{GL}$ formula H does not hold in the current state. The test game $?H$ challenges Angel and she loses immediately if she fails. Angel does not win just because she passed the challenge $?H$, but at least the game continues. So passing challenges is a necessary condition to win games. Failing challenges, instead, immediately makes Angel lose.

The *compound games* of d$\mathcal{GL}$ are sequential, choice, repetition, and duals. The *sequential game* $\alpha;\beta$ is the hybrid game that first plays hybrid game α and, when hybrid game α terminates without a player having won already (so no challenge in α failed), continues by playing game β. When playing the *choice game* $\alpha \cup \beta$, Angel chooses whether to play hybrid game α or play hybrid game β. Like all the other choices, this choice is dynamic, i.e. every time $\alpha \cup \beta$ is played, Angel gets to choose again whether she wants to play α or β this time. The *repeated game* α^* plays hybrid game α repeatedly and Angel chooses, after each play of α that terminates without a player having won already, whether to play the game again or not, albeit she cannot choose to play indefinitely but has to stop repeating ultimately. Angel is also allowed to stop α^* right away after zero iterations of α. Most importantly, the *dual game* α^d is the same as playing the hybrid game α with the roles of the players swapped. That is Demon decides all choices in α^d that Angel has in α, and Angel decides all choices in α^d that Demon has in α. Players who are supposed to move but deadlock lose. Thus, while the test game $?H$ causes Angel to lose if formula H does not hold, the *dual test game* (or *dual challenge*) $(?H)^d$ instead causes Demon to lose if H does not hold.

For example, if α describes the game of chess, then α^d is chess where the players switch sides. If α, instead, describes the hybrid game corresponding to your lab 5 robot model where you are controlling a robot and your course instructors are controlling the roguebot, then α^d describes the dual game where you take control of the roguebot and the course instructors are stuck with your robot controls.

The dual operator ^{d} is the only syntactic difference of d$\mathcal{GL}$ for hybrid games compared to d$\mathcal{L}$ for hybrid systems [Pla08, Pla12a], but a fundamental one [Pla13], because it is the only operator where control passes from Angel to Demon or back. Without ^{d} all choices are resolved uniformly by Angel without interaction. The presence of ^{d} requires a thorough semantic generalization throughout the logic.

[1] Note that the most common case for H is a formula of first-order real arithmetic, but any d$\mathcal{GL}$ formula will work. In [Pla13], evolution domain constraints H turn out to be unnecessary, because they can be defined using hybrid games. In the ordinary differential equation $x' = \theta$, the term x' denotes the time-derivative of x and θ is a polynomial term that is allowed to mention x and other variables. More general forms of differential equations are possible [Pla10a, Pla10b], but will not be considered explicitly.

5 Differential Game Logic

Hybrid games describe how the world can unfold when Angel and Demon interact according to their respective control choices. They explain the rules of the game how Angel and Demon interact, but not who wins the game, nor what the respective objectives of the players are.[2] The winning conditions are specified by logical formulas of differential game logic. Modal formulas $\langle\alpha\rangle\phi$ and $[\alpha]\phi$ refer to hybrid games and the existence of winning strategies for Angel and Demon, respectively, in a hybrid game with a winning condition specified by a logical formula ϕ.

Definition 2 (dG$\mathcal{L}$ formulas)**.** The *formulas of differential game logic* dG$\mathcal{L}$ are defined by the following grammar (ϕ, ψ are dG$\mathcal{L}$ formulas, p is a predicate symbol of arity k, θ_i are (polynomial) terms, x a variable, and α is a hybrid game):

$$\phi, \psi \ ::= \ p(\theta_1, \dots, \theta_k) \mid \theta_1 \geq \theta_2 \mid \neg\phi \mid \phi \land \psi \mid \exists x\, \phi \mid \langle\alpha\rangle\phi \mid [\alpha]\phi$$

Other operators $>, =, \leq, <, \lor, \rightarrow, \leftrightarrow, \forall x$ can be defined as usual, e.g., $\forall x\, \phi \equiv \neg\exists x\, \neg\phi$. The modal formula $\langle\alpha\rangle\phi$ expresses that Angel has a winning strategy to achieve ϕ in hybrid game α, i.e. Angel has a strategy to reach any of the states satisfying dG$\mathcal{L}$ formula ϕ when playing hybrid game α, no matter what strategy Demon chooses. The modal formula $[\alpha]\phi$ expresses that Demon has a winning strategy to achieve ϕ in hybrid game α, i.e. a strategy to reach any of the states satisfying ϕ, no matter what strategy Angel chooses. Note that the same game is played in $[\alpha]\phi$ as in $\langle\alpha\rangle\phi$ with the same choices resolved by the same players. The difference between both dG$\mathcal{L}$ formulas is the player whose winning strategy they refer to. Both use the set of states where dG$\mathcal{L}$ formula ϕ is true as the winning states for that player. The winning condition is defined by the modal formula, α only defines the hybrid game form, not when the game is won, which is what ϕ does. Hybrid game α defines the rules of the game, including conditions on state variables that, if violated, cause the present player to lose for violation of the rules of the game. The dG$\mathcal{L}$ formulas $\langle\alpha\rangle\phi$ and $[\alpha]\neg\phi$ consider complementary winning conditions for Angel and Demon.

6 Demon's Controls

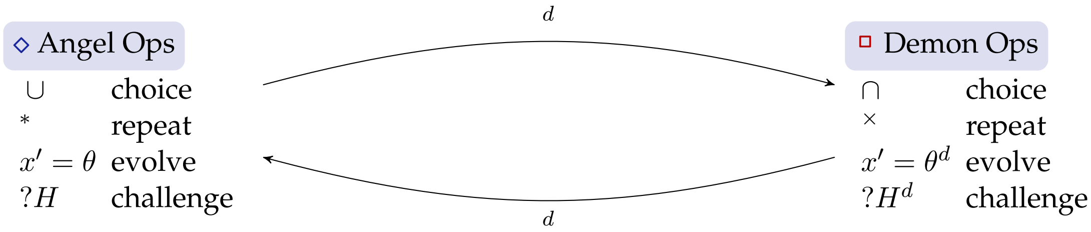
Angel has full control over all choices in each of the operators of hybrid games *except* when the operator ^{d} comes into play. All choices within the scope of (an odd number of) ^{d} belong to Demon, because ^{d} makes the players switch sides. Demon's controls, i.e. direct controls for Demon, can be defined using the duality operator ^{d} on Angel's controls.

Demonic choice between hybrid game α and β is $\alpha \cap \beta$, defined by $(\alpha^d \cup \beta^d)^d$, in which either the hybrid game α or the hybrid game β is played, by Demon's choice. *Demonic repetition* of hybrid game α is $\alpha^\times$, defined by $((\alpha^d)^*)^d$, in which α is repeated as often

[2] Except that players lose if they disobey the rules of the game by failing their respective challenges.

as Demon chooses to. In $\alpha^\times$, Demon chooses after each play of α whether to repeat the game, but cannot play indefinitely so he has to stop repeating ultimately. The *dual differential equation* $(x' = \theta \,\&\, H)^d$ follows the same dynamics as $x' = \theta \,\&\, H$ except that Demon chooses the duration, so he cannot choose a duration during which H stops to hold at any time. Hence he loses when H does not hold in the current state. Dual assignment $(x := \theta)^d$ is equivalent to $x := \theta$, because it involves no choices.

Angel's control operators and Demon's control operators correspond to each other by duality:



◇ Angel Ops	
$\cup$	choice
$*$	repeat
$x' = \theta$	evolve
$?H$	challenge

□ Demon Ops	
$\cap$	choice
$\times$	repeat
$x' = \theta^d$	evolve
$?H^d$	challenge

7 Semantics

What is the most elegant way of defining a semantics for differential game logic? How could a semantics be defined at all? First of all, the dG$\mathcal{L}$ formulas ϕ that are used in the postconditions of dG$\mathcal{L}$ modal formulas $\langle\alpha\rangle\phi$ and $[\alpha]\phi$ define the winning conditions for the hybrid game α. Thus, when playing the hybrid game α, we need to know the set of states in which the winning condition ϕ is satisfied. That set of states in which ϕ is true is denoted $[\![\phi]\!]^I$, which defines the semantics of ϕ.

The logic dG$\mathcal{L}$ has a denotational semantics. The dG$\mathcal{L}$ semantics defines, for each formula ϕ, the set $[\![\phi]\!]^I$ of states in which ϕ is true. For each hybrid game α and each set of winning states X, the dG$\mathcal{L}$ semantics defines the set $\varsigma_\alpha(X)$ of states from which Angel has a winning strategy to achieve X in hybrid game α, as well as the set $\delta_\alpha(X)$ of states from which Demon has a winning strategy to achieve X in α.

A *state* ν is a mapping from variables to $\mathbb{R}$. An *interpretation* I assigns a relation $I(p) \subseteq \mathbb{R}^k$ to each predicate symbol p of arity k. The interpretation further determines the set of states $\mathcal{S}$, which is isomorphic to a Euclidean space $\mathbb{R}^n$ when n is the number of relevant variables. For a subset $X \subseteq \mathcal{S}$ the complement $\mathcal{S} \setminus X$ is denoted $X^\complement$. Let ν_x^d denote the state that agrees with state ν except for the interpretation of variable x, which is changed to $d \in \mathbb{R}$. The value of term θ in state ν is denoted by $[\![\theta]\!]_\nu$. The denotational semantics of dG$\mathcal{L}$ formulas will be defined in Def. 3 by simultaneous induction along with the denotational semantics, $\varsigma_\alpha(\cdot)$ and $\delta_\alpha(\cdot)$, of hybrid games, defined later, because dG$\mathcal{L}$ formulas are defined by simultaneous induction with hybrid games. The *(denotational) semantics of a hybrid game* α defines for each interpretation I and each set of Angel's winning states $X \subseteq \mathcal{S}$ the *winning region*, i.e. the set of states $\varsigma_\alpha(X)$ from which Angel has a winning strategy to achieve X (whatever strategy Demon chooses). The *winning region* of Demon, i.e. the set of states $\delta_\alpha(X)$ from which Demon has a winning strategy to achieve X (whatever strategy Angel chooses) is defined later as well.

Definition 3 (dG$\mathcal{L}$ semantics)**.** The *semantics of a* dG$\mathcal{L}$ *formula* ϕ for each interpretation I with a corresponding set of states $\mathcal{S}$ is the subset $[\![\phi]\!]^I \subseteq \mathcal{S}$ of states in which ϕ is true. It is defined inductively as follows

1. $[\![p(\theta_1,\dots,\theta_k)]\!]^I = \{\nu \in \mathcal{S} \,:\, ([\![\theta_1]\!]_\nu,\dots,[\![\theta_k]\!]_\nu) \in I(p)\}$
2. $[\![\theta_1 \geq \theta_2]\!]^I = \{\nu \in \mathcal{S} \,:\, [\![\theta_1]\!]_\nu \geq [\![\theta_2]\!]_\nu\}$
3. $[\![\neg\phi]\!]^I = ([\![\phi]\!]^I)^\complement$
4. $[\![\phi \land \psi]\!]^I = [\![\phi]\!]^I \cap [\![\psi]\!]^I$
5. $[\![\exists x\, \phi]\!]^I = \{\nu \in \mathcal{S} \,:\, \nu_x^r \in [\![\phi]\!]^I \text{ for some } r \in \mathbb{R}\}$
6. $[\![\langle\alpha\rangle\phi]\!]^I = \varsigma_\alpha([\![\phi]\!]^I)$
7. $[\![[\alpha]\phi]\!]^I = \delta_\alpha([\![\phi]\!]^I)$

A dG$\mathcal{L}$ formula ϕ is *valid in* I, written $I \models \phi$, iff $[\![\phi]\!]^I = \mathcal{S}$. Formula ϕ is *valid*, $\vDash \phi$, iff $I \models \phi$ for all interpretations I.

8 Operational Game Semantics (informally)

A graphical illustration of the choices when playing hybrid games is depicted in Fig. 1. The nodes where Angel gets to decide are shown as diamonds ◇, the nodes where Demon decides are shown as boxes □. Circle nodes are shown when it depends on the remaining hybrid game which player it is that gets to decide. Dashed edges indicate Angel's actions, solid edges would indicate Demon's actions, while zigzag edges indicate that a hybrid game is played and the respective players move as specified by that game. The actions are the choice of time for $x' = \theta \,\&\, H$, the choice of playing the left or the right game for a choice game $\alpha \cup \beta$, and the choice of whether to stop or repeat in a repeated game α^*. This principle can be made rigorous in an operational game semantics [Pla13], which conveys the intuition of interactive game play for hybrid games, relates to game theory and descriptive set theory, but is also beyond the scope of these lecture notes.

As an example, consider the *filibuster formula*:

$$\langle (x := 0 \cap x := 1)^* \rangle x = 0 \tag{1}$$

It is Angel's choice whether to repeat (*), but every time Angel repeats, it is Demon's choice ($\cap$) whether to play $x := 0$ or $x := 1$. The game in this formula never deadlocks, because every player always has a remaining move (here even two). But it may appear as if the game had perpetual checks, because no strategy helps either player win the game; see Fig. 2. How could that happen and what can be done about it?

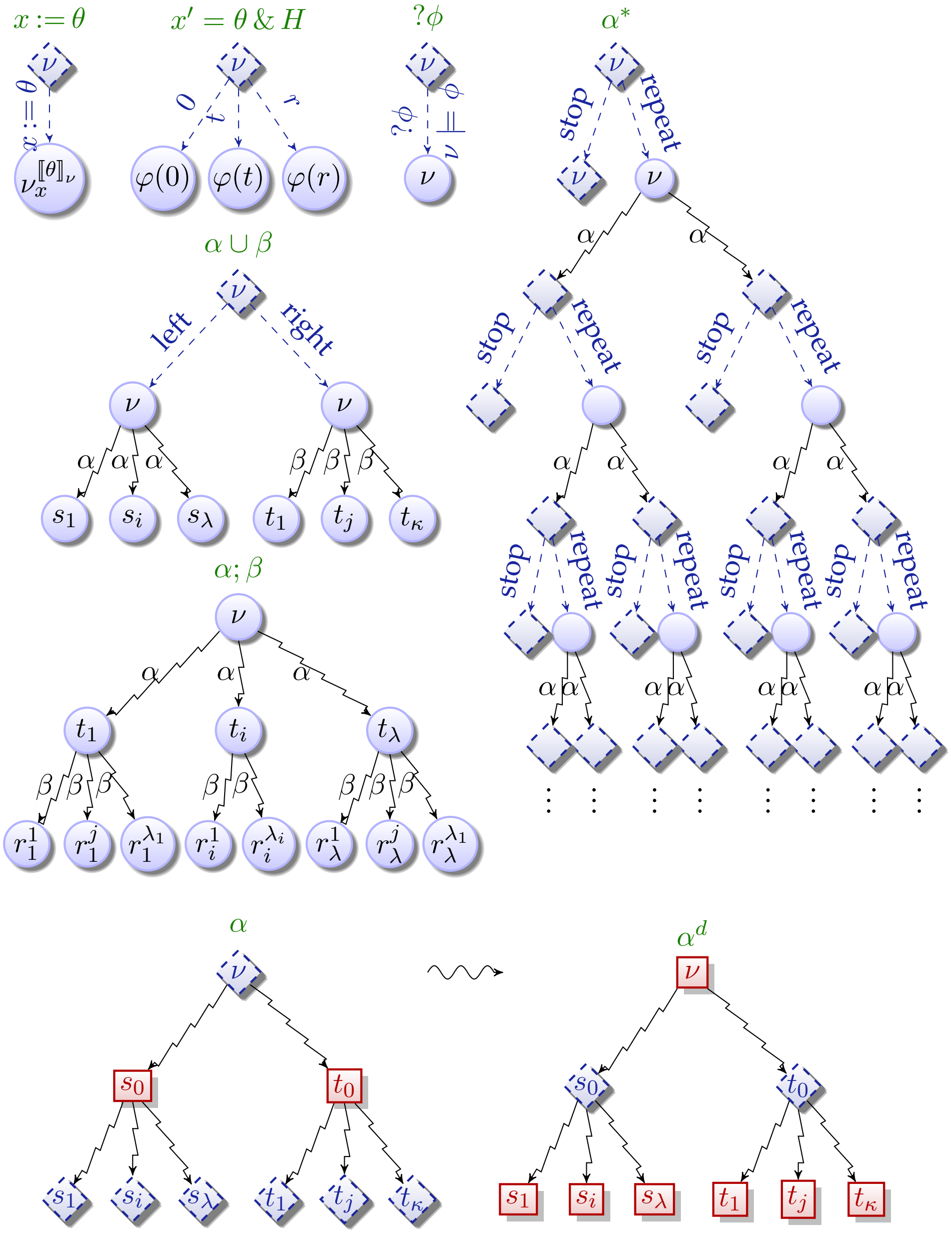


Figure 1: Operational game semantics for hybrid games of dG$\mathcal{L}$

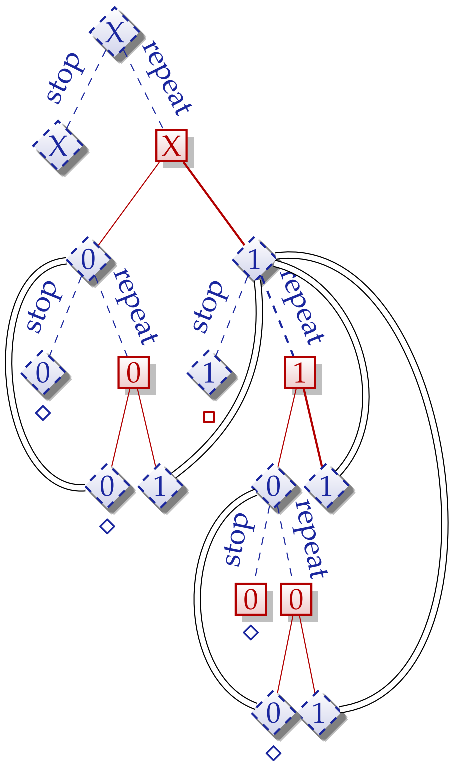


Figure 2: The filibuster game formula $\langle(x:=0 \cap x:=1)^*\rangle x = 0$ looks like it might be non-determined and not have a truth-value (unless $x = 0$ initially) when the strategies follow the thick actions. Angel's action choices are illustrated by dashed edges from dashed diamonds, Demon's action choices by solid edges from solid squares, and double lines indicate identical states with the same continuous state and a subgame of the same structure of subsequent choices. States where Angel wins are marked ◇ and states where Demon wins by □.

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

The mystery of the filibuster game can solved when we remember that the game still ultimately ought to stop. Angel is in charge of the * repetition and she can decide whether to stop or repeat. The filibuster game has no tests, so the winner only depends on the final state of the game. Angel wins a game play if $x = 0$ holds in the final state and Demon wins if $x \neq 0$ holds in the final state. What do the strategies indicated in Fig. 2 do? They postpone the end of the game forever, hence there would never be a final state in which it could be evaluated who won. That is, indeed, not a way for anybody to win anything. Yet, Angel was in charge of the repetition *, so it is really her fault if the game never comes to a stop to evaluate who won. Consequently, the semantics of hybrid games requires players to not repeat indefinitely. This will be apparent in the actual semantics of hybrid games, which is defined as a denotational semantics corresponding to winning regions.

It is of similar importance that the players cannot decide to follow a differential equation forever (duration ∞), because that would make

$$\langle (x' = 1^d; x := 0)^* \rangle x = 0 \tag{2}$$

non-determined.

Exercises

Exercise 1. Single player hybrid games, i.e. ^{d}-free hybrid games, are just hybrid programs. For each of the following formulas, convince yourself that it has the same meaning, whether you understand it as a differential dynamic logic formula with a hybrid systems or as a differential game logic formula with a hybrid game (that happens to have only a single player):

$$\begin{aligned}
&\langle x := 0 \cup x := 1 \rangle x = 0\\
&[x := 0 \cup x := 1] x = 0\\
&\langle (x := 0 \cup x := 1); ?x = 1 \rangle x = 0\\
&[(x := 0 \cup x := 1); ?x = 1] x = 0\\
&\langle (x := 0 \cup x := 1); ?x = 0 \rangle x = 0\\
&[(x := 0 \cup x := 1); ?x = 0] x = 0\\
&\langle (x := 0 \cup x := 1)^* \rangle x = 0\\
&[(x := 0 \cup x := 1)^*] x = 0\\
&\langle (x := 0 \cup x := x + 1)^* \rangle x = 0\\
&[(x := 0 \cup x := x + 1)^*] x = 0
\end{aligned}$$

Exercise 2. Consider the following dG$\mathcal{L}$ formulas and identify under which circumstance they are true

$$\langle (x := x+1; (x' = x^2)^d \cup x := x-1)^* \rangle\, (0 \le x < 1)$$

$$\langle (x := x+1; (x' = x^2)^d \cup (x := x-1 \cap x := x-2))^* \rangle (0 \le x < 1)$$

$$\begin{aligned}\langle (\ & (v := a \cup v := -a \cup v := 0);\\ & (w := b \cap w := -b \cap w := 0);\\ & x' = v, y' = w \ )^* \rangle\, (x-y)^2 \le 1\end{aligned}$$

Exercise 3. Is the following dGL formula valid? Can you identify some circumstances under which it is true? Or some circumstances under which it is false?

$$\begin{aligned}\Big\langle\Big(\ & (\omega := 1 \cup \omega := -1 \cup \omega := 0);\\ & (\varrho := 1 \cap \varrho := -1 \cap \varrho := 0);\\ & (x_1' = d_1, x_2' = d_2, d_1' = -\omega d_2, d_2' = \omega d_1, , y_1' = e_1, y_2' = e_2, e_1' = -\varrho e_2, e_2' = \varrho e_1)^d \Big)^* \Big\rangle\\ & \|x - y\| \le 1\end{aligned}$$

References

[DBL12] *Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25–28, 2012*. IEEE, 2012.

[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.

[Pla11] André Platzer. Stochastic differential dynamic logic for stochastic hybrid programs. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *CADE*, volume 6803 of *LNCS*, pages 431–445. Springer, 2011. doi:10.1007/978-3-642-22438-6_34.

[Pla12a] André Platzer. The complete proof theory of hybrid systems. In *LICS* [DBL12], pages 541–550. doi:10.1109/LICS.2012.64.

[Pla12b] André Platzer. Dynamic logics of dynamical systems. *CoRR*, abs/1205.4788, 2012. arXiv:1205.4788.

[Pla12c] André Platzer. Logics of dynamical systems. In *LICS* [DBL12], pages 13–24. doi:10.1109/LICS.2012.13.

[Pla13] André Platzer. A complete axiomatization of differential game logic for hybrid games. Technical Report CMU-CS-13-100R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January, Revised and extended in July 2013.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Winning Strategies & Regions

André Platzer

Carnegie Mellon University
Lecture 21

1 Introduction

This lecture continues the study of hybrid games and their logic, differential game logic [Pla13], that Lecture 20 on Hybrid Systems & Games started.

These lecture notes are based on [Pla13], where more information can be found on logic and hybrid games.

2 Semantics

What is the most elegant way of defining a semantics for differential game logic? How could a semantics be defined at all? First of all, the dG$\mathcal{L}$ formulas ϕ that are used in the postconditions of dG$\mathcal{L}$ modal formulas $\langle\alpha\rangle\phi$ and $[\alpha]\phi$ define the winning conditions for the hybrid game α. Thus, when playing the hybrid game α, we need to know the set of states in which the winning condition ϕ is satisfied. That set of states in which ϕ is true is denoted $[\![\phi]\!]^I$, which defines the semantics of ϕ.

The logic dG$\mathcal{L}$ has a denotational semantics. The dG$\mathcal{L}$ semantics defines, for each formula ϕ, the set $[\![\phi]\!]^I$ of states in which ϕ is true. For each hybrid game α and each set of winning states X, the dG$\mathcal{L}$ semantics defines the set $\varsigma_\alpha(X)$ of states from which Angel has a winning strategy to achieve X in hybrid game α, as well as the set $\delta_\alpha(X)$ of states from which Demon has a winning strategy to achieve X in α.

A *state* ν is a mapping from variables to $\mathbb{R}$. An *interpretation* I assigns a relation $I(p) \subseteq \mathbb{R}^k$ to each predicate symbol p of arity k. The interpretation further determines the set of states $\mathcal{S}$, which is isomorphic to a Euclidean space $\mathbb{R}^n$ when n is the number of relevant variables. For a subset $X \subseteq \mathcal{S}$ the complement $\mathcal{S} \setminus X$ is denoted $X^\complement$. Let ν_x^d denote the state that agrees with state ν except for the interpretation of variable x, which is changed to $d \in \mathbb{R}$. The value of term θ in state ν is denoted by $\llbracket\theta\rrbracket_\nu$. The denotational semantics of dG$\mathcal{L}$ formulas will be defined in Def. 1 by simultaneous induction along with the denotational semantics, $\varsigma_\alpha(\cdot)$ and $\delta_\alpha(\cdot)$, of hybrid games, defined later in Def. 2, because dG$\mathcal{L}$ formulas are defined by simultaneous induction with hybrid games. The *(denotational) semantics of a hybrid game* α defines for each interpretation I and each set of Angel's winning states $X \subseteq \mathcal{S}$ the *winning region*, i.e. the set of states $\varsigma_\alpha(X)$ from which Angel has a winning strategy to achieve X (whatever strategy Demon chooses). The *winning region* of Demon, i.e. the set of states $\delta_\alpha(X)$ from which Demon has a winning strategy to achieve X (whatever strategy Angel chooses) is defined subsequently in Def. 2 as well.

Definition 1 (dG$\mathcal{L}$ semantics)**.** The *semantics of a* dG$\mathcal{L}$ *formula* ϕ for each interpretation I with a corresponding set of states $\mathcal{S}$ is the subset $\llbracket\phi\rrbracket^I \subseteq \mathcal{S}$ of states in which ϕ is true. It is defined inductively as follows

1. $\llbracket p(\theta_1,\dots,\theta_k)\rrbracket^I = \{\nu \in \mathcal{S} : (\llbracket\theta_1\rrbracket_\nu,\dots,\llbracket\theta_k\rrbracket_\nu) \in I(p)\}$
2. $\llbracket\theta_1 \geq \theta_2\rrbracket^I = \{\nu \in \mathcal{S} : \llbracket\theta_1\rrbracket_\nu \geq \llbracket\theta_2\rrbracket_\nu\}$
3. $\llbracket\neg\phi\rrbracket^I = (\llbracket\phi\rrbracket^I)^\complement$
4. $\llbracket\phi \land \psi\rrbracket^I = \llbracket\phi\rrbracket^I \cap \llbracket\psi\rrbracket^I$
5. $\llbracket\exists x\, \phi\rrbracket^I = \{\nu \in \mathcal{S} : \nu_x^r \in \llbracket\phi\rrbracket^I \text{ for some } r \in \mathbb{R}\}$
6. $\llbracket\langle\alpha\rangle\phi\rrbracket^I = \varsigma_\alpha(\llbracket\phi\rrbracket^I)$
7. $\llbracket[\alpha]\phi\rrbracket^I = \delta_\alpha(\llbracket\phi\rrbracket^I)$

A dG$\mathcal{L}$ formula ϕ is *valid in* I, written $I \models \phi$, iff $\llbracket\phi\rrbracket^I = \mathcal{S}$. Formula ϕ is *valid*, $\vDash \phi$, iff $I \models \phi$ for all interpretations I.

Note that the semantics of $\langle\alpha\rangle\phi$ cannot be defined as it would in d$\mathcal{L}$ via

$$\llbracket\langle\alpha\rangle\phi\rrbracket^I = \{\nu \in \mathcal{S} : \omega \in \llbracket\phi\rrbracket^I \text{ for some } \omega \text{ with } (\nu,\omega) \in \rho(\alpha)$$

First of all, the reachability relation $(\nu,\omega) \in \rho(\alpha)$ is only defined when α is a hybrid program, not when it is a hybrid game. But the deeper reason is that the above shape is too harsh. Criteria of this shape would require Angel to single out a single state ν that satisfies the winning condition $\omega \in \llbracket\phi\rrbracket^I$ and then get to that state ω by playing α from ν. Yet all that Demon then has to do to spoil that plan is lead the play into a different state (e.g., one in which Angel would also have won) but which is different from the projected ω. More generally, winning into a single state is really difficult. Winning by leading the play into one of several states that satisfy the winning condition is more feasible. This is what the winning region $\varsigma_\alpha(\llbracket)\rrbracket^{\phi\phi}$ is supposed to capture.

3 Winning Regions

Def. 1 needs a definition of the winning regions $\varsigma_\alpha(\cdot)$ and $\delta_\alpha(\cdot)$ for Angel and Demon, respectively, in the hybrid game α. Rather than taking a detour for understanding those by operational game semantics (as in Lecture 20), the winning regions of hybrid games can be defined directly, giving a denotational semantics to hybrid games.

Definition 2 (Semantics of hybrid games)**.** The *semantics of a hybrid game* α is a function $\varsigma_\alpha(\cdot)$ that, for each interpretation I and each set of Angel's winning states $X \subseteq \mathcal{S}$, gives the *winning region*, i.e. the set of states $\varsigma_\alpha(X)$ from which Angel has a winning strategy to achieve X (whatever strategy Demon chooses). It is defined inductively as follows[a]

1. $\varsigma_{x:=\theta}(X) = \{\nu \in \mathcal{S} : \nu_x^{[\![\theta]\!]_\nu} \in X\}$
2. $\varsigma_{x'=\theta\,\&\,H}(X) = \{\varphi(0) \in \mathcal{S} : \varphi(r) \in X$ for some $r \in \mathbb{R}_{\geq 0}$ and (differentiable) $\varphi : [0,r] \to \mathcal{S}$ such that $\varphi(\zeta) \in [\![H]\!]^I$ and $\frac{\mathsf{d}\,\varphi(t)(x)}{\mathsf{d}t}(\zeta) = [\![\theta]\!]_{\varphi(\zeta)}$ for all $0 \leq \zeta \leq r\}$
3. $\varsigma_{?H}(X) = [\![H]\!]^I \cap X$
4. $\varsigma_{\alpha\cup\beta}(X) = \varsigma_\alpha(X) \cup \varsigma_\beta(X)$
5. $\varsigma_{\alpha;\beta}(X) = \varsigma_\alpha(\varsigma_\beta(X))$
6. $\varsigma_{\alpha^d}(X) = (\varsigma_\alpha(X^\complement))^\complement$

The *winning region* of Demon, i.e. the set of states $\delta_\alpha(X)$ from which Demon has a winning strategy to achieve X (whatever strategy Angel chooses) is defined inductively as follows

1. $\delta_{x:=\theta}(X) = \{\nu \in \mathcal{S} : \nu_x^{[\![\theta]\!]_\nu} \in X\}$
2. $\delta_{x'=\theta\,\&\,H}(X) = \{\varphi(0) \in \mathcal{S} : \varphi(r) \in X$ for all $r \in \mathbb{R}_{\geq 0}$ and (differentiable) $\varphi : [0,r] \to \mathcal{S}$ such that $\varphi(\zeta) \in [\![H]\!]^I$ and $\frac{\mathsf{d}\,\varphi(t)(x)}{\mathsf{d}t}(\zeta) = [\![\theta]\!]_{\varphi(\zeta)}$ for all $0 \leq \zeta \leq r\}$
3. $\delta_{?H}(X) = ([\![H]\!]^I)^\complement \cup X$
4. $\delta_{\alpha\cup\beta}(X) = \delta_\alpha(X) \cap \delta_\beta(X)$
5. $\delta_{\alpha;\beta}(X) = \delta_\alpha(\delta_\beta(X))$
6. $\delta_{\alpha^d}(X) = (\delta_\alpha(X^\complement))^\complement$

[a] The semantics of a hybrid game is not merely a reachability relation between states as for hybrid systems [Pla12], because the adversarial dynamic interactions and nested choices of the players have to be taken into account.

This notation uses $\varsigma_\alpha(X)$ instead of $\varsigma^I_\alpha(X)$ and $\delta_\alpha(X)$ instead of $\delta^I_\alpha(X)$, because the interpretation I that gives a semantics to predicate symbols in tests and evolution domains is clear from the context. Strategies do not occur explicitly in the dG$\mathcal{L}$ semantics, because it is based on the existence of winning strategies, not on the strategies themselves.

Just as the semantics d$\mathcal{L}$, the semantics of dG$\mathcal{L}$ is *compositional*, i.e. the semantics of a compound dG$\mathcal{L}$ formula is a simple function of the semantics of its pieces, and the semantics of a compound hybrid game is a function of the semantics of its pieces. Furthermore, existence of a strategy in hybrid game α to achieve X is independent of any game and dG$\mathcal{L}$ formula surrounding α, but just depends on the remaining game α itself and the goal X. By a simple inductive argument, this shows that one can focus on memoryless strategies, because the existence of strategies does not depend on the context, hence, by working bottom up, the strategy itself cannot depend on past states and choices, only the current state, remaining game, and goal. This also follows from a generalization of a classical result by Zermelo. Furthermore, the semantics is monotone, i.e. larger sets of winning states induce larger winning regions.

Lemma 3 (Monotonicity [Pla13]). *The semantics is* monotone, *i.e.* $\varsigma_\alpha(X) \subseteq \varsigma_\alpha(Y)$ *and* $\delta_\alpha(X) \subseteq \delta_\alpha(Y)$ *for all* $X \subseteq Y$.

Proof. A simple check based on the observation that X only occurs with an even number of negations in the semantics. For example, $X \subseteq Y$ implies $X^\complement \supseteq Y^\complement$, hence $\varsigma_\alpha(X^\complement) \supseteq \varsigma_\alpha(Y^\complement)$, so $\varsigma_{\alpha^d}(X) = (\varsigma_\alpha(X^\complement))^\complement \subseteq (\varsigma_\alpha(Y^\complement))^\complement = \varsigma_{\alpha^d}(Y)$. □

Before going any further, however, we need to define a semantics for repetition, which will turn out to be surprisingly difficult.

4 Examples

Consider the following examples and find out whether the formulas are valid or not.

$$\langle (x := x+1; (x' = x^2)^d \cup x := x-1)^* \rangle\,(0 \leq x < 1)$$

$$\langle (x := x+1; (x' = x^2)^d \cup (x := x-1 \cap x := x-2))^* \rangle (0 \leq x < 1)$$

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

$$\vDash \langle (x := x+1; (x' = x^2)^d \cup x := x-1)^* \rangle \, (0 \leq x < 1)$$

$$\nvDash \langle (x := x+1; (x' = x^2)^d \cup (x := x-1 \cap x := x-2))^* \rangle (0 \leq x < 1)$$

5 Advance Notice Repetitions

The semantics of repetition in hybrid systems was

$$\rho(\alpha^*) = \bigcup_{n \in \mathbb{N}} \rho(\alpha^n)$$

with $\alpha^{n+1} \equiv \alpha^n; \alpha$ and $\alpha^0 \equiv ?true$.

So the obvious candidate for the semantics of repetition in hybrid games might be

$$\varsigma_{\alpha^*}(X) \overset{?}{=} \bigcup_{n<\omega} \varsigma_{\alpha^n}(X)$$

where ω is the first infinite ordinal (if you have never seen ordinals before, just read $n < \omega$ as natural numbers $n \in \mathbb{N}$). Would that give the intended meaning to repetition? Is Angel forced to stop in order to win if the game of repetition would be played this way? Yes, she would, because, even though there is no bound on the number of repetitions that she can choose, for each natural number n, the resulting game $\varsigma_{\alpha^n}(X)$ is finite.

Would this definition capture the intended meaning of repeated game play?

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

The issue is that each way of playing a repetition this way would require Angel to choose a natural number $n \in \mathbb{N}$ of repetitions and *expose this number to Demon* when playing α^n so that he would know how often Angel decided to repeat.

That would lead to what is called the *advance notice semantics* for α^*, which requires the players to announce the number of times that game α will be repeated when the loop begins. The advance notice semantics defines $\varsigma_{\alpha^*}(X)$ as $\bigcup_{n<\omega} \varsigma_{\alpha^n}(X)$ where $\alpha^{n+1} \equiv \alpha^n; \alpha$ and $\alpha^0 \equiv ?true$ and defines $\delta_{\alpha^*}(X)$ as $\bigcap_{n<\omega} \delta_{\alpha^n}(X)$. When playing α^*, Angel, thus, announces to Demon how many repetitions n are going to be played when the game α^* begins and Demon announces how often to repeat $\alpha^\times$. This advance notice makes it easier for Demon to win loops α^* and easier for Angel to win loops $\alpha^\times$, because the opponent announces an important feature of their strategy immediately as opposed to revealing whether or not to repeat the game once more one iteration at a time as in Def. 2. Angel announces the number $n < \omega$ of repetitions when α^* starts.

The following formula, for example, turns out to be valid in dG$\mathcal{L}$ (see Fig. 1), but would not be valid in the advance notice semantics:

$$x = 1 \land a = 1 \rightarrow \langle ((x := a; a := 0) \cap x := 0)^* \rangle x \neq 1 \tag{1}$$

If, in the advance notice semantics, Angel announces that she has chosen n repetitions of the game, then Demon wins (for $a \neq 0$) by choosing the $x := 0$ option $n - 1$ times followed by one choice of $x := a; a := 0$ in the last repetition. This strategy would not work in the dG$\mathcal{L}$ semantics, because Angel is free to decide whether to repeat α^* after each repetition based on the resulting state of the game. The winning strategy for (1) indicated in Fig. 1(left) shows that this dG$\mathcal{L}$ formula is valid.

Since the advance notice semantics misses out on the existence of perfectly reasonable winning strategies, dG$\mathcal{L}$ does not choose this semantics. Nevertheless, the advance notice semantics can be a useful semantics to consider for other purposes [QP12].

6 ω-Strategic Semantics

The trouble with the semantics in Sect. 5 is that Angel's move for the repetition reveals too much to Demon, because Demon can inspect the remaining game α^n to find out once and for all how long the game will be played before he has to do his first move.

Let's try to undo this. Another alternative choice for the semantics would have been to allow only arbitrary finite iterations of the strategy function for computing the winning region by using the *ω-strategic semantics*, which defines

$$\varsigma_{\alpha^*}(X) \overset{?}{=} \varsigma_\alpha^\omega(X) = \bigcup_{n<\omega} \varsigma_\alpha^n(X)$$

along with a corresponding definition for $\delta_{\alpha^*}(X)$. All we need to do for this is define what it means to nest the winning region construction. For any winning condition

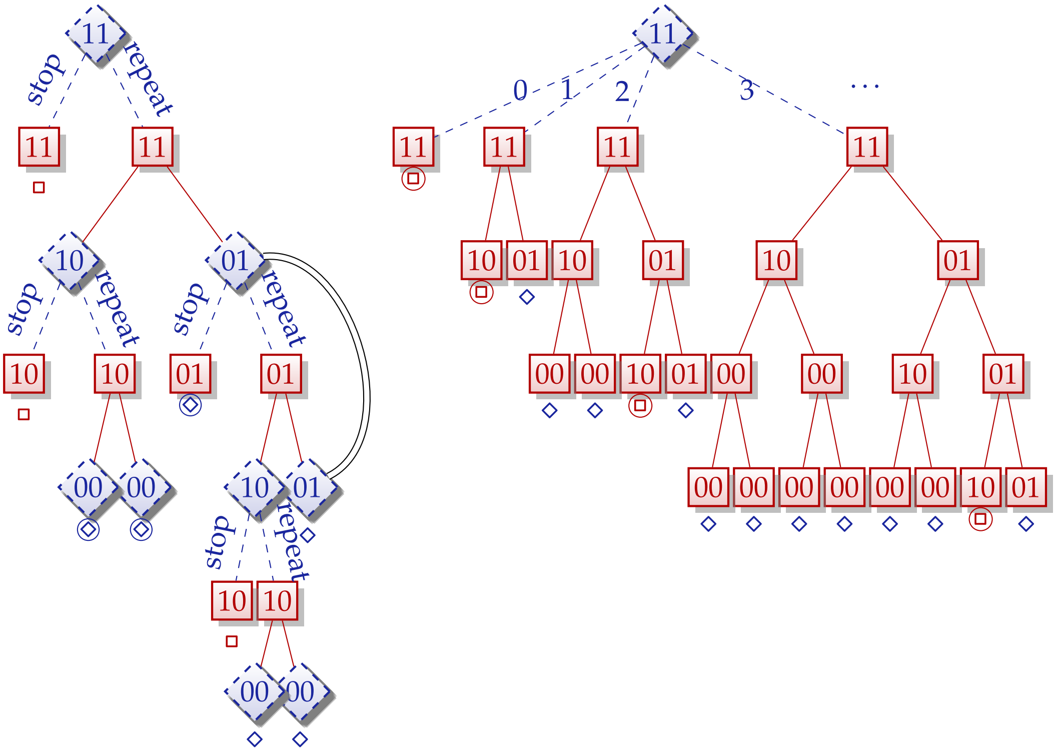


Figure 1: Game trees for $x = 1 \land a = 1 \rightarrow \langle\alpha^*\rangle x \neq 1$ with game $\alpha \equiv (x := a; a := 0) \cap x := 0$ (notation: x, a). **(left)** valid in dG$\mathcal{L}$ by strategy "repeat once and repeat once more if $x = 1$, then stop" **(right)** false in advance notice semantics by the strategy "$n-1$ choices of $x := 0$ followed by $x := a; a := 0$ once", where n is the number of repetitions Angel announced

$X \subseteq \mathcal{S}$ the iterated winning region of α is defined inductively as:

$$\begin{aligned}\varsigma_\alpha^0(X) &\overset{\text{def}}{=} X\\ \varsigma_\alpha^{\kappa+1}(X) &\overset{\text{def}}{=} X \cup \varsigma_\alpha(\varsigma_\alpha^\kappa(X))\end{aligned}$$

Does this give the right semantics for repetition of hybrid games? Does it match the existence of winning strategies that we were hoping to define? See Fig. 2 for an illustration.

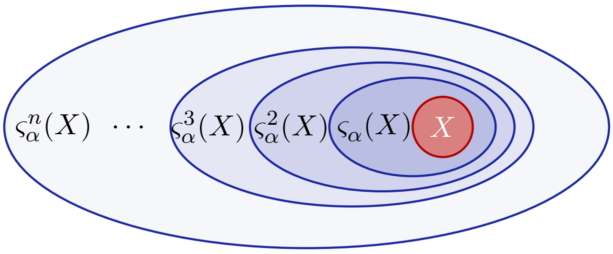


Figure 2: Iteration $\varsigma_\alpha^n(X)$ of $\varsigma_\alpha(\cdot)$ from winning condition X.

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

The surprising answer is *no* for a very subtle but also very fundamental reason. The existence of winning strategies for α^* does not coincide with the ωth iteration of α. This will be investigated further in the next lecture.

References

[Pla12] André Platzer. The complete proof theory of hybrid systems. In *LICS*, pages 541–550. IEEE, 2012. doi:10.1109/LICS.2012.64.

[Pla13] André Platzer. A complete axiomatization of differential game logic for hybrid games. Technical Report CMU-CS-13-100R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January, Revised and extended in July 2013.

[QP12] Jan-David Quesel and André Platzer. Playing hybrid games with KeYmaera. In Bernhard Gramlich, Dale Miller, and Ulrike Sattler, editors, *IJCAR*, volume 7364 of *LNCS*, pages 439–453. Springer, 2012. doi:10.1007/978-3-642-31365-3_34.

15-424: **Foundations of Cyber-Physical Systems**

Lecture Notes on
Winning & Proving Hybrid Games

André Platzer

Carnegie Mellon University
Lecture 22

1 Introduction

This lecture continues the study of hybrid games and their logic, differential game logic [Pla13], that Lecture 20 on Hybrid Systems & Games and Lecture 21 on Winning Strategies & Regions started.

These lecture notes are based on [Pla13], where more information can be found on logic and hybrid games.

2 Deficiencies of the ω-Strategic Semantics

Lecture 21 on Winning Strategies & Regions raised the question whether the semantics of repetition could be defined by the *ω-strategic semantics*

$$\varsigma_{\alpha^*}(X) \overset{?}{=} \varsigma_\alpha^\omega(X) = \bigcup_{n<\omega} \varsigma_\alpha^n(X)$$

For winning condition $X \subseteq \mathcal{S}$ the iterated winning region of α is defined inductively:

$$\varsigma_\alpha^0(X) \overset{\text{def}}{=} X$$
$$\varsigma_\alpha^{\kappa+1}(X) \overset{\text{def}}{=} X \cup \varsigma_\alpha(\varsigma_\alpha^\kappa(X))$$

Does this give the right semantics for repetition of hybrid games? Does it match the existence of winning strategies that we were hoping to define?

Would the following dGL formula be valid in the ω-strategic semantics?

$$\langle (x := 1; x' = 1^d \cup x := x - 1)^* \rangle \, (0 \le x < 1) \tag{1}$$

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

Abbreviate

$$\langle(\underbrace{\underbrace{x:=1;x'=1^d}_{\beta} \cup \underbrace{x:=x-1}_{\gamma}}_{\alpha})^*\rangle\,(0 \leq x < 1)$$

It is easy to see that $\varsigma_\alpha^\omega([0,1)) = [0,\infty)$, because $\varsigma_\alpha^n([0,1)) = [0,n)$ for all $n \in \mathbb{N}$ by a simple inductive proof (recall $\alpha \equiv \beta \cup \gamma$):

$$\begin{aligned}
&\varsigma^1_{\beta\cup\gamma}([0,1)) = [0,1) \\
&\varsigma^{n+1}_{\beta\cup\gamma}([0,1)) = [0,1) \cup \varsigma_{\beta\cup\gamma}(\varsigma^n_{\beta\cup\gamma}([0,1))) \overset{\text{IH}}{=} [0,1) \cup \varsigma_{\beta\cup\gamma}([0,n)) \\
&= [0,1) \cup \varsigma_{\beta\cup\gamma}([0,n)) \cup \varsigma_\beta([0,n)) = [0,1) \cup \emptyset \cup [1,n+1) = [0,n+1)
\end{aligned}$$

Consequently,

$$\varsigma_\alpha^\omega([0,1)) = \bigcup_{n<\omega} \varsigma_\alpha^n([0,1)) = \bigcup_{n<\omega} [0,n) = [0,\infty)$$

Hence, the $\omega-$semantics would indicate that the hybrid game (1) can exactly be won from all initial states in $[0,\infty)$, that is, for all initial states that satisfy $0 \leq x$.

Unfortunately, this is quite some nonsense. Indeed, the hybrid game in dG$\mathcal{L}$ formula (1) can be won from all initial states that satisfy $0 \leq x$. But it can also be won from other initial states! So the ω-strategic semantics $\varsigma_\alpha^\omega([0,1))$ misses out on winning states. It is way too small for a winning region. There are cases, where the ω-semantics is minuscule compared to the true winning region and arbitrarily far away from the truth [Pla13].

In (1), this ω-level of iteration of the strategy function for winning regions misses out on Angel's perfectly reasonable winning strategy "first choose $x:=1;x'=1^d$ and then always choose $x:=x-1$ until stopping at $0 \leq x < 1$". This winning strategy wins from every initial state in $\mathbb{R}$, which is a much bigger set than $\varsigma_\alpha^\omega([0,1)) = [0,\infty)$.

Now this is the final answer for the winning region of (1). In particular, the dG$\mathcal{L}$ formula (1) is valid. Yet, is there a direct way to see that $\varsigma_\alpha^\omega([0,1)) = [0,\infty)$ is not the final answer for (1) without having to put the winning region computations aside and constructing a separate ingenious winning strategy?

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

The crucial observation is the following. The fact $\varsigma_\alpha^\omega([0,1)) = [0,\infty)$ shows that the hybrid game in (1) can be won from all nonnegative initial values with at most ω ("first countably infinitely many") steps. Let's recall how the proof worked, which showed $\varsigma_\alpha^n([0,1)) = [0,n)$ for all $n \in \mathbb{N}$. Its inductive step basically showed that if, for whatever reason (by inductive hypothesis really), $[0,n)$ is in the winning region, then $[0,n+1)$ also is in the winning region by simply applying $\varsigma_\alpha(\cdot)$ to $[0,n)$.

How about doing exactly that again? For whatever reason (i.e. by the above argument), $[0,\infty)$ is in the winning region. Doesn't that mean that $\varsigma_\alpha([0,\infty))$ should again be in the winning region by exactly the same inductive argument above?

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

Note 1. *Whenever a set Y is in the winning region $\varsigma_{\alpha^*}(X)$ of repetition, then $\varsigma_\alpha(Y)$ also should be in the winning region $\varsigma_{\alpha^*}(X)$, because it is just one step away from Y and α^* could simply repeat once more.*

Thus, the winning region $\varsigma_{(\beta\cup\gamma)^*}([0,\infty))$ should also contain

$$\varsigma_{\beta\cup\gamma}([0,\infty)) = \varsigma_\beta([0,\infty)) \cup \varsigma_\gamma([0,\infty)) = \mathbb{R} \cup [0,\infty) = \mathbb{R}$$

Beyond that, the winning region cannot contain anything else, because $\mathbb{R}$ is the whole state space. And, indeed, trying to use the winning region construction once more on $\mathbb{R}$ does not change the result:

$$\varsigma_{\beta\cup\gamma}(\mathbb{R}) = \varsigma_\beta(\mathbb{R}) \cup \varsigma_\gamma(\mathbb{R}) = \mathbb{R} \cup [0,\infty) = \mathbb{R}$$

This result, then coincides with what the ingenious winning strategy above told us as well: formula (1) is valid, because there is a winning strategy for Angel from every initial state. Except that the repeated $\varsigma_{\beta\cup\gamma}(\cdot)$ winning region construction seems more systematic than an ingenious guess of a smart winning strategy. So it gives a more constructive and explicit semantics.

Let's recap. In order to find the winning region of the hybrid game described in (1), it took us not just infinitely many steps, but more than that. After ω many iterations to arrive at $\varsigma_\alpha^\omega([0,1)) = [0,\infty)$, it took us one more step to arrive at

$$\varsigma_{(\beta\cup\gamma)^*}([0,1)) = \varsigma_\alpha^{\omega+1}([0,1)) = \mathbb{R}$$

where we denote the number of steps we took overall by $\omega+1$, since it was one more step than (first countable) infinitely many (i.e. ω many); see Fig. 1 for an illustration. More than infinitely many steps to get somewhere are plenty. Even worse: there are cases where even $\omega+1$ has not been enough of iteration to get to the repetition. The number of iterations needed to find $\varsigma_{\alpha^*}(X)$ could in general by much larger [Pla13].

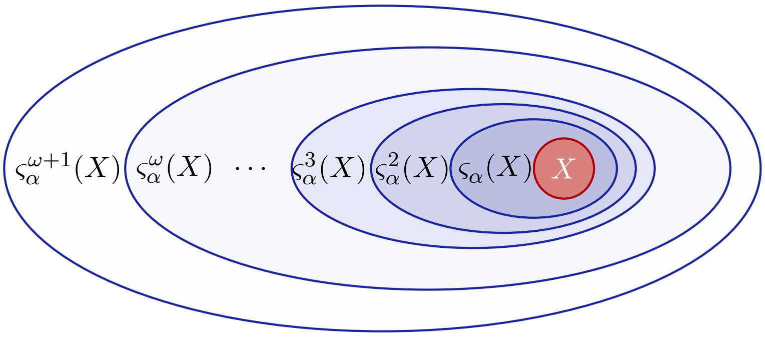

$\varsigma_\alpha^{\omega+1}(X)$
$\varsigma_\alpha^{\omega}(X)$
$\cdots$
$\varsigma_\alpha^{3}(X)$
$\varsigma_\alpha^{2}(X)$
$\varsigma_\alpha(X)$
X

Figure 1: Iteration $\varsigma_\alpha^{\omega+1}(X)$ of $\varsigma_\alpha(\cdot)$ from winning condition $X = [0,1)$ stops when applying $\varsigma_\alpha(\cdot)$ to the ωth infinite iteration $\varsigma_\alpha^\omega(X)$.

The existence of the above winning strategy is only found at the level $\varsigma_\alpha^{\omega+1}([0,1)) = \varsigma_\alpha([0,\infty)) = \mathbb{R}$. Even though any particular use of the winning strategy in any game play uses only some finite number of repetitions of the loop, the argument why it will always work requires $> \omega$ many iterations of $\varsigma_\alpha(\cdot)$, because Demon can change x to an arbitrarily big value, so that ω many iterations of $\varsigma_\alpha(\cdot)$ are needed to conclude that Angel has a winning strategy for any positive value of x. There is no smaller upper bound on the number of iterations it takes Angel to win, in particular Angel cannot promise ω as a bound on the repetition count, which is what the ω-semantics would effectively require her to do. But strategies do converge after $\omega + 1$ iterations.

Note 2. *The ω-semantics is inappropriate, because it can be arbitrarily far away from characterizing the winning region of hybrid games.*

3 Characterizing Winning Repetitions

Is there a more immediate way of characterizing the winning region $\varsigma_{\alpha^*}(X)$ of repetition?

Whenever a set Y is in the winning region $\varsigma_{\alpha^*}(X)$ of repetition, then $\varsigma_\alpha(Y)$ also should be in the winning region $\varsigma_{\alpha^*}(X)$, because it is just one step away from Y and α^* could simply repeat once more. Thus,

$$Y \subseteq \varsigma_{\alpha^*}(X) \;\;\Rightarrow\;\; \varsigma_\alpha(Y) \subseteq \varsigma_{\alpha^*}(X)$$

In particular, the set $Y \overset{\text{def}}{=} \varsigma_{\alpha^*}(X)$ itself is expected to satisfy

$$\varsigma_\alpha(\varsigma_{\alpha^*}(X)) \subseteq \varsigma_{\alpha^*}(X) \tag{2}$$

because repeating α once more from the winning region $\varsigma_{\alpha^*}(X)$ of repetition of α should not give us any states that did not already have a winning strategy in α^*. Consequently, a set $Z \subseteq \mathcal{S}$ only qualifies as a candidate for being the winning region $\varsigma_{\alpha^*}(X)$ of repetition if

$$\varsigma_\alpha(Z) \subseteq Z \tag{3}$$

That is, strategyzing along α from Z does not give anything that Z would not already know about.

So what is this set Z? Is there only one choice? Or multiple? If there are multiple choices, which Z is it? Does such a Z always exist, even?

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

One such Z always exist, even though it may be rather boring. The empty set $Z \overset{\text{def}}{=} \emptyset$ certainly satisfies $\varsigma_\alpha(\emptyset) = \emptyset$, because it is rather hard to win a game that requires Angel to enter the empty set of states to win.

But the empty set is maybe a bit small. The winning region $\varsigma_{\alpha^*}(X)$ of repetition of α should at least contain the winning condition X, because the winning condition X is particularly easy to reach from states in X that have already let Angel won by simply suggesting Angel to repeat zero times. Consequently, the only Z that qualify as a candidate for being $\varsigma_{\alpha^*}(X)$ should satisfy (3) and

$$X \subseteq Z \tag{4}$$

Both conditions (3) and (4) together can be summarized in a single condition as follows:

Note 3 (Prefixpoint). *Every candidate Z for the winning region $\varsigma_{\alpha^*}(X)$ satisfies:*

$$X \cup \varsigma_\alpha(Z) \subseteq Z \tag{5}$$

Again: what is this set Z that satisfies (5)? Is there only one choice? Or multiple? If there are multiple choices, which Z is it? Does such a Z always exist, even?

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

One such Z certainly exists. The empty set does not qualify unless $X = \emptyset$. The set X itself is too small unless the game has no incentive to start repeating, because $\varsigma_\alpha(X) \subseteq X$. But the full space $Z \overset{\text{def}}{=} \mathcal{S}$ always satisfies (5) trivially. Now, the whole space is a little big to call it Angel's winning region independently of the hybrid game α. Even if the full space may very well be the winning region for some particularly Demonophobic Angel-friendly hybrid games like (1), it is hardly the right winning region for any arbitrary α^*. For example for Demon's favorite game where he always wins, $\varsigma_{\alpha^*}(X)$ had better be $\emptyset$, not $\mathcal{S}$. Thus, the largest solution Z of (5) hardly qualifies.

So which solution Z of (5) should be the definition of $\varsigma_{\alpha^*}(X)$ now?

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

Among the many Z that solve (5), the largest one is not informative, because the largest Z simply degrades to $\mathcal{S}$. So smaller solutions Z are preferable. How do multiple solutions relate at all? Suppose Y, Z are both solutions of (5). That is

$$X \cup \varsigma_\alpha(Y) \subseteq Y \tag{6}$$
$$X \cup \varsigma_\alpha(Z) \subseteq Z \tag{7}$$

Then, by monotonicity lemma, Lemma 3:

$$X \cup \varsigma_\alpha(Y \cap Z) \overset{\text{mon}}{\subseteq} X \cup \varsigma_\alpha(Y) \cap \varsigma_\alpha(Z) \overset{(6),(7)}{\subseteq} Y \cap Z \tag{8}$$

Hence, by (8), the intersection $Y \cap Z$ of solutions Y and Z of (5) also is a solution of (5):

Lemma 1 (Intersection closure)**.** *Whenever there are two solutions Y, Z of* (5)*, a (possibly) smaller solution of* (5) *can be obtained by intersection $Y \cap Z$.*

So whenever there are two solutions Z_1, Z_2 of (5), their intersection $Y_1 \cap Z_2$ solves (5). When there's yet another solution Z_3 of (5), their intersection $Y_1 \cap Y_2 \cap Y_3$ also solves (5). Similarly for *any* larger family of solutions. If we keep on intersecting solutions, we will arrive at smaller solutions until, some fine day, there's not going to be a smaller one. This yields the smallest solution Z of (5) which can be characterized directly.

Among the many Z that solve (5), the smallest Z that solves (5) is informative and can be used to define $\varsigma_{\alpha^*}(X)$:

$$\varsigma_{\alpha^*}(X) = \bigcap \{Z \subseteq \mathcal{S} \,:\, X \cup \varsigma_\alpha(Z) \subseteq Z\} \tag{9}$$

The set on the right-hand side of (9) is an intersection of solutions, thus, a solution by Lemma 1 (or its counterpart for families of solutions). Hence $\varsigma_{\alpha^*}(X)$ itself satisfies (5):

$$X \cup \varsigma_\alpha(\varsigma_{\alpha^*}(X)) \subseteq \varsigma_{\alpha^*}(X) \tag{10}$$

Also compare this with what we argued earlier in (2). Could it be the case that the inclusion in (10) is strict, i.e. not equals? No this cannot happen, because $\varsigma_{\alpha^*}(X)$ is the smallest. In detail, by (10), the set $Z \overset{\text{def}}{=} X \cup \varsigma_\alpha(\varsigma_{\alpha^*}(X))$ satisfies $Z \subseteq \varsigma_{\alpha^*}(X)$ and, thus, by Lemma 3:

$$X \cup \varsigma_\alpha(Z) \overset{\text{mon}}{\subseteq} X \cup \varsigma_\alpha(\varsigma_{\alpha^*}(X)) = Z$$

Consequently, both inclusions hold, so $\varsigma_{\alpha^*}(X)$ satisfies

$$X \cup \varsigma_\alpha(\varsigma_{\alpha^*}(X)) = \varsigma_{\alpha^*}(X) \tag{11}$$

That is, $\varsigma_{\alpha^*}(X)$ is even a *fixpoint* solving the equation

$$X \cup \varsigma_\alpha(Z) = Z \tag{12}$$

and it is the *least fixpoint*, i.e. the smallest Z solving the equation (12).

The fact that $\varsigma_{\alpha^*}(X)$ is defined as the least of the fixpoints makes sure that Angel only wins games by a well-founded number of repetitions. That is, she only wins a repetition if she ultimately stops repeating, not by postponing termination forever. See [Pla13] for more details.

It is also worth noting that it would still have been possible to make the iteration of winning region constructions work out using the seminal fixpoint theorem of Knaster-Tarski. Yet, this requires the iterated winning region constructions to go significantly transfinite [Pla13] way beyond ω.

4 Semantics of Hybrid Games

The semantics of differential game logic from Lecture 21 was still pending a definition of the winning regions $\varsigma_\alpha(\cdot)$ and $\delta_\alpha(\cdot)$ for Angel and Demon, respectively, in the hybrid game α. Rather than taking a detour for understanding those by operational game semantics (as in Lecture 20), the winning regions of hybrid games can be defined directly, giving a denotational semantics to hybrid games.

The only difference compared to the definition in Lecture 21 is the new case of repetition α^*.

Definition 2 (Semantics of hybrid games)**.** The *semantics of a hybrid game* α is a function $\varsigma_\alpha(\cdot)$ that, for each interpretation I and each set of Angel's winning states $X \subseteq \mathcal{S}$, gives the *winning region*, i.e. the set of states $\varsigma_\alpha(X)$ from which Angel has a winning strategy to achieve X (whatever strategy Demon chooses). It is defined inductively as follows[a]

1. $\varsigma_{x:=\theta}(X) = \{\nu \in \mathcal{S} : \nu_x^{[\![\theta]\!]_\nu} \in X\}$
2. $\varsigma_{x'=\theta \,\&\, H}(X) = \{\varphi(0) \in \mathcal{S} : \varphi(r) \in X$ for some $r \in \mathbb{R}_{\geq 0}$ and (differentiable) $\varphi : [0,r] \to \mathcal{S}$ such that $\varphi(\zeta) \in [\![H]\!]^I$ and $\frac{\mathrm{d}\,\varphi(t)(x)}{\mathrm{d}t}(\zeta) = [\![\theta]\!]_{\varphi(\zeta)}$ for all $0 \leq \zeta \leq r\}$
3. $\varsigma_{?H}(X) = [\![H]\!]^I \cap X$
4. $\varsigma_{\alpha\cup\beta}(X) = \varsigma_\alpha(X) \cup \varsigma_\beta(X)$
5. $\varsigma_{\alpha;\beta}(X) = \varsigma_\alpha(\varsigma_\beta(X))$
6. $\varsigma_{\alpha^*}(X) = \bigcap\{Z \subseteq \mathcal{S} : X \cup \varsigma_\alpha(Z) \subseteq Z\}$
7. $\varsigma_{\alpha^d}(X) = (\varsigma_\alpha(X^\complement))^\complement$

The *winning region* of Demon, i.e. the set of states $\delta_\alpha(X)$ from which Demon has a winning strategy to achieve X (whatever strategy Angel chooses) is defined inductively as follows

1. $\delta_{x:=\theta}(X) = \{\nu \in \mathcal{S} : \nu_x^{[\![\theta]\!]_\nu} \in X\}$
2. $\delta_{x'=\theta \,\&\, H}(X) = \{\varphi(0) \in \mathcal{S} : \varphi(r) \in X$ for all $r \in \mathbb{R}_{\geq 0}$ and (differentiable) $\varphi : [0,r] \to \mathcal{S}$ such that $\varphi(\zeta) \in [\![H]\!]^I$ and $\frac{\mathrm{d}\,\varphi(t)(x)}{\mathrm{d}t}(\zeta) = [\![\theta]\!]_{\varphi(\zeta)}$ for all $0 \leq \zeta \leq r\}$
3. $\delta_{?H}(X) = ([\![H]\!]^I)^\complement \cup X$
4. $\delta_{\alpha\cup\beta}(X) = \delta_\alpha(X) \cap \delta_\beta(X)$
5. $\delta_{\alpha;\beta}(X) = \delta_\alpha(\delta_\beta(X))$
6. $\delta_{\alpha^*}(X) = \bigcup\{Z \subseteq \mathcal{S} : Z \subseteq X \cap \delta_\alpha(Z)\}$
7. $\delta_{\alpha^d}(X) = (\delta_\alpha(X^\complement))^\complement$

[a] The semantics of a hybrid game is not merely a reachability relation between states as for hybrid systems [Pla12], because the adversarial dynamic interactions and nested choices of the players have to be taken into account.

This notation uses $\varsigma_\alpha(X)$ instead of $\varsigma_\alpha^I(X)$ and $\delta_\alpha(X)$ instead of $\delta_\alpha^I(X)$, because the interpretation I that gives a semantics to predicate symbols in tests and evolution domains is clear from the context. Strategies do not occur explicitly in the d$\mathcal{GL}$ semantics, because it is based on the existence of winning strategies, not on the strategies themselves.

Just as the semantics d$\mathcal{L}$, the semantics of dG$\mathcal{L}$ is *compositional*, i.e. the semantics of a compound dG$\mathcal{L}$ formula is a simple function of the semantics of its pieces, and the semantics of a compound hybrid game is a function of the semantics of its pieces. Furthermore, existence of a strategy in hybrid game α to achieve X is independent of any game and dG$\mathcal{L}$ formula surrounding α, but just depends on the remaining game α itself and the goal X. By a simple inductive argument, this shows that one can focus on memoryless strategies, because the existence of strategies does not depend on the context, hence, by working bottom up, the strategy itself cannot depend on past states and choices, only the current state, remaining game, and goal. This also follows from a generalization of a classical result by Zermelo. Furthermore, the semantics is monotone, i.e. larger sets of winning states induce larger winning regions.

Monotonicity is what Lecture 21 looked into for the case of hybrid games without repetition. But it continues to hold for general hybrid games.

Lemma 3 (Monotonicity [Pla13])**.** *The semantics is* monotone, *i.e.* $\varsigma_\alpha(X) \subseteq \varsigma_\alpha(Y)$ *and* $\delta_\alpha(X) \subseteq \delta_\alpha(Y)$ *for all* $X \subseteq Y$*.*

Proof. A simple check based on the observation that X only occurs with an even number of negations in the semantics. For example, $\varsigma_{\alpha^*}(X) = \bigcap\{Z \subseteq \mathcal{S} : X \cup \varsigma_\alpha(Z) \subseteq Z\} \subseteq \bigcap\{Z \subseteq \mathcal{S} : Y \cup \varsigma_\alpha(Z) \subseteq Z\} = \varsigma_{\alpha^*}(Y)$ if $X \subseteq Y$. Likewise, $X \subseteq Y$ implies $X^\complement \supseteq Y^\complement$, hence $\varsigma_\alpha(X^\complement) \supseteq \varsigma_\alpha(Y^\complement)$, so $\varsigma_{\alpha^d}(X) = (\varsigma_\alpha(X^\complement))^\complement \subseteq (\varsigma_\alpha(Y^\complement))^\complement = \varsigma_{\alpha^d}(Y)$. □

Monotonicity implies that the least fixpoint in $\varsigma_{\alpha^*}(X)$ and the greatest fixpoint in $\delta_{\alpha^*}(X)$ are well-defined [HKT00, Lemma 1.7]. The semantics of $\varsigma_{\alpha^*}(X)$ is a least fixpoint, which results in a well-founded repetition of α, i.e. Angel can repeat any number of times but she ultimately needs to stop at a state in X in order to win. The semantics of $\delta_{\alpha^*}(X)$ is a greatest fixpoint, instead, for which Demon needs to achieve a state in X after every number of repetitions, because Angel could choose to stop at any time, but Demon still wins if he only postpones $X^\complement$ forever, because Angel ultimately has to stop repeating. Thus, for the formula $\langle\alpha^*\rangle\phi$, Demon already has a winning strategy if he only has a strategy that is not losing by preventing ϕ indefinitely, because Angel eventually has to stop repeating anyhow and will then end up in a state not satisfying ϕ, which makes her lose. The situation for $[\alpha^*]\phi$ is dual.

5 Hybrid Game Axioms

An axiomatization for differential game logic has been found in previous work [Pla13], where we refer to for more details. The study of proof rules for differential game logic will be deferred to next lecture. But its axioms can be discussed today.

Note 7 (Differential game logic axioms [Pla13]).

$$([\cdot])\ [\alpha]\phi \leftrightarrow \neg\langle\alpha\rangle\neg\phi$$

$$(\langle{:=}\rangle)\ \langle x := \theta\rangle\phi(x) \leftrightarrow \phi(\theta)$$

$$(\langle{'}\rangle)\ \langle x' = \theta\rangle\phi \leftrightarrow \exists t{\geq}0\, \langle x := y(t)\rangle\phi \qquad (y'(t) = \theta)$$

$$(\langle ? \rangle)\ \langle ?H\rangle\phi \leftrightarrow (H \land \phi)$$

$$(\langle\cup\rangle)\ \langle\alpha \cup \beta\rangle\phi \leftrightarrow \langle\alpha\rangle\phi \lor \langle\beta\rangle\phi$$

$$(\langle;\rangle)\ \langle\alpha;\beta\rangle\phi \leftrightarrow \langle\alpha\rangle\langle\beta\rangle\phi$$

$$(\langle^*\rangle)\ \phi \lor \langle\alpha\rangle\langle\alpha^*\rangle\phi \rightarrow \langle\alpha^*\rangle\phi$$

$$(\langle^d\rangle)\ \langle\alpha^d\rangle\phi \leftrightarrow \neg\langle\alpha\rangle\neg\phi$$

6 Determinacy

Every particular game play in a hybrid game is won by exactly one player, because hybrid games are zero-sum and there are no draws. Hybrid games actually satisfy a much stronger property: *determinacy*, i.e. that, from any initial situation, either one of the players always has a winning strategy to force a win, regardless of how the other player chooses to play.

If, from the same initial state, both Angel and Demon had a winning strategy for opposing winning conditions, then something would be terribly inconsistent. It cannot happen that Angel has a winning strategy in hybrid game α to get to a state where $\neg\phi$ and, from the same initial state, Demon supposedly also has a winning strategy in the same hybrid game α to get to a state where ϕ holds. After all, a winning strategy is a strategy that makes that player win no matter what strategy the opponent follows. Hence, for any initial state, at most one player can have a winning strategy for complementary winning conditions. This argues for the validity of $\vDash \neg([\alpha]\phi \land \langle\alpha\rangle\neg\phi)$, which can also be proved (Theorem 4).

So it cannot happen that both players have a winning strategy for complementary winning conditions. But it might still happen that no one has a winning strategy, i.e. both players can let the other player win, but cannot win strategically themselves (recall, e.g., the filibuster example from Lecture 20, which first appeared as if no player might have a winning strategy but then turned out to make Demon win). This does not happen for hybrid games, though, because at least one (hence exactly one) player has a winning strategy for complementary winning conditions from any initial state.

Theorem 4 (Consistency & determinacy [Pla13]). *Hybrid games are consistent and determined, i.e.* $\vDash \neg\langle\alpha\rangle\neg\phi \leftrightarrow [\alpha]\phi$.

Proof. The proof shows by induction on the structure of α that $\varsigma_\alpha(X^\complement)^\complement = \delta_\alpha(X)$ for all $X \subseteq \mathcal{S}$ and all I with some set of states $\mathcal{S}$, which implies the validity of $\neg\langle\alpha\rangle\neg\phi \leftrightarrow [\alpha]\phi$ using $X \overset{\text{def}}{=} [\![\phi]\!]^I$.

1. $\varsigma_{x:=\theta}(X^\complement)^\complement = \{\nu \in \mathcal{S} : \nu_x^{[\![\theta]\!]_\nu} \notin X\}^\complement = \varsigma_{x:=\theta}(X) = \delta_{x:=\theta}(X)$
2. $\varsigma_{x'=\theta\,\&\,H}(X^\complement)^\complement = \{\varphi(0) \in \mathcal{S} : \varphi(r) \notin X$ for some $0 \leq r \in \mathbb{R}$ and some (differentiable) $\varphi : [0,r] \to \mathcal{S}$ such that $\frac{\mathrm{d}\,\varphi(t)(x)}{\mathrm{d}t}(\zeta) = [\![\theta]\!]_{\varphi(\zeta)}$ and $\varphi(\zeta) \in [\![H]\!]^I$ for all $0 \leq \zeta \leq r\}^\complement = \delta_{x'=\theta\,\&\,H}(X)$, because the set of states from which there is no winning strategy for Angel to reach a state in $X^\complement$ prior to leaving $[\![H]\!]^I$ along $x' = \theta\,\&\,H$ is exactly the set of states from which $x' = \theta\,\&\,H$ always stays in X (until leaving $[\![H]\!]^I$ in case that ever happens).
3. $\varsigma_{?H}(X^\complement)^\complement = ([\![H]\!]^I \cap X^\complement)^\complement = ([\![H]\!]^I)^\complement \cup (X^\complement)^\complement = \delta_{?H}(X)$
4. $\varsigma_{\alpha\cup\beta}(X^\complement)^\complement = (\varsigma_\alpha(X^\complement) \cup \varsigma_\beta(X^\complement))^\complement = \varsigma_\alpha(X^\complement)^\complement \cap \varsigma_\beta(X^\complement)^\complement = \delta_\alpha(X) \cap \delta_\beta(X) = \delta_{\alpha\cup\beta}(X)$
5. $\varsigma_{\alpha;\beta}(X^\complement)^\complement = \varsigma_\alpha(\varsigma_\beta(X^\complement))^\complement = \varsigma_\alpha(\delta_\beta(X)^\complement)^\complement = \delta_\alpha(\delta_\beta(X)) = \delta_{\alpha;\beta}(X)$
6. $\varsigma_{\alpha^*}(X^\complement)^\complement = \Big(\bigcap\{Z \subseteq \mathcal{S} : X^\complement \cup \varsigma_\alpha(Z) \subseteq Z\}\Big)^\complement = \Big(\bigcap\{Z \subseteq \mathcal{S} : (X \cap \varsigma_\alpha(Z)^\complement)^\complement \subseteq Z\}\Big)^\complement$
   $= \Big(\bigcap\{Z \subseteq \mathcal{S} : (X \cap \delta_\alpha(Z^\complement))^\complement \subseteq Z\}\Big)^\complement = \bigcup\{Z \subseteq \mathcal{S} : Z \subseteq X \cap \delta_\alpha(Z)\} = \delta_{\alpha^*}(X).$ [1]
7. $\varsigma_{\alpha^d}(X^\complement)^\complement = (\varsigma_\alpha((X^\complement)^\complement)^\complement)^\complement = \delta_\alpha(X^\complement)^\complement = \delta_{\alpha^d}(X)$ □

Exercises

Exercise 1. Explain how often you will have to repeat the winning region construction to show that the following dG$\mathcal{L}$ formula is valid:

$$\langle (x := x+1; x' = 1^d \cup x := x-1)^* \rangle\, (0 \leq x < 1)$$

Exercise 2. Can you find dG$\mathcal{L}$ formulas for which the winning region construction takes even longer to terminate? How far can you push this?

Exercise 3. Carefully identify how determinacy relates to the two possible understandings of the filibuster example discussed in an earlier lecture.

[1] The penultimate equation follows from the μ-calculus equivalence $\nu Z.\Upsilon(Z) \equiv \neg\mu Z.\neg\Upsilon(\neg Z)$ and the fact that least pre-fixpoints are fixpoints and that greatest post-fixpoints are fixpoints for monotone functions.

Exercise 4. Prove the elided cases of Lemma 3.

Exercise 5. Find the appropriate soundness notion for the axioms of $\mathsf{dG}\mathcal{L}$ and prove that the axioms are sound.

Exercise 6. Write down a valid formula that characterizes an interesting game between two robots.

References

[HKT00] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic logic*. MIT Press, 2000.

[Pla12] André Platzer. The complete proof theory of hybrid systems. In *LICS*, pages 541–550. IEEE, 2012. doi:10.1109/LICS.2012.64.

[Pla13] André Platzer. A complete axiomatization of differential game logic for hybrid games. Technical Report CMU-CS-13-100R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January, Revised and extended in July 2013.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Game Proofs & Separations

André Platzer

Carnegie Mellon University
Lecture 23

1 Introduction

This lecture continues the study of hybrid games and their logic, differential game logic [Pla13]. Lecture 20 on Hybrid Systems & Games introduced hybrid games, Lecture 21 on Winning Strategies & Regions studied the winning region semantics, and Lecture 22 on Winning & Proving Hybrid Games identified the winning region semantics for loops in hybrid games as well as a study of the axioms of hybrid games.

These lecture notes are based on [Pla13], where more information can be found on logic and hybrid games.

2 Recall: Semantics of Hybrid Games

Recall the semantics of hybrid games and two results from Lecture 22 on Winning & Proving Hybrid Games.

Definition 1 (Semantics of hybrid games). The *semantics of a hybrid game* α is a function $\varsigma_\alpha(\cdot)$ that, for each interpretation I and each set of Angel's winning states $X \subseteq \mathcal{S}$, gives the *winning region*, i.e. the set of states $\varsigma_\alpha(X)$ from which Angel has a winning strategy to achieve X (whatever strategy Demon chooses). It is defined inductively as follows[a]

1. $\varsigma_{x:=\theta}(X) = \{\nu \in \mathcal{S} : \nu_x^{[\![\theta]\!]_\nu} \in X\}$
2. $\varsigma_{x'=\theta \,\&\, H}(X) = \{\varphi(0) \in \mathcal{S} : \varphi(r) \in X$ for some $r \in \mathbb{R}_{\geq 0}$ and (differentiable) $\varphi : [0, r] \to \mathcal{S}$ such that $\varphi(\zeta) \in [\![H]\!]^I$ and $\frac{\mathrm{d}\,\varphi(t)(x)}{\mathrm{d}t}(\zeta) = [\![\theta]\!]_{\varphi(\zeta)}$ for all $0 \leq \zeta \leq r\}$
3. $\varsigma_{?H}(X) = [\![H]\!]^I \cap X$
4. $\varsigma_{\alpha\cup\beta}(X) = \varsigma_\alpha(X) \cup \varsigma_\beta(X)$
5. $\varsigma_{\alpha;\beta}(X) = \varsigma_\alpha(\varsigma_\beta(X))$
6. $\varsigma_{\alpha^*}(X) = \bigcap\{Z \subseteq \mathcal{S} : X \cup \varsigma_\alpha(Z) \subseteq Z\}$
7. $\varsigma_{\alpha^d}(X) = (\varsigma_\alpha(X^\complement))^\complement$

The *winning region* of Demon, i.e. the set of states $\delta_\alpha(X)$ from which Demon has a winning strategy to achieve X (whatever strategy Angel chooses) is defined inductively as follows

1. $\delta_{x:=\theta}(X) = \{\nu \in \mathcal{S} : \nu_x^{[\![\theta]\!]_\nu} \in X\}$
2. $\delta_{x'=\theta \,\&\, H}(X) = \{\varphi(0) \in \mathcal{S} : \varphi(r) \in X$ for all $r \in \mathbb{R}_{\geq 0}$ and (differentiable) $\varphi : [0, r] \to \mathcal{S}$ such that $\varphi(\zeta) \in [\![H]\!]^I$ and $\frac{\mathrm{d}\,\varphi(t)(x)}{\mathrm{d}t}(\zeta) = [\![\theta]\!]_{\varphi(\zeta)}$ for all $0 \leq \zeta \leq r\}$
3. $\delta_{?H}(X) = ([\![H]\!]^I)^\complement \cup X$
4. $\delta_{\alpha\cup\beta}(X) = \delta_\alpha(X) \cap \delta_\beta(X)$
5. $\delta_{\alpha;\beta}(X) = \delta_\alpha(\delta_\beta(X))$
6. $\delta_{\alpha^*}(X) = \bigcup\{Z \subseteq \mathcal{S} : Z \subseteq X \cap \delta_\alpha(Z)\}$
7. $\delta_{\alpha^d}(X) = (\delta_\alpha(X^\complement))^\complement$

[a] The semantics of a hybrid game is not merely a reachability relation between states as for hybrid systems [Pla12], because the adversarial dynamic interactions and nested choices of the players have to be taken into account.

Lemma 2 (Monotonicity [Pla13]). *The semantics is* monotone, *i.e.* $\varsigma_\alpha(X) \subseteq \varsigma_\alpha(Y)$ *and* $\delta_\alpha(X) \subseteq \delta_\alpha(Y)$ *for all* $X \subseteq Y$.

Theorem 3 (Consistency & determinacy [Pla13]). *Hybrid games are consistent and determined, i.e.* $\vDash \neg\langle\alpha\rangle\neg\phi \leftrightarrow [\alpha]\phi$.

3 Hybrid Game Proofs

An axiomatization for differential game logic has been found in previous work [Pla13], where we refer to for more details.

Note 4 (Differential game logic axiomatization [Pla13]).

$$([\cdot])\quad [\alpha]\phi \leftrightarrow \neg\langle\alpha\rangle\neg\phi$$

$$(\langle{:=}\rangle)\quad \langle x := \theta\rangle\phi(x) \leftrightarrow \phi(\theta)$$

$$(\langle{}'\rangle)\quad \langle x' = \theta\rangle\phi \leftrightarrow \exists t{\geq}0\, \langle x := y(t)\rangle\phi \qquad (y'(t) = \theta)$$

$$(\langle?\rangle)\quad \langle ?H\rangle\phi \leftrightarrow (H \land \phi)$$

$$(\langle\cup\rangle)\quad \langle\alpha \cup \beta\rangle\phi \leftrightarrow \langle\alpha\rangle\phi \lor \langle\beta\rangle\phi$$

$$(\langle;\rangle)\quad \langle\alpha;\beta\rangle\phi \leftrightarrow \langle\alpha\rangle\langle\beta\rangle\phi$$

$$(\langle^*\rangle)\quad \phi \lor \langle\alpha\rangle\langle\alpha^*\rangle\phi \rightarrow \langle\alpha^*\rangle\phi$$

$$(\langle^d\rangle)\quad \langle\alpha^d\rangle\phi \leftrightarrow \neg\langle\alpha\rangle\neg\phi$$

$$(\text{M})\quad \frac{\phi \rightarrow \psi}{\langle\alpha\rangle\phi \rightarrow \langle\alpha\rangle\psi}$$

$$(\text{FP})\quad \frac{\phi \lor \langle\alpha\rangle\psi \rightarrow \psi}{\langle\alpha^*\rangle\phi \rightarrow \psi}$$

$$(\text{ind})\quad \frac{\phi \rightarrow [\alpha]\phi}{\phi \rightarrow [\alpha^*]\phi}$$

The proof rules FP and ind are equivalent in the sense that one can be derived from the other in the dGL calculus [Pla13].

Example 4. The dual filibuster game formula from Lecture 20 proves easily in the dGL calculus by going back and forth between players [Pla13]:

$$
\begin{array}{rl}
 & \qquad * \\ \cline{2-2}
\mathbb{R} & x=0 \to 0=0 \lor 1=0 \\ \cline{2-2}
\langle:=\rangle & x=0 \to \langle x:=0\rangle x=0 \lor \langle x:=1\rangle x=0 \\ \cline{2-2}
\langle\cup\rangle & x=0 \to \langle x:=0 \cup x:=1\rangle x=0 \\ \cline{2-2}
\langle^d\rangle & x=0 \to \neg\langle x:=0 \cap x:=1\rangle \neg x=0 \\ \cline{2-2}
[\cdot] & x=0 \to [x:=0 \cap x:=1]x=0 \\ \cline{2-2}
\text{ind} & x=0 \to [(x:=0 \cap x:=1)^*]x=0 \\ \cline{2-2}
\langle^d\rangle & x=0 \to \langle (x:=0 \cup x:=1)^\times\rangle x=0
\end{array}
$$

4 Soundness

Theorem 5 (Soundness [Pla13])**.** *The* $\mathsf{d}\mathcal{GL}$ *proof calculus in Fig. 4 is sound, i.e. all provable formulas are valid.*

Proof. The full proof can be found in [Pla13]. We just consider a few cases to exemplify the fundamentally more general semantics of hybrid games arguments compared to hybrid systems arguments. To prove soundness of an equivalence axiom $\phi \leftrightarrow \psi$, show $[\![\phi]\!]^I = [\![\psi]\!]^I$ for all interpretations I with any set of states $\mathcal{S}$.

$\langle\cup\rangle$ $[\![\langle\alpha\cup\beta\rangle\phi]\!]^I = \varsigma_{\alpha\cup\beta}([\![\phi]\!]^I) = \varsigma_\alpha([\![\phi]\!]^I) \cup \varsigma_\beta([\![\phi]\!]^I) = [\![\langle\alpha\rangle\phi]\!]^I \cup [\![\langle\beta\rangle\phi]\!]^I = [\![\langle\alpha\rangle\phi \lor \langle\beta\rangle\phi]\!]^I$

$\langle;\rangle$ $[\![\langle\alpha;\beta\rangle\phi]\!]^I = \varsigma_{\alpha;\beta}([\![\phi]\!]^I) = \varsigma_\alpha(\varsigma_\beta([\![\phi]\!]^I)) = \varsigma_\alpha([\![\langle\beta\rangle\phi]\!]^I) = [\![\langle\alpha\rangle\langle\beta\rangle\phi]\!]^I$.

$\langle?\rangle$ $[\![\langle ?H\rangle\phi]\!]^I = \varsigma_{?H}([\![\phi]\!]^I) = [\![H]\!]^I \cap [\![\phi]\!]^I = [\![H \land \phi]\!]^I$

$[\cdot]$ is sound by Theorem 3.

M Assume the premise $\phi \to \psi$ is valid in interpretation I, i.e. $[\![\phi]\!]^I \subseteq [\![\psi]\!]^I$. Then the conclusion $\langle\alpha\rangle\phi \to \langle\alpha\rangle\psi$ is valid in I, i.e. $[\![\langle\alpha\rangle\phi]\!]^I = \varsigma_\alpha([\![\phi]\!]^I) \subseteq \varsigma_\alpha([\![\psi]\!]^I) = [\![\langle\alpha\rangle\psi]\!]^I$ by monotonicity (Lemma 2). □

5 Separating Axioms

The axioms of differential game logic in Fig. 4 are sound for hybrid systems as well, because every hybrid system is a (single player) hybrid game. With a few exceptions, they look surprisingly close to the axioms for hybrid systems from Lecture 5. In order to understand the fundamental difference between hybrid systems and hybrid games, it is instructive to also investigate separating axioms, i.e. axioms of hybrid systems that are not sound for hybrid games. Some of these are summarized in Fig. 1, referring to [Pla13] for details.

$$
\begin{array}{llll}
\cancel{\mathrm{K}} & [\alpha](\phi \to \psi) \to ([\alpha]\phi \to [\alpha]\psi) & \mathrm{M} & \langle\alpha\rangle\phi \lor \langle\alpha\rangle\psi \to \langle\alpha\rangle(\phi \lor \psi) \\
\cancel{\mathrm{G}} & \dfrac{\phi}{[\alpha]\phi} & \mathrm{M}_{[\cdot]} & \dfrac{\phi \to \psi}{[\beta]\phi \to [\beta]\psi} \\
\cancel{\mathrm{R}} & \dfrac{\phi_1 \land \phi_2 \to \psi}{[\alpha]\phi_1 \land [\alpha]\phi_2 \to [\alpha]\psi} & & \\
\cancel{\mathrm{B}} & \langle\alpha\rangle\exists x\,\phi \to \exists x\,\langle\alpha\rangle\phi \qquad (x \notin \alpha) & \overleftarrow{\mathrm{B}} & \exists x\,\langle\alpha\rangle\phi \to \langle\alpha\rangle\exists x\,\phi \qquad (x \notin \alpha) \\
\cancel{\mathrm{I}} & [\alpha^*](\phi \to [\alpha]\phi) \to (\phi \to [\alpha^*]\phi) & & \\
\cancel{\mathrm{FA}} & \langle\alpha^*\rangle\phi \to \phi \lor \langle\alpha^*\rangle(\neg\phi \land \langle\alpha\rangle\phi) & &
\end{array}
$$

Figure 1: Separating axioms: The axioms and rules on the left are sound for hybrid systems but not for hybrid games. The related axioms on the right are sound for hybrid games.

6 Repetitive Diamonds – Convergence vs. Iteration

More fundamental differences between hybrid systems and hybrid games also exist in terms of convergence rules, even if these have played a less prominent role in this course so far. These differences are discussed in detail elsewhere [Pla13]. In a nutshell, Harel's convergence rule [HMP77] is not a separating axiom, because it is sound for dG$\mathcal{L}$, just unnecessary, and, furthermore, not even particularly useful for hybrid games [Pla13]. The hybrid version of Harel's convergence rule [Pla08] for d$\mathcal{L}$ reads as follows (it assumes that v does not occur in α):

$$
(\mathrm{con})\ \frac{\varphi(v+1) \land v+1>0 \vdash \langle\alpha\rangle\varphi(v)}{\Gamma, \exists v\,\varphi(v) \vdash \langle\alpha^*\rangle\exists v{\leq}0\,\varphi(v), \Delta}
$$

The d$\mathcal{L}$ proof rule con expresses that the variant $\varphi(v)$ holds for some real number $v \leq 0$ after repeating α sufficiently often if $\varphi(v)$ holds for some real number at all in the beginning (antecedent) and, by premise, $\varphi(v)$ can decrease after some execution of α by 1 (or another positive real constant) if $v > 0$. This rule can be used to show positive progress (by 1) with respect to $\varphi(v)$ by executing α. Just like the induction rule ind is often used with a separate premiss for the initial and postcondition check (*ind′* from Lecture 7 on Loops & Invariants), rule con is often used in the following derived form:

$$
(con')\ \frac{\Gamma \vdash \exists v\,\varphi(v), \Delta \quad \forall v{>}0\,(\varphi(v) \to \langle\alpha\rangle\varphi(v-1)) \quad \exists v{\leq}0\,\varphi(v) \vdash \psi}{\Gamma \vdash \langle\alpha^*\rangle\psi, \Delta}
$$

The following sequent proof shows how convergence rule *con′* can be used to prove a simple dℒ liveness property of a hybrid program:

$$\to\!r\,\dfrac{con'\,\dfrac{\mathbb{R}\,\dfrac{*}{x \geq 0 \vdash \exists n\, x < n+1} \qquad \langle:=\rangle\,\dfrac{\mathbb{R}\,\dfrac{*}{x < n+2 \land n+1 > 0 \vdash x-1 < n+1}}{x < n+2 \land n+1 > 0 \vdash \langle x := x-1\rangle x < n+1} \qquad \mathbb{R}\,\dfrac{*}{\exists n{\leq}0\, x < n+1 \vdash x < 1}}{x \geq 0 \vdash \langle (x := x-1)^*\rangle 0 \leq x < 1}}{x \geq 0 \to \langle (x := x-1)^*\rangle x < 1}$$

Let's compare how dGℒ proves diamond properties of repetitions based on the iteration axiom $\langle * \rangle$.

Example 6 (Non-game system). The simple non-game dGℒ formula

$$x \geq 0 \to \langle (x := x-1)^*\rangle 0 \leq x < 1$$

is provable, shown in Fig. 2, where $\langle \alpha^* \rangle 0 \leq x < 1$ is short for $\langle (x := x-1)^* \rangle (0 \leq x < 1)$.

$$\text{MP}\,\dfrac{\langle * \rangle,\forall\,\dfrac{\text{US}\,\dfrac{\langle:=\rangle\,\dfrac{\mathbb{R}\,\dfrac{*}{\forall x\, (0 \leq x < 1 \lor p(x-1) \to p(x)) \to (x \geq 0 \to p(x))}}{\forall x\, (0 \leq x < 1 \lor \langle x := x-1\rangle p(x) \to p(x)) \to (x \geq 0 \to p(x))}}{\forall x\, (0 \leq x < 1 \lor \langle x := x-1\rangle\langle \alpha^*\rangle 0 \leq x < 1 \to \langle \alpha^*\rangle 0 \leq x < 1) \to (x \geq 0 \to \langle \alpha^*\rangle 0 \leq x < 1)}}{\forall x\, (0 \leq x < 1 \lor \langle x := x-1\rangle\langle \alpha^*\rangle 0 \leq x < 1 \to \langle \alpha^*\rangle 0 \leq x < 1)}}{x \geq 0 \to \langle \alpha^*\rangle 0 \leq x < 1}$$

Figure 2: dGℒ Angel proof for non-game system Example 6
$x \geq 0 \to \langle (x := x-1)^*\rangle 0 \leq x < 1$

Example 7 (Choice game). The dGℒ formula

$$x = 1 \land a = 1 \to \langle (x := a; a := 0 \cap x := 0)^*\rangle x \neq 1$$

is provable as shown in Fig. 3, where $\beta \cap \gamma$ is short for $x := a; a := 0 \cap x := 0$ and $\langle (\beta \cap \gamma)^* \rangle x \neq 1$ short for $\langle (x := a; a := 0 \cap x := 0)^* \rangle x \neq 1$:

Example 8 (2-Nim-type game). The dGℒ formula

$$x \geq 0 \to \langle (x := x-1 \cap x := x-2)^*\rangle 0 \leq x < 2$$

is provable as shown in Fig. 3, where $\beta \cap \gamma$ is short for $x := x-1 \cap x := x-2$ and $\langle (\beta \cap \gamma)^* \rangle 0{\leq}x{<}2$ short for $\langle (x := x-1 \cap x := x-2)^* \rangle 0 \leq x < 2$:

Example 9 (Hybrid game). The dGℒ formula

$$\langle (x := 1; x' = 1^d \cup x := x-1)^*\rangle 0 \leq x < 1$$

is provable as shown in Fig. 5, where the notation $\langle (\beta \cup \gamma)^* \rangle 0 \leq x < 1$ is short for $\langle (x := 1; x' = 1^d \cup x := x-1)^* \rangle (0 \leq x < 1)$: The proof steps for β use in $\langle ' \rangle$ that $t \mapsto x + t$ is the solution of the differential equation, so the subsequent use of $\langle := \rangle$ substitutes 1 in for x to obtain $t \mapsto 1 + t$. Recall from Lecture 22 that the winning regions for this formula need $>\omega$ iterations to converge. It is still provable easily.

$$\begin{array}{rr}
 & * \\ \hline
\mathbb{R} & \forall x\,(x \neq 1 \lor p(a,0) \land p(0,a) \to p(x,a)) \to (\mathit{true} \to p(x,a)) \\ \hline
\langle ; \rangle,\langle := \rangle & \forall x\,(x \neq 1 \lor \langle \beta \rangle p(x,a) \land \langle \gamma \rangle p(x,a) \to p(x,a)) \to (\mathit{true} \to p(x,a)) \\ \hline
\langle \cup \rangle,\langle ^{d} \rangle & \forall x\,(x \neq 1 \lor \langle \beta \cap \gamma \rangle p(x,a) \to p(x,a)) \to (\mathit{true} \to p(x,a)) \\ \hline
\text{US} & \forall x\,(x \neq 1 \lor \langle \beta \cap \gamma \rangle \langle (\beta \cap \gamma)^* \rangle x \neq 1 \to \langle (\beta \cap \gamma)^* \rangle x \neq 1) \to (\mathit{true} \to \langle (\beta \cap \gamma)^* \rangle x \neq 1) \\ \hline
\langle * \rangle,\forall,\text{MP} & \mathit{true} \to \langle (\beta \cap \gamma)^* \rangle x \neq 1 \\ \hline
\mathbb{R} & x = 1 \land a = 1 \to \langle (\beta \cap \gamma)^* \rangle x \neq 1
\end{array}$$

Figure 3: dG$\mathcal{L}$ Angel proof for choice game Example 7
$x = 1 \land a = 1 \to \langle (x := a; a := 0 \cap x := 0)^* \rangle x \neq 1$

$$\begin{array}{rr}
 & * \\ \hline
\mathbb{R} & \forall x\,(0{\leq}x{<}2 \lor p(x-1) \land p(x-2) \to p(x)) \to (\mathit{true} \to p(x)) \\ \hline
\langle := \rangle & \forall x\,(0{\leq}x{<}2 \lor \langle \beta \rangle p(x) \land \langle \gamma \rangle p(x) \to p(x)) \to (\mathit{true} \to p(x)) \\ \hline
\langle \cup \rangle,\langle ^{d} \rangle & \forall x\,(0{\leq}x{<}2 \lor \langle \beta \cap \gamma \rangle p(x) \to p(x)) \to (\mathit{true} \to p(x)) \\ \hline
\text{US} & \forall x\,(0{\leq}x{<}2 \lor \langle \beta \cap \gamma \rangle \langle (\beta \cap \gamma)^* \rangle 0{\leq}x{<}2 \to \langle (\beta \cap \gamma)^* \rangle 0{\leq}x{<}2) \to (\mathit{true} \to \langle (\beta \cap \gamma)^* \rangle 0{\leq}x{<}2) \\ \hline
\langle * \rangle,\forall,\text{MP} & \mathit{true} \to \langle (\beta \cap \gamma)^* \rangle 0{\leq}x{<}2 \\ \hline
\mathbb{R} & x \geq 0 \to \langle (\beta \cap \gamma)^* \rangle 0{\leq}x{<}2
\end{array}$$

Figure 4: dG$\mathcal{L}$ Angel proof for 2-Nim-type game Example 8
$x \geq 0 \to \langle (x := x-1 \cap x := x-2)^* \rangle 0 \leq x < 2$

$$\begin{array}{rr}
 & * \\ \hline
\mathbb{R} & \forall x\,(0 \leq x < 1 \lor \forall t{\geq}0\, p(1+t) \lor p(x-1) \to p(x)) \to (\mathit{true} \to p(x)) \\ \hline
\langle := \rangle & \forall x\,(0 \leq x < 1 \lor \langle x := 1 \rangle \neg \exists t{\geq}0\, \langle x := x+t \rangle \neg p(x) \lor p(x-1) \to p(x)) \to (\mathit{true} \to p(x)) \\ \hline
\langle ' \rangle & \forall x\,(0 \leq x < 1 \lor \langle x := 1 \rangle \neg \langle x' = 1 \rangle \neg p(x) \lor p(x-1) \to p(x)) \to (\mathit{true} \to p(x)) \\ \hline
\langle ; \rangle,\langle ^{d} \rangle & \forall x\,(0 \leq x < 1 \lor \langle \beta \rangle p(x) \lor \langle \gamma \rangle p(x) \to p(x)) \to (\mathit{true} \to p(x)) \\ \hline
\langle \cup \rangle & \forall x\,(0 \leq x < 1 \lor \langle \beta \cup \gamma \rangle p(x) \to p(x)) \to (\mathit{true} \to p(x)) \\ \hline
\text{US} & \forall x\,(0{\leq}x{<}1 \lor \langle \beta \cup \gamma \rangle \langle (\beta \cup \gamma)^* \rangle 0 \leq x < 1 \to \langle (\beta \cup \gamma)^* \rangle 0 \leq x < 1) \to (\mathit{true} \to \langle (\beta \cup \gamma)^* \rangle 0 \leq x < 1) \\ \hline
\langle * \rangle,\forall,\text{MP} & \mathit{true} \to \langle (\beta \cup \gamma)^* \rangle 0 \leq x < 1
\end{array}$$

Figure 5: dG$\mathcal{L}$ Angel proof for hybrid game Example 9
$\langle (x := 1; x' = 1^{d} \cup x := x-1)^* \rangle 0 \leq x < 1$

7 There and Back Again Game

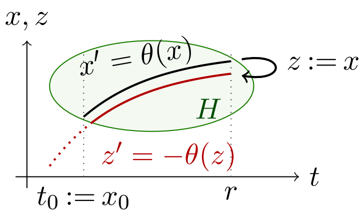
Quite unlike in hybrid systems and (poor test) differential dynamic logic [Pla08, Pla12], every hybrid game containing a differential equation $x' = \theta \,\&\, H$ with evolution domain constraints H can be replaced equivalently by a hybrid game without evolution domain constrains (even using poor tests, i.e. each test $?H$ uses only first-order formulas H). Evolution domains are definable in hybrid games and can, thus, be removed equivalently.

Lemma 10 (Domain reduction [Pla13, Pla12]). *Evolution domains of differential equations are definable as hybrid games: For every hybrid game there is an equivalent hybrid game that has no evolution domain constraints, i.e. all continuous evolutions are of the form $x' = \theta$.*

Proof. For notational convenience, assume the (vectorial) differential equation $x' = \theta(x)$ to contain a clock $x_0' = 1$ and that t_0 and z are fresh variables. Then $x' = \theta(x) \,\&\, H(x)$ is equivalent to the hybrid game:

$$t_0 := x_0; x' = \theta(x); (z := x; z' = -\theta(z))^d; ?(z_0 \geq t_0 \rightarrow H(z)) \tag{1}$$

See Fig. 6 for an illustration. Suppose the current player is Angel. The idea behind game equivalence (1) is that the fresh variable t_0 remembers the initial time x_0, and Angel then evolves forward along $x' = \theta(x)$ for any amount of time (Angel's choice). Afterwards, the opponent Demon copies the state x into a fresh variable (vector) z that he can evolve backwards along $(z' = -\theta(z))^d$ for any amount of time (Demon's choice). The original player Angel must then pass the challenge $?(z_0 \geq t_0 \rightarrow H(z))$, i.e. Angel loses immediately if Demon was able to evolve backwards and leave region $H(z)$ while satisfying $z_0 \geq t_0$, which checks that Demon did not evolve backward for longer than Angel evolved forward. Otherwise, when Angel passes the test, the extra variables t_0, z become irrelevant (they are fresh) and the game continues from the current state x that Angel chose in the first place (by selecting a duration for the evolution that Demon could not invalidate). □



Angel plays forward game, reverts flow and time x_0;
Demon checks H in backwards game until initial t_0

Figure 6: "There and back again game": Angel evolves x forwards in time along $x' = \theta(x)$, Demon checks evolution domain backwards in time along $z' = -\theta(z)$ on a copy z of the state vector x

Lemma 10 can eliminate all evolution domain constraints equivalently in hybrid games from now on. While evolution domain constraints are fundamental parts of standard hybrid systems [Hen96, HKPV95, ACHH92, Pla08], they turn out to be mere convenience notation for hybrid games. In that sense, hybrid games are more fundamental than hybrid systems, because they feature elementary operators.

Exercises

Exercise 1 (***). The following formula was proved using dG$\mathcal{L}$'s hybrid games type proof rules in Fig. 2

$$x \geq 0 \rightarrow \langle (x := x - 1)^* \rangle 0 \leq x < 1$$

Try to prove it using the convergence rule *con*′ instead.

References

[ACHH92] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, volume 736 of *LNCS*, pages 209–229. Springer, 1992.

[Hen96] Thomas A. Henzinger. The theory of hybrid automata. In *LICS*, pages 278–292, Los Alamitos, 1996. IEEE Computer Society. doi:10.1109/LICS.1996.561342.

[HKPV95] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In Frank Thomson Leighton and Allan Borodin, editors, *STOC*, pages 373–382. ACM, 1995. doi:10.1145/225058.225162.

[HMP77] David Harel, Albert R. Meyer, and Vaughan R. Pratt. Computability and completeness in logics of programs (preliminary report). In *STOC*, pages 261–268. ACM, 1977.

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

[Pla12] André Platzer. The complete proof theory of hybrid systems. In *LICS*, pages 541–550. IEEE, 2012. doi:10.1109/LICS.2012.64.

[Pla13] André Platzer. A complete axiomatization of differential game logic for hybrid games. Technical Report CMU-CS-13-100R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January, Revised and extended in July 2013.

15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Logical Theory & Completeness

André Platzer

Carnegie Mellon University
Lecture 24

1 Introduction

This course has studied a number of logics, first-order logic FOL in Lecture 2, differential dynamic logic d$\mathcal{L}$ [Pla08, Pla10a, Pla12c, Pla12b] in Lecture 3 and Lecture 4 and following, differential temporal dynamic logic dTL [Pla07, Pla10a, Chapter 4] in Lecture 16 and 17, as well as differential game logic dG$\mathcal{L}$ [Pla13] since Lecture 22. There are other logics for cyber-physical systems that have not been included in this course, but share similar principles for further dynamical aspects. Such logics include quantified differential dynamic logic Qd$\mathcal{L}$ for distributed hybrid systems [Pla10b, Pla12a], which are systems that are simultaneously distributed systems and hybrid systems, as well as stochastic differential dynamic logic Sd$\mathcal{L}$ for stochastic hybrid systems [Pla11], which simultaneously involve stochastic dynamics and hybrid dynamics. Logics play a stellar role not just in cyber-physical systems, but also many other contexts. Other important logics include propositional logic, restrictions of first-order logic to certain theories, such as first-order logic of real arithmetic [Tar51], and higher-order logic [And02]. But there are numerous other important and successful logics.

In this lecture, we take a step back and study some common important concepts in logic. This study will necessarily be hopelessly incomplete for lack of time. But it should give you a flavor of important principles and concepts in logic that we have not already run across explicitly in earlier lectures of this course. We will also have the opportunity to apply these more general concepts to cyber-physical systems and learn more about them in the next lecture.

2 Soundness

The most important parts of a logic $\mathcal{L}$ are the following. The logic $\mathcal{L}$ defines what the *syntactically well-formed formulas* are. Every well-formed formula carries meaning, which the *semantics of formulas* in $\mathcal{L}$ defines. The semantics defines a relation $\vDash$ between sets of formulas and formulas, in which $\Phi \vDash \phi$ holds iff ϕ is a semantic consequence of the set of formulas Φ, i.e. ϕ is true (usually written $\nu \models \phi$) in every interpretation ν for which all formulas $\psi \in \Phi$ are true. The most important case for our purposes is the case $\Phi = \emptyset$ of validity, in which case $\vDash \phi$ holds iff ϕ is valid, i.e. true ($\nu \models \phi$) in all interpretations ν of $\mathcal{L}$. An interpretation ν in which ϕ is true (i.e. $\nu \models \phi$) is also called a *model* of ϕ.

For the case of first-order logic FOL, Lecture 2 defined both their syntax and semantics. The syntax and semantics of differential dynamic logic d$\mathcal{L}$ has been defined in Lecture 3 and Lecture 4.

The syntax of a logic $\mathcal{L}$ defines what we can write down that carries meaning. The semantics of a logic $\mathcal{L}$ then defines what the meaning of the syntactic formulas is. The semantics, in particular, defines which formulas express true facts about the world, either in a particular interpretation ν or about the world in general (for valid formulas, which are true regardless of the interpretation). Yet, the semantics is usually highly ineffective, so that it cannot be used directly to find out whether a formula is valid. Just think of formulas in differential dynamic logic that express safety properties of hybrid systems. It would not get us very far if we were to try to establish the truth of such a formula by literally computing the semantics (which includes executing the hybrid system) in every initial state, of which there are uncountably infinitely many.

Instead, logics come with *proof calculi* that can be used to establish validity of logical formulas in the logic $\mathcal{L}$. Those proof calculi comprised *axioms* (Lecture 5) and *proof rules* (Lecture 6 and others), which can be combined to prove or derive logical formulas of the logic $\mathcal{L}$. The proof calculus of the logic $\mathcal{L}$ defines a relation $\vdash$ between sets of formulas and formulas, in which $\Phi \vdash \phi$ holds iff ϕ is provable from the set of formulas Φ. That is, there is a proof of ϕ in the proof calculus of $\mathcal{L}$ that uses only assumptions from Φ. The most important case for our purposes is again $\Phi = \emptyset$, in which case $\vdash \phi$ holds iff ϕ is provable in the proof calculus of $\mathcal{L}$, i.e. there is a proof of ϕ.

Of course, only some formulas of $\mathcal{L}$ are provable, not all of them. The formula $p \land \neg p$ should not be provable in any proper logic, because it is inconsistently false and, thus, cannot possibly be valid.

We could have written down any arbitrary axiom, or we could have accidentally had a typo in the axioms. So a crucial question we have to ask (and have asked every time we introduced an axiom in other lectures of this course) is whether the axioms and proof rules are sound. In a nutshell, a proof calculus is sound if all provable formulas are valid.

Theorem 1 (Soundness [Pla08, Pla10a, Pla12b])**.** *The proof calculus of differential dynamic logic is* sound*, i.e.* $\vdash \subseteq \vDash$*, which means that* $\vdash \phi$ *implies* $\vDash \phi$ *for all* d$\mathcal{L}$ *formulas* ϕ*. That is, all provable* d$\mathcal{L}$ *formulas are valid.*

The significance of soundness is that, whatever formula we derive by using the d$\mathcal{L}$ proof rules and axioms, we can rest assured that it is valid, i.e. true in all states. In particular, it does not matter how big and complicated the formula might be, we know that it is valid as long as we have a proof for it. About the axioms, we can easily convince ourselves using a soundness proof why they are valid, and then conclude that all provable formulas are also valid, because they follow from sound axioms by sound proof rules.

Note 2 (Necessity of soundness)**.** *Soundness is a must for otherwise we could not trust our own proofs.*

3 Soundness Challenge for CPS

What good would it do to analyze safety of a CPS using a technique that is as faulty as the original CPS? If an unsound analysis technique says that a CPS is correct, we are, fundamentally, not much better off than without any analysis, because all we can conclude is that we did not find problems, not that there are none.[1] After all, an unsound analysis technique could say "correct", which might turn out to be a lie because the correctness statement itself was not valid.

Note 3 (Challenge of soundness)**.** *In a domain that is as challenging as cyber-physical systems and hybrid systems, it is surprisingly easy for analysis techniques to become unsound due to subtle flaws. Necessary conditions for soundness and the numerical decidability frontier have been identified in the literature [PC07, Col07]. The crux of the matter is that hybrid systems are subject to a numerical analogue of the halting problem of Turing machines [PC07].*

There is a shockingly large number of approaches that, for subtle reasons, are subject to the unsoundness resulting from non-observance of the conditions identified in [PC07, Col07]. Consequently, such approaches need some of the additional assumptions identified in [PC07, Col07] to have a chance to become sound.

[1] Notwithstanding of the fact that unsound analysis techniques can still be very useful in practice, especially if they identify problems in system designs. Yet, we should exercise great care in concluding anything from unsound techniques that have not found a problem. As has been aptly phrased by Dijkstra [Dij70]: "Program testing can be used to show the presence of bugs, but never to show their absence!"

4 First-Order Logic

Even though this course primarily studied extensions of first-order logic by dynamic modalities for hybrid systems instead of pure first-order logic, the sequent proof rules of propositional logic and quantifiers (instantiation and Skolemization) give a suitable proof calculus for first-order logic. And this suitability of the proof calculus for first-order logic is a much stronger statement than soundness.

Soundness is the question whether all provable formulas are valid and is a minimal requirement for proper logics. Completeness studies the converse question whether all valid formulas are provable.

The first-order logic proof calculus can be shown to be both sound and complete, which is a result that originates from Gödel's PhD thesis [Göd30], albeit in a different form.

Theorem 2 (Soundness & completeness of first-order logic)**.** *First-order logic is* sound, *i.e.* $\vdash\ \subseteq\ \vDash$, *which means that* $\vdash \phi$ *implies* $\vDash \phi$ *for all first-order formulas* ϕ *(all provable formulas are valid). First-order logic is* complete, *i.e.* $\vDash\ \subseteq\ \vdash$, *which means that* $\vDash \phi$ *implies* $\vdash \phi$ *for all first-order formulas* ϕ *(all valid formulas are provable). In particular, the provability relation* $\vdash$ *and the validity relation* $\vDash$ *coincide for first-order logic:* $\vdash\ =\ \vDash$. *The same holds in the presence of a set of assumptions* Γ, *i.e.* $\Gamma \vdash \phi$ *iff* $\Gamma \vDash \phi$.

This lecture will not set out for a direct proof of this result, because the techniques used for those proofs are interesting but would lead us too far astray. An indirect justification for what makes first-order logic so special that Theorem 2 can hold will be discussed later.

The following central result about compactness of first-order logic is of similar importance. Compactness is involved in most proofs of Theorem 2, but also easily follows from Theorem 2.

Theorem 3 (Compactness of first-order logic)**.** *First-order logic is* compact, *i.e.*

$$\Gamma \vDash A \iff E \vDash A \text{ for some finite } E \subseteq \Gamma \tag{1}$$

Proof. By Theorem 2, $\vdash\ =\ \vDash$. By completeness, semantic compactness theorem (1) is equivalent to the syntactic compactness theorem:

$$\Gamma \vdash A \iff E \vdash A \text{ for some finite } E \subseteq \Gamma \tag{2}$$

Condition (2) is obvious, because provability implies that there is a proof, which can, by definition, only use finitely many assumptions $E \subseteq \Gamma$. □

Compactness is equivalent to the finiteness property, which, for that reason, is usually simply referred to as compactness:

Corollary 4 (Finiteness)**.** *First-order logic satisfies the* finiteness property*, i.e.*

$$\Gamma \textit{ has a model} \iff \textit{all finite } E \subseteq \Gamma \textit{ have a model} \tag{3}$$

Proof. Compactness (Theorem 3) implies the finiteness property. The key observation is that Γ has no model iff $\Gamma \vDash \mathit{false}$, because if Γ has no model, then false holds in all models of Γ of which there are none. Conversely, the only chance for false to hold in all models of Γ is if there are no such models, since false never holds. By Theorem 3,

$$\Gamma \vDash \mathit{false} \iff \exists \text{finite } E \subseteq \Gamma \quad E \vDash \mathit{false}$$

Hence,

$$\Gamma \text{ has a model} \iff \Gamma \nvDash \mathit{false} \iff \forall \text{finite } E \subseteq \Gamma \quad E \nvDash \mathit{false} \iff \text{all finite } E \subseteq \Gamma \text{ have a model}$$

It is worth noting that, conversely, the finiteness property implies compactness.

$$\begin{aligned} \Gamma \vDash A &\iff \Gamma \cup \{\neg A\} \text{ has no model} \\ &\iff \text{some finite } E \subseteq \Gamma \cup \{\neg A\} \text{ has no model} && \text{by finiteness} \\ &\iff E \vDash A \text{ for some finite } E \subseteq \Gamma \end{aligned}$$

The last equivalence uses that we might as well include $\neg A$ in E, because if E has no model then neither does $E \cup \{\neg A\}$. □

5 Skolem-Herbrand-Löwenheim Theory

The value of a logical formula is subject to interpretation in the semantics of the logic. In a certain sense maybe the most naïve interpretation of first-order logic interprets all terms as themselves. Such an interpretation I is called *Herbrand model*. It stubbornly interprets a term $f(g(a), h(b))$ in the logic as itself: $[\![f(g(a), h(b))]\!]_I = f(g(a), h(b))$. And likewise for all other ground terms.

That may sound like a surprising and stubborn interpretation. But, even more surprisingly, it is not at all an uninsightful one, at least for first-order logic. So insightful, that it even deserves a name: Herbrand models. Certainly, it is one of the many permitted interpretations.

Definition 5 (Herbrand Model)**.** An interpretation I is called *Herbrand model* if it has the free semantics for ground terms, i.e.:

1. The domain D is the ground terms (i.e. terms without variables) $\mathrm{Trm}^0(\Sigma)$ over Σ
2. $I(f) : D^n \to D; (t_1, \dots, t_n) \mapsto f(t_1, \dots, t_n)$ for each function symbol f of arity n

Let Γ be a set of closed universal formulas. $\mathrm{Trm}^0(\Sigma)(\Gamma)$ is the set of all ground term instances of the formulas in Γ, i.e. with (all possible) ground terms in $\mathrm{Trm}^0(\Sigma)$ instantiated for the variables of the universal quantifier prefix.

$$\begin{aligned}\mathrm{Trm}^0(\Sigma)(\Gamma) = \{\phi(t_1,t_2,\dots,t_n) \;:\; & (\forall x_1\,\forall x_2\,\dots\forall x_n\,\phi(x_1,x_2,\dots,x_n)) \in \Gamma\\ & t_1,\dots,t_n \in \mathrm{Trm}^0(\Sigma),\ \text{for any } n \in \mathbb{N}\}\end{aligned}$$

That is, for any $n \in \mathbb{N}$ and for any formula

$$\forall x_1\,\forall x_2\,\dots\forall x_n\,\phi(x_1,x_2,\dots,x_n)$$

in Γ and for any ground terms $t_1,\dots,t_n \in \mathrm{Trm}^0(\Sigma)$, the set $\mathrm{Trm}^0(\Sigma)(\Gamma)$ contains the following ground instance of ϕ:

$$\phi(t_1,t_2,\dots,t_n)$$

Theorem 6 (Herbrand [Her30]). *Let Γ be a (suitable) set of first-order formulas (i.e. closed* universal formulas *without equality and with signature Σ having at least one constant).*

$$\begin{aligned}\Gamma \textit{ has a model} &\iff \Gamma \textit{ has a Herbrand model}\\ &\iff \textit{ground term instances } \mathrm{Trm}^0(\Sigma)(\Gamma) \textit{ of } \Gamma \textit{ have a model}\end{aligned}$$

Using the Herbrand theorem twice gives:
Γ has a model $\iff$ ground term instances $\mathrm{Trm}^0(\Sigma)(\Gamma)$ of Γ have a Herbrand model

Corollary 7. *Validity in first-order logic is semidecidable.*

Proof. For suitable first-order formulas F (i.e. $\neg F$ satisfies the assumptions of Theorem 6), semidecidability follows from the following reductions:

$$\begin{aligned}F \text{ valid} &\iff \neg F \text{ unsatisfiable}\\ &\iff \mathrm{Trm}^0(\Sigma)(\neg F) \text{ have no model} && \text{by Theorem 6}\\ &\iff \text{some finite subset of } \mathrm{Trm}^0(\Sigma)(\neg F) \text{ has no Herbrand model} && \text{by Corollary 4}\end{aligned}$$

Thus, it remains to consider the assumptions in Theorem 6 whether first-order formulas that are not suitable can be turned into formulas that are suitable. First of all, Σ can be assumed without loss of generality to have at least one constant symbol for, otherwise, a constant can be added to Σ without changing validity of F. Furthermore, a formula F is valid iff its universal closure is, where the universal closure of a formula F is obtained by prefixing F with universal quantifiers $\forall x$ for each variable x that occurs free in F. Finally, existential quantifiers in first-order formula $\neg F$ can be removed without affecting satisfiability by Skolemization, which introduces new function symbols much like the quantifier proof rules from Lecture 6 did. □

Note 10 (Limitations of Herbrand models)**.** *Herbrand models are not the cure for everything in first-order logic, because they unwittingly forget about the intimate relationship of the term $2+5$ to the term $5+2$ and, for that matter, to the term $8-1$. All those terms ought to denote the same identical object, but end up denoting different ground terms in Herbrand models. In particular, a Herbrand model would not mind at all if a unary predicate p would hold of $2+5$ but not hold for $5+2$ even though both ought to denote the same object. Thus, Herbrand models are a little weak in arithmetic, but otherwise incredibly powerful.*

Herbrand's theorem has a second form with a close resemblance to the core arguments of quantifier elimination in first order logic of real arithmetic from Lecture 18 and Lecture 19.

Theorem 8 (Herbrand's theorem: Herbrand disjunctions [Her30])**.** *For a quantifier-free formula $\phi(x)$ of a free variable x without equality*

$$\exists x\,\phi(x) \text{ valid} \iff \phi(t_1)\vee\dots\vee\phi(t_n) \text{ valid for some } n\in\mathbb{N} \text{ and ground terms } t_1,\dots,t_n$$

Proof. The proof follows directly from Theorem 6 and Corollary 4:

$$\begin{aligned}
&\exists x\,\phi(x) \text{ valid} && \\
&\iff \neg\exists x\,\phi(x) \text{ unsatisfiable} && \\
&\iff \forall x\,\neg\phi(x) \text{ has no model} && \\
&\iff \mathrm{Trm}^0(\Sigma)(\forall x\,\neg\phi(x)) \text{ has no model} && \text{by Theorem 6}\\
&\iff \{\neg\phi(t) \ :\ t \text{ ground term}\} \text{ has no model} && \text{by definition}\\
&\iff \{\neg\phi(t_1),\dots,\neg\phi(t_n)\} \text{ has no model for some } t_1,\dots,t_n \text{ and some } n && \text{by Corollary 4}\\
&\iff \neg\phi(t_1)\wedge\dots\wedge\neg\phi(t_n) \text{ has no model for some } n \text{ and some } t_1,\dots,t_n && \\
&\iff \phi(t_1)\vee\dots\vee\phi(t_n) \text{ valid for some } n \text{ and some } t_1,\dots,t_n && \square
\end{aligned}$$

Theorem 8 holds for first-order formulas with multiple existential quantifiers. More general forms of the Herbrand theorem hold for arbitrary first-order formulas that are not in the specific form assumed above [Her30].

These more general Herbrand theorems won't be necessary for us, because, for validity purposes, first-order formulas can be turned into the form $\exists x_1 \dots \exists x_n\,\phi(x_1,\dots,x_n)$ with quantifier-free $\phi(x_1,\dots,x_n)$ by introducing new function symbols for the universal quantifiers using essentially the quantifier proof rules from Lecture 6:[2]

[2] The new function symbols are usually called Skolem functions and the process called Skolemization, because Thoralf Skolem introduced them in the first correct proof of the Skolem-Löwenheim theorem [Sko20]. Strictly speaking, however, Herbrand functions and Herbrandization are the more adequate names, because Jacques Herbrand introduced this dual notion for the first proof of the Herbrand theorem [Her30]. Skolemization and Herbrandization are duals. Skolemization preserves satisfiability while Herbrandization preserves validity.

$$(\forall r)\ \frac{\Gamma \vdash \phi(s(X_1,..,X_n)), \Delta}{\Gamma \vdash \forall x\, \phi(x), \Delta}{}^{1} \qquad (\exists l)\ \frac{\Gamma, \phi(s(X_1,..,X_n)) \vdash \Delta}{\Gamma, \exists x\, \phi(x) \vdash \Delta}{}^{1}$$

[1] s is a new (Skolem-Herbrand) function and $X_1,..,X_n$ are all (existential) free logical variables of $\forall x\, \phi(x)$.

The clou about quantifier rules ∀r,∃l is that they preserve validity. By soundness, if their premiss is valid then so is their conclusion. Yet, in the case of rules ∀r,∃l the converse actually holds as well. If their conclusion is valid then so is their premiss. For rule ∀r, for example, the conclusion says that $\phi(x)$ holds for all values of x in all interpretations where Γ holds and Δ does not. Consequently, in those interpretations, $\phi(s(X_1,..,X_n))$ holds whatever the interpretation of s is, because s is a fresh function symbol, which, thus, does not appear in Γ, Δ.

Lemma 9 (Herbrandization)**.** *With each first-order logic formula ψ, a formula*

$$\exists x_1 \ldots \exists x_n\, \phi(x_1, \ldots, x_n)$$

with quantifier-free $\phi(x_1, \ldots, x_n)$ can be associated effectively that is valid if and only if ψ is. The formula $\exists x_1 \ldots \exists x_n\, \phi(x_1, \ldots, x_n)$ uses additional function symbols that do not occur in ψ.

Theorem 8 enables a second, more straightforward proof of the semidecidability of the validity problem of first-order logic:

Proof of Corollary 7. The semidecision procedure for validity of first-order logic formulas ψ proceeds as follows:

1. Herbrandize ψ to obtain a formula $\exists x_1 \ldots \exists x_n\, \phi(x_1, \ldots, x_n)$ by Lemma 9, which preserves validity.
2. Enumerate all $m \in \mathbb{N}$ and all ground terms t_i^j ($1 \leq j \leq n, 1 \leq i \leq m$), over the new signature.
   a) If the *propositional* formula

   $$\phi(t_1^1, \ldots, t_1^n) \vee \cdots \vee \phi(t_m^1, \ldots, t_m^n)$$

   is valid, then so is $\exists x_1 \ldots \exists x_n\, \phi(x_1, \ldots, x_n)$ and, hence, ψ is valid.

By Theorem 8 and Lemma 9, the procedure terminates for all valid first-order formulas. □

The procedure in this proof will always succeed but it enumerates the ground terms for instantiation rather blindly, which can cause for quite a bit of waiting. Nevertheless, refinements of this idea lead to very successful automated theorem proving techniques for first-order logic known as *instance-based methods* [BT10], which restrict the instantiation to instantiation-on-demand in various ways to make the procedure more goal-directed. There are also many successful automatic theorem proving procedures for first-order logic that are based on different principles, including tableaux and resolution [Fit96].

6 Back to CPS

First-order logic is beautiful, elegant, expressive, and simple. Unfortunately, however, it is not expressive enough for hybrid systems [Pla10a, Pla12b, Pla13]. As soon as we come back to studying hybrid systems, the situation gets more difficult. And that is not by accident, but, instead, a fundamental property of first-order logic and of hybrid systems. Per Lindström characterized first-order logic in a way that limits which properties stronger logics could possess [Lin69]. Hybrid systems themselves are also known not to be semidecidable.

Given that differential dynamic logic talks about properties of hybrid systems, and Turing machines are a special case, undecidability is not surprising. We show a very simple standalone proof of incompleteness by adapting a proof for programs, e.g., [Pla10c].

Theorem 10 (Incompactness)**.** *Differential dynamic logic is not compact.*

Proof. It is easy to see that there is a set of formulas that has no model even though all finite subsets have a model, consider:

$$\{\langle(x := x+1)^*\rangle x > y\} \cup \{\neg(x+n > y) \ : \ n \in \mathbb{N}\} \qquad \square$$

Hence, differential dynamic logic does not have the finiteness property, which is equivalent to compactness (Corollary 4).

Since soundness and completeness imply compactness (see proof of Theorem 3), incompactness implies incompleteness[3], because d$\mathcal{L}$ is sound. An explicit proof is as follows:

Theorem 11 (Incompleteness [Pla08])**.** *Differential dynamic logic has no effective sound and complete calculus.*

Proof. Suppose there was an effective sound and complete calculus for d$\mathcal{L}$. Consider a set Γ of formulas that has no model in which all finite subsets have a model, which exists by Theorem 10. Then $\Gamma \vDash 0 > 1$ is valid, thus provable by completeness. But since the proof is effective, it can only use finitely many assumptions $E \subset \Gamma$. Thus $E \vDash 0 > 1$ by soundness. But then the finite set E has no model, which is a contradiction. □

[3] Strictly speaking, incompleteness only follows for effective calculi. *Relative* soundness and completeness can still be proved for d$\mathcal{L}$ [Pla08, Pla10a, Pla12b], which gives very insightful characterizations of the challenges and complexities of hybrid systems.

Having said these negative (but necessary) results about differential dynamic logic (and, by classical arguments, any other approach for hybrid systems), let's return to the surprisingly amazing positive properties that differential dynamic logic possesses.

For one thing, the basis of differential dynamic logic is the first-order logic of real arithmetic, not arbitrary first-order logic. This enables a particularly pleasant form of Herbrand disjunctions resulting from quantifier elimination in real arithmetic (recall Lecture 18 and Lecture 19).

Definition 12 (Quantifier elimination)**.** A first-order theory admits *quantifier elimination* if, with each formula ϕ, a quantifier-free formula $\mathrm{QE}(\phi)$ can be associated effectively that is equivalent, i.e. $\phi \leftrightarrow \mathrm{QE}(\phi)$ is valid (in that theory).

Theorem 13 (Tarski [Tar51])**.** *The first-order logic of real arithmetic admits quantifier elimination and is, thus, decidable.*

Also recall from Lecture 18 and Lecture 19 that the quantifier-free formula $\mathrm{QE}(\phi)$ is constructed by substitution or virtual substitution from ϕ, with some side constraints on the parameter relations. The quantifier-elimination instantiations are more useful than Theorem 8, because the required terms for instantiation can be computed effectively and the equivalence holds whether or not the original formula ϕ was valid. This makes it possible to use the proof calculus of differential dynamic logic to synthesize constraints on the parameters to make an intended conjecture valid [Pla10a].

Exercises

Exercise 1. The arguments for incompleteness and incompactness of $\mathsf{d}\mathcal{L}$ hardly depend on $\mathsf{d}\mathcal{L}$, but, rather, only on $\mathsf{d}\mathcal{L}$'s ability to characterize natural numbers. Incompleteness and incompactness hold for other logics that characterize natural numbers due to a famous result of Gödel [Göd31]. Both the discrete and the continuous fragment of $\mathsf{d}\mathcal{L}$ can characterize the natural numbers [Pla08].

1. Show that the natural numbers can be characterized in the discrete fragment of $\mathsf{d}\mathcal{L}$, i.e. only using assignments and repetition.

2. Then go on to show that the natural numbers can also be characterized in the continuous fragment of $\mathsf{d}\mathcal{L}$, i.e. using only differential equations.

3. Conclude from this that both the discrete and the continuous fragment of $\mathsf{d}\mathcal{L}$ are not compact, nor is any other logic that can characterize the natural numbers.

References

[And02] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Kluwer, 2nd edition, 2002.

[BT10] Peter Baumgartner and Evgenij Thorstensen. Instance based methods - a brief overview. *KI*, 24(1):35–42, 2010.

[Col07] Pieter Collins. Optimal semicomputable approximations to reachable and invariant sets. *Theory Comput. Syst.*, 41(1):33–48, 2007. doi:10.1007/s00224-006-1338-3.

[DBL12] *Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25–28, 2012*. IEEE, 2012.

[Dij70] Edsger Wybe Dijkstra. Structured programming. In John Buxton and Brian Randell, editors, *Software Engineering Techniques. NATO Software Engineering Conference 1969*. NATO Scientific Committee, 1970.

[Fit96] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, New York, 2nd edition, 1996.

[Göd30] Kurt Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Mon. hefte Math. Phys.*, 37:349–360, 1930.

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

[Her30] Jacques Herbrand. Recherches sur la théorie de la démonstration. *Travaux de la Société des Sciences et des Lettres de Varsovie, Class III, Sciences Mathématiques et Physiques*, 33:33–160, 1930.

[Lin69] Per Lindström. On extensions of elementary logic. *Theoria*, 35:1–11, 1969. doi:10.1111/j.1755-2567.1969.tb00356.x.

[PC07] André Platzer and Edmund M. Clarke. The image computation problem in hybrid systems model checking. In Alberto Bemporad, Antonio Bicchi, and Giorgio Buttazzo, editors, *HSCC*, volume 4416 of *LNCS*, pages 473–486. Springer, 2007. doi:10.1007/978-3-540-71493-4_37.

[Pla07] André Platzer. A temporal dynamic logic for verifying hybrid system invariants. In Sergei N. Artëmov and Anil Nerode, editors, *LFCS*, volume 4514 of *LNCS*, pages 457–471. Springer, 2007. doi:10.1007/978-3-540-72734-7_32.

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

[Pla10b] André Platzer. Quantified differential dynamic logic for distributed hybrid systems. In Anuj Dawar and Helmut Veith, editors, *CSL*, volume 6247 of *LNCS*, pages 469–483. Springer, 2010. doi:10.1007/978-3-642-15205-4_36.

[Pla10c] André Platzer. Theory of dynamic logic. Lecture Notes 15-816 Modal Logic, Carnegie Mellon University, 2010. URL: http://www.cs.cmu.edu/~fp/courses/15816-s10/lectures/25-DLtheo.pdf.

[Pla11] André Platzer. Stochastic differential dynamic logic for stochastic hybrid programs. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *CADE*, volume 6803 of *LNCS*, pages 431–445. Springer, 2011. doi:10.1007/978-3-642-22438-6_34.

[Pla12a] André Platzer. A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems. *Logical Methods in Computer Science*, 8(4):1–44, 2012. Special issue for selected papers from CSL'10. doi:10.2168/LMCS-8(4:17)2012.

[Pla12b] André Platzer. The complete proof theory of hybrid systems. In *LICS* [DBL12], pages 541–550. doi:10.1109/LICS.2012.64.

[Pla12c] André Platzer. Logics of dynamical systems. In *LICS* [DBL12], pages 13–24. doi:10.1109/LICS.2012.13.

[Pla13] André Platzer. A complete axiomatization of differential game logic for hybrid games. Technical Report CMU-CS-13-100R, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, January, Revised and extended in July 2013.

[Sko20] Thoralf Skolem. Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze nebst einem Theorem über dichte Mengen. *Videnskapsselskapet Skrifter, I. Matematisk-naturvidenskabelig Klasse*, 6:1–36, 1920.

[Tar51] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, 2nd edition, 1951.