André Platzer

# Lecture Notes on Foundations of Cyber-Physical Systems

15-424/624/824 Foundations of Cyber-Physical Systems

# Chapter 5
# Dynamical Systems & Dynamic Axioms

**Synposis** This central chapter develops a logical characterization of the dynamics of hybrid programs in differential dynamic logic. It investigates fundamental compositional reasoning principles that capture how the truth of a property of a more complex hybrid program relates to the truth of corresponding properties of simpler program fragments. This leads to dynamic axioms for dynamical systems, with one axiom for each type of dynamics. These dynamic axioms enable rigorous reasoning for CPS models and begin an axiomatization of differential dynamic logic, which turns the specification logic dL into a verification logic for CPS. While more advanced aspects of loops and differential equations are discussed in subsequent chapters, this chapter lays a pivotal foundation for all dynamical aspects of differential dynamic logic and its hybrid programs.

## 5.1 Introduction

Chap. 4 demonstrated how useful and crucial CPS contracts are for CPS. Their role and understanding goes beyond dynamic testing. In CPS, proven CPS contracts are infinitely more valuable than dynamically tested contracts, because, without sufficient care, 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 margins and fallback plans remain, the CPS 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

[1] Although, in combination with formal verification, the Simplex architecture can be understood as exploiting the relationship of dynamic contracts for safety purposes [15]. ModelPlex, which is based on differential dynamic logic, lifts this observation to a fully verified link from verified CPS models to verified CPS executions [5].

the effects of hybrid program operators is not only useful for conducting proofs, but also for developing and sharpening our intuition about hybrid programs. This phenomenon illustrates the more general point that proof and effect (and/or meaning) are intimately linked. Truly understanding effect is ultimately the same as, as well as a prerequisite to, understanding how to prove properties of that effect [7, 9, 10, 12]. You may have seen this point demonstrated already in other treatises on programming language, but it will shine in this chapter.

The route we choose to get to this level of understanding 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 [6, 7, 9, 10, 12]. While there are many more interesting things to say about dynamical systems and Kripke structures, this chapter will limit information to the truly essential parts that are crucial right now and leave more elaboration for later chapters. This chapter will give us the essential reasoning tools for cyber-physical systems and is, thus, of central importance.

The focus of this chapter is on a systematic development of the basic reasoning principles for cyber-physical systems. The goal is to cover all cyber-physical systems by identifying one fundamental reasoning principle for each of the operators of differential dynamic logic and, specifically, its hybrid programs. Once we have such a reasoning principle for each of the operators, the basic idea is that any arbitrary cyber-physical system can be analyzed by combining the various reasoning principles with one another, compositionally, by inspecting one operator at a time.

**Note 29 (Logical guiding principle: Compositionality)** *Since every CPS is modeled by a hybrid program*[a] *and all hybrid programs are combinations of simpler hybrid programs by one of a handful of program operators (such as* $\cup$ *and* ; *and* $^*$*), all CPS can be analyzed if only we identify one suitable analysis technique for each of the operators.*

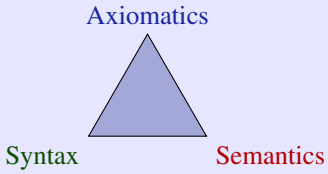
[a] To faithfully represent complex CPS, some models need an extension of hybrid programs, e.g., to hybrid games [11] or distributed hybrid programs [8], in which case suitable generalizations of the logical approach presented here work.

With enough understanding, this guiding principle ultimately succeeds [10–12]. It does, however, take more than one chapter to get there. This chapter settles for a systematic development of the reasoning principles for elementary operators in hybrid programs, leaving a detailed development of the others to later chapters.

This chapter is of central significance for the Foundations of Cyber-Physical Systems. It is the first of many chapters in this textbook where we observe a logical trichotomy between syntax, semantics, and axiomatics.

**Note 30 (Logical trinity)** *The concepts developed in this chapter illustrate the more general relation of* syntax *(which is notation),* semantics *(what carries meaning), and* axiomatics *(which internalizes semantic relations into uni-*

*versal syntactic transformations). These concepts and their relations jointly form the significant* logical trinity *of syntax, semantics, and axiomatics.*

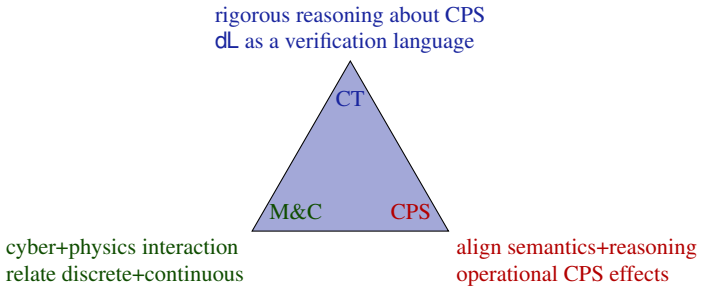



The most important learning goals of this chapter are:

**Modeling and Control:** We will understand the core principles behind CPS by understanding analytically and semantically how cyber and physical aspects are integrated and interact in CPS. This chapter will also begin to explicitly relate discrete and continuous systems, which ultimately leads to a fascinating view on understanding hybridness [10].

**Computational Thinking:** This chapter is devoted to the core aspects of reasoning rigorously about CPS models, which is critical to getting CPS right. CPS designs can be flawed for very subtle reasons. Without sufficient rigor in their analysis it can be impossible to spot the flaws, and it can be even more challenging to say for sure whether and why a design is no longer faulty. This chapter systematically develops one reasoning principle for each of the operators of hybrid programs. This chapter begins an *axiomatization* of differential dynamic logic dL [9, 10, 12] to lift dL from a specification language to a verification language for CPS.

**CPS Skills:** We will develop a deep understanding of the semantics of CPS models by carefully relating their semantics to their reasoning principles and aligning them in perfect unison. This understanding will also enable us to develop a better intuition for the operational effects involved in CPS.



## 5.2 Intermediate Conditions for CPS

Recall the bouncing ball from p. 109 in Chap. 4:

$$0 \leq x \land x = H \land v = 0 \land g > 0 \land 1 \geq c \geq 0 \rightarrow \\ \left[\left(\{x' = v, v' = -g \,\&\, x \geq 0\};\ (?x = 0; v := -cv \cup ?x \neq 0)\right)^*\right](0 \leq x \land x \leq H) \tag{4.24*}$$

To simplify the subsequent discussion, let's again drop the repetition ($^*$) for now:

$$0 \leq x \land x = H \land v = 0 \land g > 0 \land 1 \geq c \geq 0 \rightarrow \\ \left[\{x' = v, v' = -g \,\&\, x \geq 0\}; (?x = 0; v := -cv \cup ?x \neq 0)\right](0 \leq x \land x \leq H) \tag{5.1}$$

Of course, dropping the repetition grotesquely changes the behavior of the bouncing ball. It cannot even really bounce any longer now. It can merely fall and revert its velocity vector when on the ground but is then stuck. The single hop bouncing ball can only follow the first blue hop but not the gray remainder hops in Fig. 5.1. This degenerate model fragment is, nevertheless, an insightful stepping stone toward a proof of the full model. If we manage to prove (5.1), we certainly have not shown the full bouncing ball formula (4.24) with its loop. But it's a start, because the behavior modeled in (5.1) is a part of the behavior of (4.24). So it is useful (and easier) to understand (5.1) first.

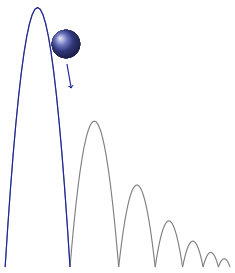


**Fig. 5.1** Sample trajectory of a single-hop bouncing ball (plotted as height over time) which can follow the first blue hop but is incapable of following the remaining hops shown in gray.

The dL formula (5.1) hasassumptions $0 \leq x \land x = H \land v = 0 \land g > 0 \land 1 \geq c \geq 0$ that can be used during the proof. It claims that the postcondition $0 \leq x \land x \leq H$ holds after all runs of the HP in the $[\cdot]$ modality. The top-level operator in the modality of (5.1) is a sequential composition (;), for which we need to find a proof argument.[2]

The HP in (5.1) follows a differential equation first and then, after the sequential composition (;), proceeds to run a discrete program $(?x = 0; v := -cv \cup ?x \neq 0)$. Depending on how long the HP follows its differential equation, the intermediate state after the differential equation and before the discrete program will be different.

[2] The way we proceed here to prove (5.1) is actually not the recommended way. We will develop an easier way. But it is instructive to understand the more verbose approach we take first. The first approach also prepares us for the challenges that lie ahead when proving properties of loops.

**Note 31 (Intermediate states of sequential compositions)** *The first HP $\alpha$ in a sequential compositions $\alpha;\beta$ 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 $\alpha$ and initial states for $\beta$. The intermediate states of $\alpha;\beta$ are the states $\mu$ in the semantics $[\![\alpha;\beta]\!]$ from Chap. 3:*

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

One can summarize what all intermediate states between the differential equation and the discrete program of (5.1) have in common. They differ by how long the CPS has followed the differential equation. But the intermediate states still have in common that they satisfy a logical formula $E$. Which logical formula that is, is, in fact, instructive to find out, but of no immediate concern for the rest of this chapter. So we invite you to find out how to choose $E$ for (5.1) before you compare your answer to the one we developed in Sect. 4.8.1.

For a HP that is a sequential composition $\alpha;\beta$ an *intermediate condition* is a formula that characterizes the intermediate states in between HP $\alpha$ and $\beta$. That is, for a dL formula

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

an intermediate condition is a formula $E$ such that the following dL formulas are valid:

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

The first dL formula expresses that intermediate condition $E$ characterizes the intermediate states accurately, i.e. $E$ actually holds after all runs of HP $\alpha$ from states satisfying $A$. The second dL 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 $\beta$ end up in $B$. That is, from all states satisfying $E$ (in particular from those that result by running $\alpha$ from a state satisfying $A$), $B$ holds after all runs of $\beta$. Depending on the precision of the intermediate condition $E$, this argument may require showing that $B$ holds after all runs of $\beta$ from extra states that are not reachable from $\omega$ by running $\alpha$ but happen to satisfy $E$ (blank in Fig. 5.2).

Intermediate condition contracts for sequential compositions are captured more concisely in the following proof rule by Tony Hoare [3]:

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

The two dL formulas above the bar of the rule are called *premises*. The dL formula below the bar is called *conclusion*. The above argument (informally) justifies the proof rule: if both premises are valid then the conclusion is valid, too. So, if we have a proof for each of the two premises, rule HM; gave us a proof of the conclusion.

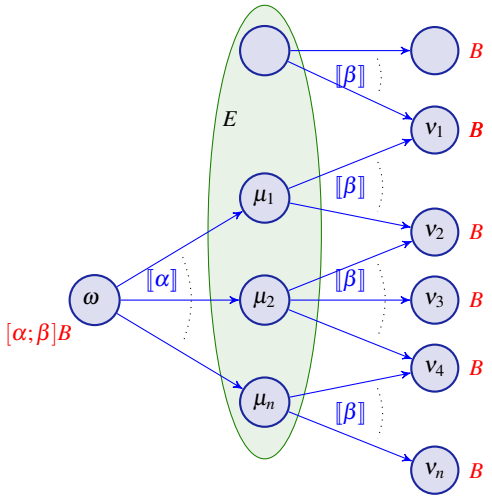



**Fig. 5.2** Intermediate conditions for sequential compositions

Since we will soon identify a better way of proving properties of sequential compositions, we do not pursue rule HM; further now. Yet, there are some circumstances under which an intermediate condition as in HM; actually simplifies your reasoning.

For now, we remark that, given an intermediate condition $E$, the rule HM; splits a proof of (5.1) into a proof of the following two premises, which we saw in Chap. 4:

$$0 \leq x \wedge x{=}H \wedge v{=}0 \wedge g > 0 \wedge 1{\geq}c{\geq}0 \rightarrow \left[x' = v, v' = -g \,\&\, x \geq 0\right] E \tag{4.29*}$$

$$E \rightarrow \left[?x = 0; v := -cv \cup ?x \neq 0\right] (0{\leq}x \wedge x{\leq}H) \tag{4.30*}$$

## 5.3 Dynamic Axioms for Dynamical Systems

This section develops axioms for decomposing dynamical systems, which are of central significance in this textbook. Each axiom describes the effect of one operator on hybrid programs in terms of simpler hybrid programs, thereby simultaneously serving as an explanation and as a basis for rigorous reasoning.

### *5.3.1 Dynamic Axioms for Nondeterministic Choices*

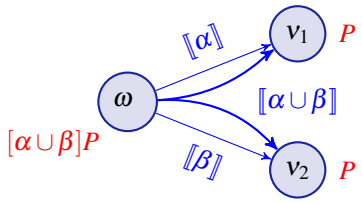
By the logical guiding principle of compositionality (Note 29), the next operator that we need to understand in order to finally proceed with a proof of (5.1) is the nondeterministic choice $?x = 0; v := -cv \cup ?x \neq 0$ which is the top-level operator in the modality of (4.30). By the compositionality principle, we zero in on the nondeterministic choice operator $\cup$ and pretend we already knew how to handle all

other operators in the formula. If we succeed in reducing the property of the nondeterministic choice in formula (4.30) to properties of its subprograms, then we can subsequently develop axioms that actually handle the remaining operators.

Recall the semantics of nondeterministic choices from Sect. 3.3.2:

$$[\![\alpha \cup \beta]\!] = [\![\alpha]\!] \cup [\![\beta]\!] \tag{5.2}$$

Remember that $[\![\alpha]\!]$ is a reachability relation on states, where $(\omega, \nu) \in [\![\alpha]\!]$ iff HP $\alpha$ can run from state $\omega$ to state $\nu$. Let us illustrate graphically what (5.2) means:




According to the reachability relation $[\![\alpha]\!]$, a number of states $\nu_i$ are reachable by running HP $\alpha$ from some initial state $\omega$.[3] According to $[\![\beta]\!]$, a number of (possibly different) states $\nu_i$ are reachable by running HP $\beta$ from the same initial state $\omega$. By the semantic equation (5.2), running $\alpha \cup \beta$ from $\omega$ can exactly give us any of those possible outcomes that result from either running $\alpha$ or $\beta$. And there was nothing special about the initial state $\omega$. The same principle holds for all other states as well.

**Note 32 ($\cup$)** *The nondeterministic choice $\alpha \cup \beta$ can lead to exactly the states to which either $\alpha$ could take us or to which $\beta$ 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 $\alpha$ or of $\beta$, nondeterministically. So both the behaviors of $\alpha$ and $\beta$ are possible when running $\alpha \cup \beta$.*

If we want to understand whether and where dL formula $[\alpha \cup \beta]P$ is true, we need to understand which states the modality $[\alpha \cup \beta]$ refers to. In which states does $P$ have to be true so that $[\alpha \cup \beta]P$ is true in state $\omega$?

By definition of the semantics, $P$ needs to be true in all states that $\alpha \cup \beta$ can reach according to $[\![\alpha \cup \beta]\!]$ from $\omega$ for $[\alpha \cup \beta]P$ to be true in $\omega$. Referring to semantics (5.2) or looking at its illustration, shows us that this includes exactly all states that $\alpha$ can reach from $\omega$ according to $[\![\alpha]\!]$, hence $[\alpha]P$ has to be true in $\omega$. It also includes all states that $\beta$ can reach from $\omega$, hence $[\beta]P$ has to be true in $\omega$.

Consequently,

$$\omega \in [\![[\alpha]P]\!] \quad \text{and} \quad \omega \in [\![[\beta]P]\!] \tag{5.3}$$

are necessary conditions for

$$\omega \in [\![[\alpha \cup \beta]P]\!] \tag{5.4}$$

[3] The diagram only illustrates one such state $\nu_1$ for visual conciseness. But $\nu_1$ should be thought of as a generic representative for any such state that $\alpha$ can reach from the initial state $\omega$.

That is, unless (5.3) holds, (5.4) cannot possibly hold. So (5.3) is necessary for (5.4). Are there any states missing? Are there any states that (5.4) would require to satisfy $P$, which (5.3) does not already ensure to satisfy $P$? No, because, by (5.2), $\alpha \cup \beta$ does not admit any behavior that neither $\alpha$ nor $\beta$ can exhibit. Hence (5.3) is also sufficient for (5.4), i.e. (5.3) implies (5.4). So (5.3) and (5.4) are equivalent.

When adopting a more logical language again, this justifies:

$$\omega \in [\![[\alpha \cup \beta]P \leftrightarrow [\alpha]P \wedge [\beta]P]\!]$$

This reasoning did not depend on the particular state $\omega$ but holds for all $\omega$. Therefore, the formula $[\alpha \cup \beta]P \leftrightarrow [\alpha]P \wedge [\beta]P$ is valid, written:

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

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

**Lemma 5.1 ($[\cup]$ soundness).** *The axiom of (nondeterministic) choice is* sound*, i.e. all its instances are valid:*

$$[\cup] \quad [\alpha \cup \beta]P \leftrightarrow [\alpha]P \wedge [\beta]P$$

Nondeterministic choices split into their alternatives in axiom $[\cup]$. From right to left: If all $\alpha$ runs lead to states satisfying $P$ (i.e., $[\alpha]P$ is true) and all $\beta$ runs lead to states satisfying $P$ (i.e., $[\beta]P$ is true), then all runs of HP $\alpha \cup \beta$, which may choose between following $\alpha$ and following $\beta$, also lead to states satisfying $P$ (i.e., $[\alpha \cup \beta]P$ is true). The converse implication from left to right holds, because $\alpha \cup \beta$ can run all runs of $\alpha$ and all runs of $\beta$, so all runs of $\alpha$ (and of $\beta$) lead to states satisfying $P$ if $[\alpha \cup \beta]P$. We will mark the structurally complex formula in blue.

Armed with this axiom $[\cup]$ at our disposal, we can now easily make the following inference just by invoking the equivalence that $[\cup]$ justifies.

$$[\cup]\frac{A \rightarrow [\alpha]B \wedge [\beta]B}{A \rightarrow [\alpha \cup \beta]B}$$

Let's elaborate. If we want to prove the conclusion

$$A \rightarrow [\alpha \cup \beta]B \tag{5.5}$$

then we can instead prove the premise

$$A \rightarrow [\alpha]B \wedge [\beta]B \tag{5.6}$$

because by $[\cup]$, or rather an instance of $[\cup]$ formed by using $B$ for $P$, we know:

$$[\alpha \cup \beta]B \leftrightarrow [\alpha]B \wedge [\beta]B \tag{5.7}$$

Since (5.7) is a valid equivalence, its left-hand side and its right-hand side are equivalent. Wherever its left-hand side occurs, we can equivalently replace it by its right-hand side, since both are equivalent.[4] Thus, replacing the place where the left-hand side of (5.7) occurs in (5.5) by the right-hand side of (5.7) gives us the formula (5.6) that is equivalent to (5.5). After all, according to the valid equivalence (5.7) justified by axiom $[\cup]$, (5.6) can be obtained from (5.5) just by replacing a formula with one that is equivalent (recall Exercise 4.2).

Actually, stepping back, the same argument can be made to go from (5.6) to (5.5) instead of from (5.5) to (5.6), because (5.7) is an equivalence. Both ways of using $[\cup]$ are perfectly correct. Although the direction that gets rid of the $\cup$ operator is more useful, because it made progress (getting rid of an HP operator).

Yet axiom $[\cup]$ can also be useful in many more situations. For example, axiom $[\cup]$ also justifies the inference

$$[\cup]\frac{[\alpha]A \wedge [\beta]A \rightarrow B}{[\alpha \cup \beta]A \rightarrow B}$$

which follows from the left-to-right implication of equivalence axiom $[\cup]$.

For the bouncing ball, axiom $[\cup]$ will decompose formula (4.30) and reduce it to:

$$E \rightarrow \left[?x = 0; v := -cv\right](0 \leq x \wedge x \leq H) \wedge \left[?x \neq 0\right](0 \leq x \wedge x \leq H) \tag{5.8}$$

A general design principle behind all dL axioms is most noticeable in axiom $[\cup]$. All equivalence axioms of dL 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 $\alpha \cup \beta$ into separate properties of smaller fragments $\alpha$ and $\beta$. While we might end up with more subproperties (like we do in the case of axiom $[\cup]$), each of them is structurally simpler, because it involves less program operators. This decomposition of systems into their fragments makes the verification problem tractable and is good for scalability purposes, because it reduces the study of complex systems successively to a study of many but smaller subsystems of which there are only finitely many. For these symbolic structural decompositions, it is very helpful that dL is a full logic that is closed under all logical operators [11, 12], including disjunction and conjunction, for then both sides in axiom $[\cup]$ are dL formulas again (unlike in Hoare logic [3]). This also turns out to be an advantage for computing invariants [2, 7, 13].

Axiom $[\cup]$ allows us to understand and handle $[\alpha \cup \beta]P$ properties. If we find appropriate axioms for all the other operators of hybrid programs, including $;, ^*, :=, x'$, then we have a way of handling all hybrid programs, because we merely need to simplify the verification question by subsequently decomposing it with the respective axiom. Even if a full account of this principle is significantly more complicated, such recursive decomposition indeed ultimately succeeds [10, 12].

[4] This will be made rigorous in Chap. 6 following contextual equivalence reasoning [12].

### 5.3.2 Soundness

The definition of soundness in Lemma 5.1 was not specific to axiom [∪], but applies to all dL axioms, so we discuss soundness once and for all.

**Definition 5.1 (Soundness).** An axiom is *sound* iff all its instances are valid, i.e. true in all states.

From now on, every time we see a formula of the form $[\alpha \cup \beta]P$, we remember that axiom [∪] identifies the corresponding formula $[\alpha]P \wedge [\beta]P$ that is equivalent to it. Whenever we find a formula $[\gamma \cup \delta]Q$, we also remember that axiom [∪] says that formula $[\gamma]Q \wedge [\delta]Q$ is equivalent to it, just by instantiation [12] of axiom [∪]. The fact that axiom [∪] is sound ensures that we do not need to worry about whether such reasoning is correct every time we need it. Soundness of [∪] guarantees that wvery instance of [∪] is sound [12]. We can, thus, treat axiom [∪] syntactically and mechanically and apply it as needed, like a machine would.

But because soundness is such a big deal (a *conditio sine qua non* in logic, i.e., something without which logic could not be), we will prove soundness of axiom [∪] carefully, even if we essentially already did in our informal argument above.

*Proof (of Lemma 5.1).* The fact that axiom [∪] is sound can be proved as follows. Since $[\![\alpha \cup \beta]\!] = [\![\alpha]\!] \cup [\![\beta]\!]$, we have that $(\omega, \nu) \in [\![\alpha \cup \beta]\!]$ iff $(\omega, \nu) \in [\![\alpha]\!]$ or $(\omega, \nu) \in [\![\beta]\!]$. Thus, $\omega \in [\![[\alpha \cup \beta]P]\!]$ iff $\omega \in [\![[\alpha]P]\!]$ and $\omega \in [\![[\beta]P]\!]$. □

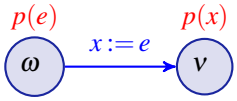
Why is soundness so critical? Well, because, without it, we could accidentally declare a system safe that is not in fact safe, which would defeat the whole purpose of verification and possibly put human lives in jeopardy when they are trusting their lives on an unsafe CPS. Unfortunately, soundness is actually not granted in all verification techniques for hybrid systems. But we will make it a point in this book to only ever use sound reasoning and scrutinizing all verification for soundness right away. Soundness is something that is comparably easy to establish in logic and proof approaches, because it localizes into the separate study of soundness of each of its axioms.

### 5.3.3 Dynamic Axioms for Assignments

Part of the dL formula (5.8) that remains to be shown after using the [∪] axiom involves another sequential composition $?x = 0; v := -cv$. But even if Sect. 5.2 already discussed one possibility for reducing safety properties of sequential compositions to properties of the parts using appropriate intermediate conditions, we still need a way of handling the remaining assignment and test. Let's start with the assignment.

HPs may involve discrete assignments. Recall their semantics from Sect. 3.3.2:

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




How can dL formula $[x:=e]p(x)$ be rephrased equivalently in simpler ways? It expresses that $p(x)$ holds always after assigning the value of term $e$ to variable $x$. Well, in fact, there is only exactly one way of assigning $e$ to $x$. So, the formula $[x:=e]p(x)$ expresses that $p(x)$ holds after changing the value of $x$ to that of $e$.

**Lemma 5.2 ([:=] soundness).** *The assignment axiom is sound:*

$$[:=] \quad [x:=e]p(x) \leftrightarrow p(e)$$

The assignment axiom [:=] expresses that $p(x)$ is true after the discrete assignment assigning term $e$ to $x$ iff $p(e)$ has already been true before that change, since the assignment $x:=e$ will change the value of variable $x$ to the value of $e$.

For example, axiom [:=] immediately allows us to conclude that the dL formula $[x:=a\cdot x]x\cdot(x+1)\geq 0$ is equivalent to first-order formula $(a\cdot x)\cdot(a\cdot x+1)\geq 0$, which is formed by replacing all free occurrences of $x$ in postcondition $x\cdot(x+1)\geq 0$ by its new value $a\cdot x$.

If we succeed splitting (5.8) into pieces, including its sequential composition $[?x=0;v:=-cv](0\leq x\wedge x\leq H)$ with yet another intermediate condition $F$ according to Sect. 5.2, we will eventually have to prove a number of dL formulas including:

$$F\rightarrow[v:=-cv](0\leq x\wedge x\leq H)$$

The assignment axiom [:=] equivalently reduces this formula to $F\rightarrow 0\leq x\wedge x\leq H$, because the affected variable $v$ does not occur in the postcondition. That is quite peculiar for the bouncing ball, because it will make the damping coefficient $c$ disappear entirely from the proof. While we are always happy to see complexity reduced, this should make us pause our train of thought for a moment. Chapter 4 taught us that the safety of the bouncing ball depends on the damping coefficient being $c\leq 1$. How could the proof ever possibly succeed if we misplace $c$ in the proof?

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

While the bouncing ball exhibits unsafe behavior if $c>1$, that counterexample required the ball to bounce back up from the ground, which is a capability that the simplified single-hop bouncing ball (5.1) lost compared to the full model with its repetition. That explains why our proof attempt of (5.1) can succeed when it removes $c$, but also indicates that we will need to make sure not to ignore the velocity $v$ whose change depends on $c$ in a proof of the repetitive bouncing ball (Chap. 7).

**Expedition 5.1 (Admissibility caveats for the $p(x)$ notation in axioms)**

There is a simple and elegant way of understanding the notation $p(x)$ and $p(e)$ in axiom [:=], which, unfortunately, needs more elaboration than the present stage of this textbook provides. After that sophistication, uniform substitutions [12] justify that both can merely be read as predicate symbol $p$ applied to the variable $x$ to form $p(x)$ and applied to the term $e$ to form $p(e)$, respectively. Uniform substitutions replace predicate symbols (similarly for function symbols) as long as no $p(e)$ occurs in the scope of a quantifier or modality binding a variable of their replacements other than the occurrences in $e$ itself.

Till then, we need a simple, intuitive, but correct reading of $p(x)$ and $p(e)$ in axiom [:=] and elsewhere. The basic idea is that $p(e)$ stands for the same formula that $p(x)$ does, except that all free occurrences of $x$ are replaced by $e$ and that this substitution requires that $x$ does not occur in $p(x)$ in a quantifier for or a modality with an assignment or with a differential equation for $x$ or a variable of $e$. For example, $[x:=x+y]x \leq y^2 \leftrightarrow x+y \leq y^2$ is an instance of [:=], but $[x:=x+y]\forall y\, x \leq y^2 \leftrightarrow \forall y\, x+y \leq y^2$ is not, because a variable of $x+y$ is bound in $p(x)$. Indeed, $y$ would refer to different variables in both sides of the resulting formula. Free and bound variables will be defined in Sect. 5.6.5.

### 5.3.4 Dynamic Axioms for Differential Equations

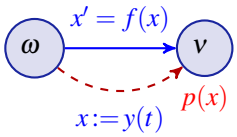
Some of the decompositions of bouncing ball safety property led to safety properties of differential equations, for example (4.29) discussed again at the end of Sect. 5.2.

HPs often involve differential equations. Recall their semantics from Sect. 3.3.2.

**Definition 3.3 (Transition semantics of ODEs).**

$$[\![x' = f(x)\,\&\,Q]\!] = \big\{(\omega,\nu) : \varphi(0) = \omega \text{ except at } x' \text{ and } \varphi(r) = \nu \text{ for a solution } \varphi : [0,r] \to \mathcal{S} \text{ of any duration } r \text{ satisfying } \varphi \models x' = f(x) \wedge Q\big\}$$

where $\varphi \models x' = f(x) \wedge Q$, iff for all times $0 \leq z \leq r$: $\varphi(z) \in [\![x' = f(x) \wedge Q]\!]$ with $\varphi(z)(x') \overset{\text{def}}{=} \frac{\mathsf{d}\varphi(t)(x)}{\mathsf{d}t}(z)$ and $\varphi(z) = \varphi(0)$ except at $x, x'$.




One possible approach of proving properties of differential equations is to work with a solution if one is available (and expressible in dL). Indeed, the first thing you learned about what to do with differential equations is probably to solve them.

**Lemma 5.3 ([′] soundness).** *The solution axiom schema is sound:*

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

*where $y(\cdot)$ solves the symbolic initial value problem $y'(t) = f(y), y(0) = x$.*

Solution $y(\cdot)$ is unique since $f(x)$ is smooth (Theorem 2.2). Given such a solution $y(\cdot)$, continuous evolution along differential equation $x' = f(x)$ 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 axiom [′] and other axioms. Conventional initial value problems (Definition 2.1) are numerical with concrete numbers as initial values, not symbolic variables $x$. 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, since we could hardly solve uncountable many numerical initial value problems. Observe that axiom [′] refers to $y(t)$ at all times $t \geq 0$, which implies that the global solution $y$ needs to exist for all times for axiom [′].

**Note 33 (Discrete vs. continuous dynamics)** *Notice something rather intriguing and peculiar about axiom [′]. It relates a property of a continuous system to a property of a discrete system. The HP on the left-hand side describes a smoothly changing continuous process, while the right-hand side describes an abruptly, instantaneously changing discrete process. Still, their respective properties coincide, thanks to the time quantifier. This is the beginning of an astonishingly intimate relationship of discrete and continuous dynamics [10].*

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 $(x' = f(x)\,\&\,q(x))$. It also does not yet tell us what to do if we cannot solve the differential equation or if the solution is too complicated to be expressed as a term. We will get to those matters in more detail in Part II. But the evolution domain constraint $q(x)$ can be handled directly as well by adding a condition checking that the evolution domain was always true until the point in time of interest:

**Lemma 5.4 ([′] with domains soundness).** *This axiom schema is sound:*

$$[']\quad [x' = f(x)\,\&\,q(x)]p(x) \leftrightarrow \forall t{\geq}0\,\big((\forall 0{\leq}s{\leq}t\; q(y(s))) \rightarrow [x{:=}y(t)]p(x)\big)$$

*where $y(\cdot)$ solves the symbolic initial value problem $y'(t) = f(y), y(0) = x$.*

The effect of the additional constraint on $q(x)$ is to restrict the continuous evolution such that its solution $y(s)$ remains in the evolution domain $q(x)$ at all intermediate times $s \leq t$. This constraint simplifies to *true* if the evolution domain $q(x)$ is *true*, which makes sense, because there are no special constraints on the evolution (other than the differential equations) if the evolution domain is described by *true*, hence is the full state space. In fact, because both axioms from Lemmas 5.3 and 5.4 give

essentially the same result if $q(x)$ is *true*, we gave both the same name $[']$. For axiom schema $[']$, it is important, however, that $x' = f(x) \& q(x)$ is an explicit differential equation, so no $x'$ occurs in $f(x)$ or $q(x)$ (or $p(x)$), because otherwise the notions of solutions become more complicated.

Axiom $[']$ explains the role of evolution domain constraints quite directly. The dL formula $[x' = f(x) \& q(x)]p(x)$ is true iff the postcondition $p(x)$ is true in all states that can be reached by following the solution of the differential equation $x' = f(x)$ *if* that solution has always been in the evolution domain $q(x)$ at all times.

In order to build our way up to using axiom $[']$ on the formula (4.29) that was repeated in Sect. 5.2, first consider a case with only one differential equation:

$$A \rightarrow [x' = v]E$$

Axiom $[']$ swiftly uses the unique solution $x(t) = x + vt$ to reduce this formula to:

$$A \rightarrow \forall t \geq 0\, [x := x + vt]E$$

Let's add the second differential equation, the one for velocity $v$, back in:

$$A \rightarrow [x' = v, v' = -g]p(x) \tag{5.9}$$

Axiom $[']$ simplifies this formula, too, but things become a bit more complicated:

$$A \rightarrow \forall t \geq 0\, [x := x + vt - \frac{g}{2}t^2]p(x) \tag{5.10}$$

First of all, the solution for $x$ became more complex, because the velocity $v$ now keeps changing over time in (5.9). That is perfectly in line with what we have learned about the solutions of differential equations in Chap. 2. Thinking carefully, we notice that (5.10) only is the correct reduction of (5.9) by axiom schema $[']$ if its postcondition $p(x)$ indeed only mentions the position $x$ and not the velocity $v$. In fact, this is the reason why the postcondition was somewhat suggestively phrased as $p(x)$ in (5.9) to indicate that it is a condition on $x$. For a postcondition $p(x,v)$ on $x$ and $v$ or a general postcondition $E$ that can mention any variable, axiom schema $[']$ for differential equation systems solves all differential equations. Thus,

$$A \rightarrow [x' = v, v' = -g]E \tag{5.11}$$

simplifies by axiom schema $[']$ to:

$$A \rightarrow \forall t \geq 0\, [x := x + vt - \frac{g}{2}t^2; v := v - gt]E \tag{5.12}$$

Now, the only remaining issue is that even this insight does not quite take care of handling the bouncing ball's gravity property (4.29), since that is restricted to the evolution domain constraint $x \geq 0$. But adapting these thoughts to the presence of an evolution domain constraint has now become as easy as switching from using axiom schema $[']$ from Lemma 5.3 without evolution domains to, instead, using axiom

schema $[']$ from Lemma 5.4 with evolution domains. Then

$$A \to [x' = v, v' = -g \,\&\, x \geq 0]E$$

simplifies by the evolution domain solution axiom schema $[']$ from Lemma 5.4 to:

$$A \to \forall t{\geq}0 \left( \left( \forall 0{\leq}s{\leq}t \, (x + vs - \frac{g}{2}s^2 \geq 0) \right) \to [x := x + vt - \frac{g}{2}t^2; v := v - gt]E \right)$$

### 5.3.5 Dynamic Axioms for Tests

The bouncing ball formula includes test statements, which we need to handle by an appropriate axiom as well. Recall their semantics from Sect. 3.3.2:

$$[\![?Q]\!] = \{(\omega, \omega) \ : \ \omega \in [\![Q]\!]\}$$


if $\omega \in [\![Q]\!]$ if $\omega \notin [\![Q]\!]$

How could we equivalently rephrase $[?Q]P$, which expresses that $P$ always holds after running the test $?Q$ successfully? Tests do not change the state but impose conditions on the current state. Hence, $P$ is always true after running $?Q$ only if $P$ is already true initially, But there only even is a way of running $?Q$ if $Q$ is true at all.

**Lemma 5.5 ([?] soundness).** *The test axiom is sound:*

$$[?] \quad [?Q]P \leftrightarrow (Q \to P)$$

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

For example, the part $[?x \neq 0](0{\leq}x \wedge x{\leq}H)$ of dL formula (5.8) can be replaced equivalently by the first-order formula $x \neq 0 \to 0{\leq}x \wedge x{\leq}H$ using axiom [?]. After all, both formulas are equivalent according to the equivalence in axiom [?].

### 5.3.6 Dynamic Axioms for Sequential Compositions

For sequential compositions $\alpha;\beta$, Sect. 5.2 proposed the use of an intermediate condition $E$ characterizing all intermediate states between $\alpha$ and $\beta$ by way of Hoare's proof rule following the idea from Fig. 5.2:

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

This proof rule can, indeed, sometimes be useful, but it comes with a significant nuisance compared to the simplicity and elegance of axiom $[\cup]$. When using rule HM; from the desired conclusion to the premises, it does not say how to choose the intermediate condition $E$. Using rule HM; 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 Sect. 4.8.1. If rule HM; were all we have at our disposal for sequential compositions, then we would have to invent a good intermediate condition $E$ for every single sequential composition in the system.

Fortunately, differential dynamic logic provides a better way that we also identify by investigating the dynamical system resulting from $\alpha;\beta$. Recall from Sect. 3.3.2:

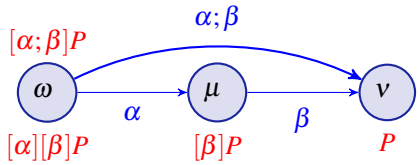
$$[\![\alpha;\beta]\!] = [\![\alpha]\!] \circ [\![\beta]\!] \overset{\text{def}}{=} \{(\omega,\nu) : (\omega,\mu) \in [\![\alpha]\!], (\mu,\nu) \in [\![\beta]\!]\} \tag{5.13}$$

By its semantics, the dL formula $[\alpha;\beta]P$ is true in a state $\omega$ iff $P$ is true in all states that $\alpha;\beta$ can reach according to $[\![\alpha;\beta]\!]$ from $\omega$, i.e. all those states for which $(\omega,\nu) \in [\![\alpha;\beta]\!]$. Which states are those? And how do they relate to the states reachable by $\alpha$ or by $\beta$ alone? 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 (5.13).

Postcondition $P$ has to be true in all states reachable by $\alpha;\beta$ from $\omega$ for $[\alpha;\beta]P$ to be true at $\omega$. By (5.13), those are exactly the states $\nu$ to which we can get by running $\beta$ from an intermediate state $\mu$ to which we have gotten from $\omega$ by running $\alpha$. Thus, for $[\alpha;\beta]P$ to be true at $\omega$, it is necessary that $P$ holds in all states $\nu$ to which we can get by running $\beta$ from an intermediate state $\mu$ to which we get by running $\beta$ from $\omega$. Consequently, $[\alpha;\beta]P$ is only true at $\omega$ if $[\beta]P$ holds in all those intermediate states $\mu$ to which we can get from $\omega$ by running $\alpha$. How do we characterize those states? 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 $[\beta]P$ holds in all states $\mu$ to which we can get to from $\omega$ by running $\alpha$, then that is exactly what truth of dL formula $[\alpha][\beta]P$ at $\omega$ means, because this is the semantics of the modality $[\beta]$:




Consequently, if $[\alpha][\beta]P$ is true in $\omega$, then so is $[\alpha;\beta]P$:

$$\omega \in [\![[\alpha][\beta]P \to [\alpha;\beta]P]\!]$$

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

$$\omega \in [\![[\alpha;\beta]P \to [\alpha][\beta]P]\!]$$

The same argument works for all states $\omega$, so both implications are even valid.

**Lemma 5.6 ([;] soundness).** *The composition axiom is sound:*

$$[;] \quad [\alpha;\beta]P \leftrightarrow [\alpha][\beta]P$$

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

Sequential compositions are proven using nested modalities in axiom [;]. From right to left: If, after all $\alpha$-runs, it is the case that all $\beta$-runs lead to states satisfying $P$ (i.e., $[\alpha][\beta]P$ holds), then all runs of the sequential composition $\alpha;\beta$ lead to states satisfying $P$ (i.e., $[\alpha;\beta]P$ holds), because $\alpha;\beta$ cannot go anywhere but following $\alpha$ through some intermediate state to running $\beta$. The converse implication uses the fact that if after all $\alpha$-runs all $\beta$-runs lead to $P$ (i.e., $[\alpha][\beta]P$), then all runs of $\alpha;\beta$ lead to $P$ (that is, $[\alpha;\beta]P$), because the runs of $\alpha;\beta$ are exactly those that first do any $\alpha$-run, followed by any $\beta$-run. Again, it is crucial that dL is a full logic that considers reachability statements as modal operators, which can be nested, for then both sides in axiom [;] are dL formulas.

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

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

Rule [;]R is easily justified from axiom [;] just by applying the equivalence [;]. Comparing rule [;]R to Hoare's rule HM;, the new rule [;]R is easier to use, because it does not require us to first identify and provide an intermediate condition $E$ like rule

HM; would. It does not branch into two premises, which helps keeping the proof lean. Is there a way of reuniting [;]R with HM; by using the expressive power of dL?

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

Yes, indeed, there is a smart choice for the intermediate condition $E$ that makes HM; behave almost as the more efficient [;]R 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, because all formulas imply themselves, and makes the left premise identical to that of rule [;]R. Consequently, differential dynamic logic internalizes ways of expressing necessary and possible properties of hybrid programs and makes both first-class citizens in the logic. That cuts down on the amount of input that is needed when conducting proofs. Referring back to Fig. 5.2, rule [;]R corresponds to the case of rule HM; when using $[\beta]B$ as the most precise intermediate condition $E$ that implies that all runs of $\beta$ satisfy $B$, see Fig. 5.3.

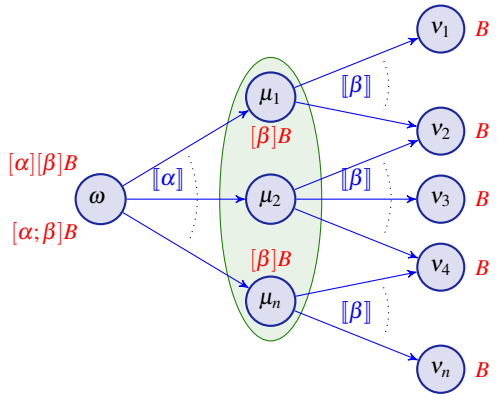



**Fig. 5.3** Illustration of dynamic axiom for sequential compositions

The sequential composition axiom [;] can be used to justify rule [;]R, which is why the latter rule will not be mentioned any more. The axiom can also be used directly to justify replacing any subformula of the form $[\alpha;\beta]P$ by the corresponding $[\alpha][\beta]P$ or vice versa. For example, axiom [;] can be used to simplify formula (5.8) to the following equivalent:

$$E \to [?x=0][v:=-cv](0\leq x \wedge x\leq H) \wedge \big[?x\neq 0\big]\,(0\leq x \wedge x\leq H)$$

This formula can be simplified further by axioms [?] and [:=] to a first-order formula:

$$E \to (x=0 \to 0\leq x \wedge x\leq H) \wedge (x\neq 0 \to 0\leq x \wedge x\leq H)$$

### 5.3.7 Dynamic Axioms for Loops

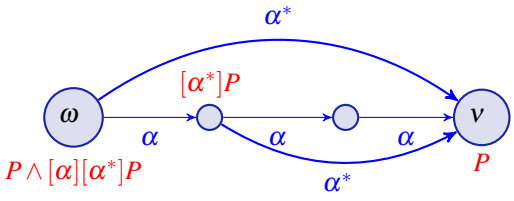
At this point, all HP operators have an axiom except for repetitions. Recall the semantics of loops from Sect. 3.3.2:

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

How could we prove the property $[\alpha^*]P$ of a loop? Is there a way of reducing properties of loops to properties of simpler systems in similar ways as the other axioms of differential dynamic logic?

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

It turns out that repetitions do not support such an equally straightforward decomposition into obviously simpler pieces as the other HP operators did. Why is that? Running a nondeterministic choice $\alpha\cup\beta$ amounts to running either HP $\alpha$ or $\beta$, both of which are smaller than the original $\alpha\cup\beta$. Running a sequential composition $\alpha;\beta$ amounts to first running HP $\alpha$ and then running $\beta$, both of which are smaller. But running a nondeterministic repetition $\alpha^*$ amounts to either not running anything at all or to running $\alpha$ at least one time, so running $\alpha$ once and then subsequently running $\alpha^*$ to run any number of repetitions of $\alpha$ again. The latter hardly is much of a simplification compared to what HP $\alpha^*$ started out with. Nevertheless, there is a way of casting these thoughts into an axiom that reduces dL formula $[\alpha^*]P$ to an equivalent one at least in some sense of the word "reduction".




**Lemma 5.7 ($[^*]$ soundness).** *The iteration axiom is sound:*

$$[^*] \quad [\alpha^*]P \leftrightarrow P \wedge [\alpha][\alpha^*]P$$

Axiom $[^*]$ is the iteration axiom, which partially unwinds loops. It uses the fact that $P$ always holds after repeating $\alpha$ (i.e., $[\alpha^*]P$), if $P$ holds at the beginning (so $P$ holds after zero repetitions), and if, after one run of $\alpha$, $P$ holds after every number of repetitions of $\alpha$, including zero repetitions (i.e., $[\alpha][\alpha^*]P$). So axiom $[^*]$ expresses that $[\alpha^*]P$ holds iff $P$ holds immediately and after one or more repetitions of $\alpha$.

The same axiom $[^*]$ can be used to unwind loops $N \in \mathbb{N}$ times, which corresponds to Bounded Model Checking [1], as Chap. 7 will investigate. 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 how often we unrolled the loop if the formula is actually valid, because we can never find a bug then. Chap. 7 will discuss proof techniques for repetitions based on loop invariants that are not subject to this issue. In particular, axiom $[^*]$ is characteristically different from the other axioms discussed in this chapter. 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.[5]

## 5.4 A Proof of a Short Bouncing Ball

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

$$0 \leq x \land x = H \land v = 0 \land g > 0 \land 1 \geq c \geq 0 \rightarrow \\ \left[\{x' = v, v' = -g \,\&\, x \geq 0\}; (?x = 0; v := -cv \cup ?x \neq 0)\right](0 \leq x \land x \leq H) \quad (5.1^*)$$

Before proceeding, let's modify the hybrid program ever so subtly in two ways so that there's no more evolution domains, just so that we do not yet have to deal with the evolution domains yet. We boldly drop the evolution domain constraint and make up for it by modifying the condition in the second test:

$$0 \leq x \land x = H \land v = 0 \land g > 0 \land 1 \geq c \geq 0 \rightarrow \\ \left[\{x' = v, v' = -g\}; (?x = 0; v := -cv \cup ?x \geq 0)\right](0 \leq x \land x \leq H) \quad (5.14)$$

Hold on, why is that okay? Doesn't our previous investigation say that Quantum could suddenly fall through the cracks in the floor if physics evolves for hours before giving the poor bouncing ball controller a chance to react? To make sure Quantum does not panic in light of this threat, solve Exercise 5.13 to investigate.

To fit things on the page more succinctly, abbreviate

$$A \overset{\text{def}}{\equiv} 0 \leq x \land x = H \land v = 0 \land g > 0 \land 1 \geq c \geq 0$$
$$B(x,v) \overset{\text{def}}{\equiv} 0 \leq x \land x \leq H$$
$$(x'' = -g) \overset{\text{def}}{\equiv} \{x' = v, v' = -g\}$$

With these abbreviations, (5.14) turns into

$$A \rightarrow [x'' = -g; (?x = 0; v := -cv \cup ?x \geq 0)]B(x,v) \quad (5.14^*)$$

[5] With a much more subtle and tricky analysis, it is possible to prove that $[^*]$ still makes sufficient progress [11]. But this is far beyond the scope of this book.

Successively applying the axioms is all it takes to obtain a proof for bouncing balls:

$$\dfrac{A \to \forall t{\geq}0\,\big((H-\frac{g}{2}t^2=0 \to B(H-\frac{g}{2}t^2,-c(-gt))) \land (H-\frac{g}{2}t^2\geq 0 \to B(H-\frac{g}{2}t^2,-gt))\big)}{A \to \forall t{\geq}0\,[x:=H-\frac{g}{2}t^2]\big((x=0 \to B(x,-c(-gt))) \land (x\geq 0 \to B(x,-gt))\big)}\,[:=]$$

$$\dfrac{A \to \forall t{\geq}0\,[x:=H-\frac{g}{2}t^2]\big((x=0 \to B(x,-cv)) \land (x\geq 0 \to B(x,v))\big)}{A \to \forall t{\geq}0\,[x:=H-\frac{g}{2}t^2;v:=-gt]\big((x=0 \to B(x,-cv)) \land (x\geq 0 \to B(x,v))\big)}\,[:=]$$

$$\dfrac{A \to \forall t{\geq}0\,[x:=H-\frac{g}{2}t^2;v:=-gt]\big((x=0 \to B(x,-cv)) \land (x\geq 0 \to B(x,v))\big)}{A \to [x''=-g]\big((x=0 \to B(x,-cv)) \land (x\geq 0 \to B(x,v))\big)}\,[;]$$

$$\dfrac{A \to [x''=-g]\big((x=0 \to B(x,-cv)) \land (x\geq 0 \to B(x,v))\big)}{A \to [x''=-g]\big((x=0 \to [v:=-cv]B(x,v)) \land (x\geq 0 \to B(x,v))\big)}\,[']$$

$$\dfrac{A \to [x''=-g]\big((x=0 \to [v:=-cv]B(x,v)) \land (x\geq 0 \to B(x,v))\big)}{A \to [x''=-g]\big([?x=0][v:=-cv]B(x,v) \land [?x\geq 0]B(x,v)\big)}\,[:=]$$

$$\dfrac{A \to [x''=-g]\big([?x=0][v:=-cv]B(x,v) \land [?x\geq 0]B(x,v)\big)}{A \to [x''=-g]\big([?x=0;v:=-cv]B(x,v) \land [?x\geq 0]B(x,v)\big)}\,[?],[?]$$

$$\dfrac{A \to [x''=-g]\big([?x=0;v:=-cv]B(x,v) \land [?x\geq 0]B(x,v)\big)}{A \to [x''=-g][?x=0;v:=-cv\cup ?x\geq 0]B(x,v)}\,[;]$$

$$\dfrac{A \to [x''=-g][?x=0;v:=-cv\cup ?x\geq 0]B(x,v)}{A \to [x''=-g;(?x=0;v:=-cv\cup ?x\geq 0)]B(x,v)}\,[\cup]$$

The dL axioms indicated on the left justify that the dL formulas in the two adjacent rows are equivalent. Since each step in this proof is justified by using a dL axiom, the conclusion at the very bottom of this derivation is proved if the premise at the very top can be proved, because truth then inherits from the top to the bottom. That premise

$$A \to \forall t{\geq}0\,\big((H-\frac{g}{2}t^2=0 \to B(H-\frac{g}{2}t^2,cgt)) \land (H-\frac{g}{2}t^2\geq 0 \to B(H-\frac{g}{2}t^2,-gt))\big)$$

expands out to a real arithmetic formula when expanding the abbreviations:

$$\begin{aligned}0\leq x\land x=H\land v=0\land g>0\land 1\geq c\geq 0 \to\\ \forall t{\geq}0\,\big((H-\frac{g}{2}t^2=0 \to 0\leq H-\frac{g}{2}t^2\land H-\frac{g}{2}t^2\leq H)\\ \land (H-\frac{g}{2}t^2\geq 0 \to 0\leq H-\frac{g}{2}t^2\land H-\frac{g}{2}t^2\leq H)\big)\end{aligned}$$

In this case, this remaining premise can be easily seen to be valid. The assumption $H-\frac{g}{2}t^2=0\to\ldots$ in the middle line directly implies the first conjunct of the middle line's right-hand side

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

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

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

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

How exactly 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 chapter. For now, we are happy to report that we have just formally verified our very first CPS. We have found a proof of (5.14). Exciting!

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 big steps for mankind start with a small step by someone.

Yet, before we get too carried away in the excitement, we still need to remember that the formula (5.14) that we proved only expresses safety of 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 anyhow without hitting the ground first, which is where the model (5.14) stops prematurely, because it is missing a repetition. So let's put worrying about loops on the agenda for an upcoming chapter (Chap. 7).

Yet, there is one more pressing issue with the proof for the bouncing ball that we derived. It works in a somewhat undisciplined chaotic way, by using dL axioms all over the place. This liberal proof style can be useful for manual proofs and creative shortcuts. Since the dL axioms are sound, even such a liberal proof is a still proof. And liberal proofs could even be very creative. But liberal proofs are also somewhat unfocused and non-systematic, which makes them unreasonable for automation purposes and can also get humans 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 in Chap. 6.

The other thing to observe is that the above proof, however liberal it might have been, already had a lot more structure to it than we have made explicit so far. This structure will be uncovered in the next chapter as well.

## 5.5 Summary

The differential dynamic logic axioms that we have seen in this chapter are summarized in Fig. 5.4. The axioms of dL that are listed in Fig. 5.4 are sound, i.e., valid and so are all their instances. There are further axioms and proof rules of differential dynamic logic that later chapters will examine [9, 10, 12], but the reasoning principles and axioms identified here are fundamental and we will carry them with us throughout the whole textbook.

The equivalence axioms in Fig. 5.4 are primarily meant to be used by replacing the left-hand side (marked in blue) by the structurally simpler right-hand side. With the notable exception of iteration axiom $[^*]$, using these equivalences from left to right decomposes a property of a more complex HP into properties of obviously simpler subprograms.

$[:=]$ $[x:=e]p(x) \leftrightarrow p(e)$

$[?]$ $[?Q]P \leftrightarrow (Q \rightarrow P)$

$[']$ $[x' = f(x)]p(x) \leftrightarrow \forall t \geq 0\, [x:=y(t)]p(x) \quad (y'(t) = f(y))$

$[\cup]$ $[\alpha \cup \beta]P \leftrightarrow [\alpha]P \wedge [\beta]P$

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

$[^*]$ $[\alpha^*]P \leftrightarrow P \wedge [\alpha][\alpha^*]P$

**Fig. 5.4** Summary of sound differential dynamic logic axioms from this chapter

## 5.6 Appendix

This appendix provides additional general axioms for dL that provide conceptual insights and are sometimes useful. These additions are not on the critical path for most CPS, but still advance the general structural understanding.

### *5.6.1 Modal Modus Ponens has Implications on Boxes*

Each of the axioms discussed in this chapter were specific to one operator of dL. But there are also axioms that are common for all hybrid programs $\alpha$.

The next axiom provides a way of shoving implications through box modalities. If we would like to show $[\alpha]Q$ but only know $[\alpha]P$, then what do we need to know about the relationship of $P$ and $Q$ for $[\alpha]P$ to imply $[\alpha]Q$?

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

It would suffice if $P$ were to imply $Q$ unconditionally, because if all runs of $\alpha$ satisfy $P$, then all runs of $\alpha$ also satisfy $Q$ when $P$ implies $Q$. But, in fact, it suffices if $P$ only implies $Q$ after all runs of $\alpha$, not in general.

**Lemma 5.8 (K soundness).** *The modal modus ponens axiom is sound:*

$$\mathrm{K} \quad [\alpha](P \rightarrow Q) \rightarrow ([\alpha]P \rightarrow [\alpha]Q)$$

*Proof.* To show soundness, consider any state $\omega$ in which the assumed left-hand side $[\alpha](P \rightarrow Q)$ of the implication is true, so $\omega \in [\![[\alpha](P \rightarrow Q)]\!]$, and show that the right-hand side is true as well. In order to show that $[\alpha]P \rightarrow [\alpha]Q$ is true in $\omega$, assume that its respective left-hand side is true, so $\omega \in [\![[\alpha]P]\!]$, and show that its right-hand side $[\alpha]Q$ is true. In order to show the latter $\omega \in [\![[\alpha]Q]\!]$, we consider any state $\nu$ with

$(\omega,\nu) \in [\![\alpha]\!]$ and need to show $\nu \in [\![Q]\!]$. Using the assumption $\omega \in [\![[\alpha]P]\!]$, we know that $\nu \in [\![P]\!]$ since $(\omega,\nu) \in [\![\alpha]\!]$. Using the assumption $\omega \in [\![[\alpha](P \to Q)]\!]$, we also know that $\nu \in [\![P \to Q]\!]$ since $(\omega,\nu) \in [\![\alpha]\!]$. Now $\nu \in [\![P]\!]$ and $\nu \in [\![P \to Q]\!]$ imply $\nu \in [\![Q]\!]$, which concludes the proof of $\omega \in [\![[\alpha]Q]\!]$, because $\nu$ was an arbitrary state with $(\omega,\nu) \in [\![\alpha]\!]$. □

Axiom K distributes an implication over a box modality. It can be used to show a more general postcondition $P$ of HP $\alpha$ instead of $[\alpha]Q$ if that more general postcondition $P$ implies the original postcondition $Q$ after all runs of the respective program $\alpha$. For example, $x^2 > 0$ implies $x > 0$ after all runs of $x := x \cdot x$, but not in general, because that particular program happens to imply that $x$ is nonnegative, which is required for $x^2 > 0$ to imply $x > 0$. But showing $x^2 > 0$ after $x := x \cdot x$ would still suffice by K to also show postcondition $x > 0$ for this program. Admittedly, that is no easier in this particular case but may help in others. Of course, if $P \to Q$ is valid, so true in all states, then it is also true after all runs of $\alpha$ (Gödel's generalization rule G explored in Sect. 5.6.3), such that formula $[\alpha]P$ will imply $[\alpha]Q$ by axiom K.

Implications are not the only operator that box modalities distribute over. They also distribute over conjunctions.

**Lemma 5.9 ([]∧ Boxes distribute over conjunctions).** *This axiom is sound:*

$$[]\wedge \quad [\alpha](P \wedge Q) \leftrightarrow [\alpha]P \wedge [\alpha]Q$$

The formula []∧ would be a sound axiom. The only reason why it is not officially adopted as an axiom is because it already follows from axiom K (Exercise 5.16).

### 5.6.2 Vacuous State Change If Nothing Relevant Ever Changes

The axioms discussed in the main part of this chapter were not just specific to one operator of dL, but also, for good reasons, very precise about capturing the exact effect of the respective programs. That is not always required. Sometimes a coarse overapproximation of the effect of a program suffices for an argument. That happens if the variables that a postcondition $p$ depends on are not even modified by program $\alpha$. In that case, $p$ is always true after running $\alpha$ if $p$ was true before, because its truth-value does not change when running a program $\alpha$ that does not change the free variables of $p$. This reasoning is captured in the vacuous axiom V.

**Lemma 5.10 (V soundness).** *The vacuous axiom is sound:*

$$\mathrm{V} \quad p \to [\alpha]p \qquad (FV(p) \cap BV(\alpha) = \emptyset)$$

*where $FV(p) \cap BV(\alpha) = \emptyset$, i.e. no free variable of $p$ is bound (written) in $\alpha$.*

Axiom V makes it possible to prove that truth of a formula $p$ is preserved when running an HP $\alpha$ that does not modify free variables of $p$. For example, if $x > z$ holds initially, then it continues to hold always after running $y' = x$, because that program changes neither $x$ nor $z$ but merely $y$ (and $y'$). Indeed, $x > z \rightarrow [y' = x]\, x > z$ by axiom V. Of course, because $y$ is changed by the HP $y' = x$ and read by postcondition $x > y$, axiom V does not apply for $x > y \rightarrow [y' = x]\, x > y$, which is indeed false in an initial state where $x = 1$ and $y = 0$.

### 5.6.3 Gödel Generalizes Validities into Boxes

Axiom V expresses that truth of a formula is preserved when running HP $\alpha$ if the formula has no free variables modified by $\alpha$. There is also a way to show that a formula $P$ always holds after running an HP $\alpha$ even if that HP modifies free variables of $P$. But that requires more than just the truth of the formula $P$ in the initial state. After all, any arbitrary HP $\alpha$ might very well try to affect the truth-value of a formula if it modifies its free variables.

But if a formula $P$ is not just true initially but valid, so true in all states, then it will certainly still be true always after running any HP $\alpha$ no matter what variables that program modifies. So, if $P$ is valid, then $[\alpha]P$ is valid as well.

**Lemma 5.11 (G soundness).** *The Gödel generalization rule is sound:*

$$\mathrm{G}\ \frac{P}{[\alpha]P}$$

*Proof.* Soundness for proof rules means that validity of the premise implies validity of the conclusion. If premise $P$ is valid, then it is true in all states. Then the conclusion $[\alpha]P$ is valid, because it is true in any state $\omega$, since $\nu \in [\![P]\!]$ for all states $\nu$ including all states $\nu$ for which $(\omega, \nu) \in [\![\alpha]\!]$. □

For example, $x^2 \geq 0$ is valid, so by G is also true after any HP, which implies validity of $[x' = x^3 + 1]\, x^2 \geq 0$. The truth of the postcondition $x^2 \geq 0$ simply does not depend on the HP $x' = x^3 + 1$ at all, because the postcondition is true in any state.

### 5.6.4 Monotonicity of Postconditions

Gödel's generalization rule G is a global version of the axiom V in the sense that rule G expresses that validity of $P$ in all states is preserved after a $[\alpha]$ modality while axiom V expresses that truth of $p$ (in the initial state) is preserved after a $[\alpha]$ modality if $\alpha$ does not modify variables of $p$.

Similarly, the proof rule M[·] is a global version of axiom K in the sense that rule M[·] uses validity of an implication $P \to Q$, while axiom K merely requires truth of $P \to Q$ (albeit after all runs of $\alpha$). Rule M[·] is the monotonicity rule expressing that if $P$ implies $Q$ so $Q$ is true in more states than $P$ (premise) then $[\alpha]P$ also implies $[\alpha]Q$ so $[\alpha]Q$ is true in more states than $[\alpha]P$ (conclusion).

**Lemma 5.12 (M[·] soundness).** *The monotonicity rules are sound:*

$$\text{M[·]}\ \frac{P \to Q}{[\alpha]P \to [\alpha]Q} \qquad \text{M}\ \frac{P \to Q}{\langle\alpha\rangle P \to \langle\alpha\rangle Q}$$

*Proof.* If the premise $P \to Q$ is valid, then $Q$ is true in every state that $P$ is true in, because $[\![P]\!] \subseteq [\![Q]\!]$. Consequently, in every state in which $[\alpha]P$ is true, $[\alpha]Q$ is true, too, because if all runs of $\alpha$ lead to states satisfying $P$ then they also all lead to states satisfying $Q$. In any state $\omega$ in which $\omega \in [\![[\alpha]P]\!]$, it is the case that $\nu \in [\![P]\!]$, hence also $\nu \in [\![Q]\!]$, for all states $\nu$ with $(\omega,\nu) \in [\![\alpha]\!]$. That implies $\omega \in [\![[\alpha]Q]\!]$. Soundness of rule M is similar. As an alternative way of justifying its soundness, rule M[·] can be derived from Gödel's generalization rule G using axiom K (Exercise 5.18). □

For example, $[x := 1; x' = x^2 + 2x^4]x^3 \geq x^2$ is difficult to prove directly. But proving $[x := 1; x' = x^2 + 2x^4]x \geq 1$ will turn out to be surprisingly easy in Part II based on the intuition that the right-hand side $x^2 + 2x^4$ of the ODE will onl makes $x$ increase above 1 never below 1. And the postcondition $x \geq 1$ easily implies the original postcondition $x^3 \geq x^2$. This example shows how monotonicity rule M[·] can simplify proofs by reducing the proof of a difficult formula $[x := 1; x' = x^2 + 2x^4]x^3 \geq x^2$ to the proof of a formula $[x := 1; x' = x^2 + 2x^4]x \geq 1$ with a different postcondition $x \geq 1$ , which implies the original postcondition $x^3 \geq x^2$.

Observe how this proof with monotonicity rule M[·] is characteristically different than what Gödel's generalization rule G gives. Gödel's G gives us a proof for a postcondition that is true in all states and, thus, not a particularly informative postcondition specific to the particular HP. Monotonicity rule M gives us an insightful postcondition $x^3 \geq x^2$ from a proof of another postcondition $x \geq 1$ that is informative, because it is always true after HP $x := 1; x' = x^2 + 2x^4$ but not after HP $x' = -1$.

Unlike Gödel's generalization rule G, monotonicity rule M[·] has a direct counterpart, rule M, for diamond modalities. A diamond counterpart for rule G does not exist, because even if $P$ is valid, that does not mean that $\langle\alpha\rangle P$ is valid, because there may not be a way of running $\alpha$ to any final state at all. For example, $\langle ?\mathit{false}\rangle \mathit{true}$ is not valid, because there is no way of satisfying its test $?\mathit{false}$. That problem does not come up in the diamond montononicity rule M, because its conclusion already assumes $\langle\alpha\rangle P$ so there is a way of running HP $\alpha$.

### 5.6.5 Of Free and Bound Variables

The vacuous axiom V from Sect. 5.6.2 and our understanding of the assignment axiom [:=] from Sect. 5.3.3 used a concept of free and bound variables. Free variables are all those that the value or semantics of an expression depends on. Bound variables are all those that can change their value during a HP.

For example, the free variables of term $x\cdot 2+y\cdot y$ are $\{x,y\}$ but not $z$, because that term does not even mention $z$ so its value depends only on the values of $x,y$ but not on that of $z$. Likewise, the free variables of HP $a:=-b;\ x'=v,v'=a$ are $\{b,x,v\}$ but not $a$, because its value is only read after $a$ is written to. The bound variables of that HP are $a,x,v$ (and also $x',v'$ if you look closely because both change their value by virtue of what it means to solve a differential equation) but not $b$, because $b$ is only read but never written to.

**Definition 5.3 (Static semantics).** The *static semantics* defines the *free variables*, which are all variables that the value of an expression depends on. It also defines the *bound variables*, which can change their value during the evaluation of an expression.

$$
\begin{aligned}
\mathrm{FV}(e) &= \bigcup\{x\in\mathscr{V} : \text{there are } \omega=\tilde{\omega} \text{ on } \{x\}^{\complement} \text{ such that } \omega[\![e]\!]\neq\tilde{\omega}[\![e]\!]\}\\
\mathrm{FV}(P) &= \bigcup\{x\in\mathscr{V} : \text{there are } \omega=\tilde{\omega} \text{ on } \{x\}^{\complement} \text{ such that } \omega\in[\![P]\!]\not\ni\tilde{\omega}\}\\
\mathrm{FV}(\alpha) &= \bigcup\{x\in\mathscr{V} : \text{there are } \omega,\tilde{\omega},\nu \text{ with } \omega=\tilde{\omega} \text{ on } \{x\}^{\complement} \text{ and } (\omega,\nu)\in[\![\alpha]\!]\\
&\qquad\qquad \text{but no } \tilde{\nu} \text{ with } \nu=\tilde{\nu} \text{ on } \{x\}^{\complement} \text{ such that } (\tilde{\omega},\tilde{\nu})\in[\![\alpha]\!]\}\\
\mathrm{BV}(\alpha) &= \bigcup\{x\in\mathscr{V} : \text{there are } (\omega,\nu)\in[\![\alpha]\!] \text{ such that } \omega(x)\neq\nu(x)\}
\end{aligned}
$$

A variable $x\in\mathscr{V}$ is a free variable of a term $e$ if its value depends on it, i.e. there are two states $\omega$ and $\tilde{\omega}$ that only differ in the value of variable $x$ (so they agree on the complement $\{x\}^{\complement}$ of the singleton set $\{x\}$) where $e$ has different values in the two states $\omega$ and $\tilde{\omega}$. Likewise $x\in\mathscr{V}$ is a free variable of a formula $P$ if there are two different states agreeing on $\{x\}^{\complement}$ such that $P$ is true in one but false in the other. For a HP $\alpha$, a variable $x\in\mathscr{V}$ is a free variable if, from two different states agreeing on $\{x\}^{\complement}$, there is a run of $\alpha$ in one of them but no corresponding run of $\alpha$ from the other (with merely a different value of $x$). A variable $x\in\mathscr{V}$ is a bound variable of HP $\alpha$ if there is a run of $\alpha$ that changes the value of $x$.

The dynamic semantics gives a precise meaning to HPs (Chap. 3) and dL formulas (Chap. 4) but is inaccessible for effective reasoning purposes (unlike the syntactic axioms that this chapter provides). By contrast, the static semantics of dL and HPs defines only simple aspects of the dynamics concerning the variable usage that can also be read off more directly from the syntactic structure without running the programs or evaluating their dynamical effects, as we will see next.

### 5.6.6 Free and Bound Variable Analysis

Computing the set of free and bound variables according to Definition 5.3 precisely is impossible, because every nontrivial property of programs is undecidable [14]. For example, it takes at least a moment's thought to see that, despite first appearances, $x$ is not actually read and $y$ not actually written to in the HP

$$z:=0;(y:=x+1;z'=1;?z<0\cup z:=z+1)$$

Fortunately, any supersets of the set of free and bound variables will work correctly, for example for the conditions for the vacuous axiom V in Sect. 5.6.2. Such supersets of the set of free variables and bound variables are quite easily computable directly from the syntax. The easiest approach would be to simply take the set of all variables that are ever read anywhere, which is a simple superset of the free variables. The set of variables that are ever written to is an obvious superset of the bound variables. Those may yield too many variables, because $a \in \mathscr{V}$ is not actually free in the following HP $a:=-b;\ x'=v,v'=a$ even if it is read somewhere, because it is written to first, so receives its value during the computation.

With more thought, more precise sound (super-)sets of free and bound variables can be computed equally easily by keeping track of variables that are written before they are read [12], but they will inevitably still be overapproximations. Bound variables $x$ of a formula are those that are bound by $\forall x$ or $\exists x$, but also those that are bound by modalities such as $[x:=5y]$ or $\langle x'=1\rangle$ or $[x:=1\cup x'=1]$ or $[x:=1\cup ?true]$ because of the assignment to $x$ or differential equation for $x$ they contain. The scope of the bound variable $x$ is limited to the quantified formula or to the postcondition and remaining program of the modality.

**Definition 5.4 (Bound variable).** The set $\mathrm{BV}(P)\subseteq\mathscr{V}$ of (syntactically) *bound variables* of dL formula $P$ is defined inductively as:

$$\begin{aligned}
\mathrm{BV}(e\geq\tilde{e}) &= \emptyset && \text{accordingly for } =,>,\leq,<\\
\mathrm{BV}(\neg P) &= \mathrm{BV}(P)\\
\mathrm{BV}(P\wedge Q) &= \mathrm{BV}(P)\cup\mathrm{BV}(Q) && \text{accordingly for } \vee,\rightarrow,\leftrightarrow\\
\mathrm{BV}(\forall x\,P) = \mathrm{BV}(\exists x\,P) &= \{x\}\cup\mathrm{BV}(P)\\
\mathrm{BV}([\alpha]P) = \mathrm{BV}(\langle\alpha\rangle P) &= \mathrm{BV}(\alpha)\cup\mathrm{BV}(P)
\end{aligned}$$

The set $\mathrm{BV}(\alpha)\subseteq\mathscr{V}$ of (syntactically) *bound variables* of HP $\alpha$, i.e. all those that may potentially be written to, is defined inductively as:

$$\begin{aligned} \mathrm{BV}(x:=e) &= \{x\} \\ \mathrm{BV}(?Q) &= \emptyset \\ \mathrm{BV}(x'=f(x)\,\&\,Q) &= \{x,x'\} \\ \mathrm{BV}(\alpha\cup\beta) = \mathrm{BV}(\alpha;\beta) &= \mathrm{BV}(\alpha)\cup\mathrm{BV}(\beta) \\ \mathrm{BV}(\alpha^*) &= \mathrm{BV}(\alpha) \end{aligned}$$

Bound by a differential equation $x'=f(x)$ are $x$ and $x'$, as both change their value.

The free variables of a quantified formula are defined by removing its bound variable as $\mathrm{FV}(\forall x P) = \mathrm{FV}(P)\setminus\{x\}$, since all occurrences of $x$ in $P$ are bound. The bound variables of a program in a modality act in a similar way, except that the program itself may read variables during the computation, so its free variables need to be taken into account. By analogy to the quantifier case, it is often suspected that $\mathrm{FV}([\alpha]P)$ could be defined as $\mathrm{FV}(\alpha)\cup(\mathrm{FV}(P)\setminus\mathrm{BV}(\alpha))$. But that would be unsound, because $[x:=1\cup y:=2]x\geq 1$ would have no free variables then, contradicting the fact that its truth-value depends on the initial value of $x$. The reason is that $x$ is a bound variable of that program, but only written to on some but not on all paths. So the initial value of $x$ may be needed to evaluate the truth of the postcondition $x\geq 1$ on some execution paths. If a variable is must-bound, so written to on all paths of the program, however, it can safely be removed from the free variables of the postcondition. The static semantics, thus, first defines the subset of variables that are must-bound ($\mathrm{MBV}(\alpha)$), so must be written to on all execution paths of $\alpha$.

**Definition 5.5 (Must-bound variable).** The set $\mathrm{MBV}(\alpha)\subseteq\mathrm{BV}(\alpha)\subseteq\mathscr{V}$ of (syntactically) *must-bound variables* of HP $\alpha$, i.e. all those that must be written to on all paths of $\alpha$, is defined inductively as:

$$\begin{aligned} \mathrm{MBV}(x:=e) &= \{x\} \\ \mathrm{MBV}(?Q) &= \emptyset \\ \mathrm{MBV}(x'=f(x)\,\&\,Q) &= \{x,x'\} \\ \mathrm{MBV}(\alpha\cup\beta) &= \mathrm{MBV}(\alpha)\cap\mathrm{MBV}(\beta) \\ \mathrm{MBV}(\alpha;\beta) &= \mathrm{MBV}(\alpha)\cup\mathrm{MBV}(\beta) \\ \mathrm{MBV}(\alpha^*) &= \emptyset \end{aligned}$$

Finally, the static semantics defines which variables are free so may be read. The definition of free variables is simultaneously inductive for formulas ($\mathrm{FV}(P)$) and programs ($\mathrm{FV}(\alpha)$) owing to their mutually recursive syntactic structure. The set $\mathrm{FV}(e)\subseteq\mathscr{V}$ of (syntactically) *free variables* of term $e$ is simply the set of those that occur in $e$, at least till Chap. 10. The differential terms $(e)'$ that will be introduced in Chap. 10 have as free variables also all differential symbols that are primed versions of the free variables of $e$, so $\mathrm{FV}((e)')=\mathrm{FV}(e)\cup\mathrm{FV}(e)'$, so that, for example $\mathrm{FV}((x+y)')=\{x,x',y,y'\}$.

**Definition 5.6 (Free variable).** The set $\mathrm{FV}(P)$ of (syntactically) *free variables* of dL formula $P$, i.e. all that occur in $P$ outside the scope of quantifiers or modalities

binding it, is defined inductively as:

$$\begin{aligned}
\mathrm{FV}(e \geq \tilde{e}) &= \mathrm{FV}(e) \cup \mathrm{FV}(\tilde{e}) && \text{accordingly for } =, \leq \\
\mathrm{FV}(\neg P) &= \mathrm{FV}(P) \\
\mathrm{FV}(P \wedge Q) &= \mathrm{FV}(P) \cup \mathrm{FV}(Q) \\
\mathrm{FV}(\forall x P) = \mathrm{FV}(\exists x P) &= \mathrm{FV}(P) \setminus \{x\} \\
\mathrm{FV}([\alpha]P) = \mathrm{FV}(\langle\alpha\rangle P) &= \mathrm{FV}(\alpha) \cup (\mathrm{FV}(P) \setminus \mathrm{MBV}(\alpha))
\end{aligned}$$

The set $\mathrm{FV}(\alpha) \subseteq \mathscr{V}$ of (syntactically) *free variables* of HP $\alpha$, i.e. all those that may potentially be read, is defined inductively as:

$$\begin{aligned}
\mathrm{FV}(x := e) &= \mathrm{FV}(e) \\
\mathrm{FV}(?Q) &= \mathrm{FV}(Q) \\
\mathrm{FV}(x' = f(x) \,\&\, Q) &= \{x\} \cup \mathrm{FV}(f(x)) \cup \mathrm{FV}(Q) \\
\mathrm{FV}(\alpha \cup \beta) &= \mathrm{FV}(\alpha) \cup \mathrm{FV}(\beta) \\
\mathrm{FV}(\alpha;\beta) &= \mathrm{FV}(\alpha) \cup (\mathrm{FV}(\beta) \setminus \mathrm{MBV}(\alpha)) \\
\mathrm{FV}(\alpha^*) &= \mathrm{FV}(\alpha)
\end{aligned}$$

The *variables* of dL formula $P$, whether free or bound, are $\mathrm{V}(P) = \mathrm{FV}(P) \cup \mathrm{BV}(P)$. The *variables* of HP $\alpha$, whether free or bound, are $\mathrm{V}(\alpha) = \mathrm{FV}(\alpha) \cup \mathrm{BV}(\alpha)$.

Both $x$ and $x'$ are bound in $x' = f(x) \,\&\, Q$ since both change their value, but only $x$ is added to the free variables, because the behavior of the differential equation can only depend on the initial value of $x$, not of that of $x'$.

These syntactic definitions are correct [12], i.e. they compute supersets of the semantic definitions of the static semantics from Definition 5.3 (Exercise 5.20). Of course [14], syntactic computations may give bigger sets, e.g., $\mathrm{FV}(x^2 - x^2) = \{x\}$ even if the value of this unsimplified term really depends on no variable, and $\mathrm{BV}(x := x) = \{x\}$ even if this change is a no-op.

## Exercises

**5.1 (Necessity of assumptions).** Identify which of the assumptions of (5.14) are actually required for the proof of (5.14). Which formulas could we have dropped from $0 \leq x \wedge x = H \wedge v = 0 \wedge g > 0 \wedge 1 \geq c \geq 0$ and still be able to prove

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

**5.2.** Show that the following axiom would be sound

$$Q \wedge P \rightarrow [?Q]P$$

**5.3.** Would the following be a sound axiom? Prove or disprove.

$$[x:=e]P \leftrightarrow \langle x:=e\rangle P$$

**5.4.** Axiom $[']$ made it possible to make use of solutions of differential equations of the form $[x' = f(x)]P$. Develop possible axioms for differential equations with evolution domains. That is, develop an axiom for $[x' = f(x)\,\&\,Q]P$. As in axiom $[']$, you can assume to have a unique solution for the corresponding symbolic initial value problem.

**5.5 (Solutions at the endpoint).** Prove that the following axiom is sound when $y$ is the unique global solution:

$$\forall t{\geq}0\,[x:=y(t)](q(x) \rightarrow p(x)) \rightarrow [x' = f(x)\,\&\,q(x)]p(x) \qquad (y'(t) = f(y))$$

**5.6.** When misplacing the parentheses in axiom $[']$ to obtain the following formula, would it be a sound axiom, too? Prove or disprove.

$$[x' = f(x)\,\&\,q(x)]p(x) \leftrightarrow \forall t{\geq}0\,\forall 0{\leq}s{\leq}t\,\big(q(y(s)) \rightarrow [x:=y(t)]p(x)\big) \quad (y'(t) = f(y))$$

As in axiom $[']$, you can assume $y$ to be the unique global solution for the corresponding symbolic initial value problem.

**5.7 (Solutions of systems).** The dL formula (5.12) was suggested as a result of using axiom schema $[']$ on dL formula (5.11). Would the converse order of solutions have worked as well to reduce (5.11) to the following formula instead?

$$A \rightarrow \forall t{\geq}0\,[v:=v-gt;\,x:=x+vt-\frac{g}{2}t^2]E$$

**5.8.** Would either of the following be a sound axiom? Prove or disprove.

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

**5.9.** Would the following be a useful replacement for the $[^*]$ axiom? Is it sound? Is it useful?

$$[\alpha^*]P \leftrightarrow P \wedge [\alpha^*]P$$

**5.10.** Would the following be a useful replacement for the $[^*]$ axiom? Is it sound? Is it useful?

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

**5.11 (Soundness of dynamic axioms).** All axioms need to be proved to be sound. This textbook only did a proper proof for some axioms, because proofs are published elsewhere [10]. Turn the informal arguments for the other axioms into proper soundness proofs using the semantics of dL formulas.

**5.12 (Diamond axioms).** This chapter identified axioms for all formulas of the form $[\alpha]P$ but none for formulas of the form $\langle\alpha\rangle P$. Identify and justify these missing axioms. Explain how they relate to the ones given in Fig. 5.4. Find out whether you made a mistake by proving them sound.

**5.13 (Give bouncing ball back its evolution domain).** Explain why the subtle transformation from (5.1) to (5.14) was okay *in this particular case*.

**5.14 (Nondeterministic assignments).** Suppose a new statement $x := *$ for nondeterministic assignment is added to HPs, which assigns an arbitrary real number to the variable $x$. The new syntactic construct of nondeterministic assignments needs a semantics to become meaningful:

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

Develop an axiom for $[x := *]P$ and an axiom for $\langle x := *\rangle P$, which rephrase both in terms of simpler logical connectives and prove soundness of these axioms.

**5.15 (Differential assignments).** Hybrid programs allow discrete assignments $x := e$ to any variable $x \in \mathcal{V}$. In Part II, we will discover the important role that differential symbols $x'$ play. Part II will end up considering differential symbols $x'$ as variables and allow discrete assignments $x' := e$ to differential symbols $x'$ that instantaneously change the value of $x'$ to that of $e$. Develop a semantics for these *differential assignments* $x' := e$. Develop an axiom for $[x' := e]p(x)$ and an axiom for $\langle x' := e\rangle p(x)$, and prove soundness of these axioms.

**5.16 (K knows that boxes distribute over conjunctions).** Show that axiom $[]\wedge$ that distributes boxes over conjunctions

$$[]\wedge \quad [\alpha](P \wedge Q) \leftrightarrow [\alpha]P \wedge [\alpha]Q$$

can be derived from the modal modus ponens axiom K from Sect. 5.6.1:

$$\text{K} \quad [\alpha](P \to Q) \to ([\alpha]P \to [\alpha]Q)$$

**5.17.** Axioms K and $[]\wedge$ show how box modalities distribute over conjunctions and implications. Do they also distribute over other logical connectives? Which of the following formulas are valid?

$$\begin{aligned} [\alpha](P \vee Q) &\to [\alpha]P \vee [\alpha]Q \\ [\alpha]P \vee [\alpha]Q &\to [\alpha](P \vee Q) \\ [\alpha](P \leftrightarrow Q) &\to ([\alpha]P \leftrightarrow [\alpha]Q) \\ ([\alpha]P \leftrightarrow [\alpha]Q) &\to [\alpha](P \leftrightarrow Q) \\ [\alpha]\neg P &\to \neg[\alpha]P \\ \neg[\alpha]P &\to [\alpha]\neg P \end{aligned}$$

How about diamond modalities? Which of the following are valid?

$$\langle\alpha\rangle(P \to Q) \to (\langle\alpha\rangle P \to \langle\alpha\rangle Q)$$
$$(\langle\alpha\rangle P \to \langle\alpha\rangle Q) \to \langle\alpha\rangle(P \to Q)$$
$$\langle\alpha\rangle(P \wedge Q) \to \langle\alpha\rangle P \wedge \langle\alpha\rangle Q$$
$$\langle\alpha\rangle P \wedge \langle\alpha\rangle Q \to \langle\alpha\rangle(P \wedge Q)$$
$$\langle\alpha\rangle(P \vee Q) \to \langle\alpha\rangle P \vee \langle\alpha\rangle Q$$
$$\langle\alpha\rangle P \vee \langle\alpha\rangle Q \to \langle\alpha\rangle(P \vee Q)$$
$$\langle\alpha\rangle(P \leftrightarrow Q) \to (\langle\alpha\rangle P \leftrightarrow \langle\alpha\rangle Q)$$
$$(\langle\alpha\rangle P \leftrightarrow \langle\alpha\rangle Q) \to \langle\alpha\rangle(P \leftrightarrow Q)$$
$$\langle\alpha\rangle\neg P \to \neg\langle\alpha\rangle P$$
$$\neg\langle\alpha\rangle P \to \langle\alpha\rangle\neg P$$

**5.18 (Monotonicity rule).** Prove that the monotonicity rule M$[\cdot]$ (Sect. 5.6.4) derives from Kripke's modal modus ponens axiom K (Sect. 5.6.1) and Gödel's generalization rule G (Sect. 5.6.3). That is, find a proof of the conclusion $[\alpha]P \to [\alpha]Q$ of M$[\cdot]$ assuming that you already have a proof of its premise $P \to Q$.

**5.19 ($\langle\cdot\rangle$ monotonicity rule).** Give a direct semantic soundness proof for the monotonicity rule M for the diamond modality in the same style that soundness was proved for the monotonicity rule M$[\cdot]$ for the box modality (Sect. 5.6.4). Then propose a diamond modality version of axiom K that would make it possible to derive M similarly to how rule M$[\cdot]$ was derived from axiom K and rule G in Exercise 5.18. Prove soundness of that newly proposed axiom.

**5.20 (***).** Show that the definitions of (syntactically) free and bound variables from Sect. 5.6.6 are correct, i.e. they are supersets of the semantic definitions of free and bound variables from Sect. 5.6.5. Under what circumstances are they proper supersets, i.e. contain additional variables?

## *References*


1. Clarke, E. M., Biere, A., Raimi, R. & Zhu, Y. Bounded Model Checking Using Satisfiability Solving. *Form. Methods Syst. Des.* **19,** 7–34 (2001).
2. Ghorbal, K. & Platzer, A. *Characterizing Algebraic Invariants by Differential Radical Invariants* in *TACAS* (eds Ábrahám, E. & Havelund, K.) **8413** (Springer, 2014), 279–294. doi:10.1007/978-3-642-54862-8_19.
3. Hoare, C. A. R. An Axiomatic Basis for Computer Programming. *Commun. ACM* **12,** 576–580 (1969).
4. *Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25–28, 2012* (IEEE, 2012).
5. Mitsch, S. & Platzer, A. ModelPlex: Verified Runtime Validation of Verified Cyber-Physical System Models. *Form. Methods Syst. Des.* **49.** Special issue of selected papers from RV'14, 33–74 (2016).

6. Platzer, A. Differential Dynamic Logic for Hybrid Systems. *J. Autom. Reas.* **41,** 143–189 (2008).
7. Platzer, A. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics* doi:10.1007/978-3-642-14509-4 (Springer, Heidelberg, 2010).
8. Platzer, A. A Complete Axiomatization of Quantified Differential Dynamic Logic for Distributed Hybrid Systems. *Log. Meth. Comput. Sci.* **8.** Special issue for selected papers from CSL'10, 1–44 (2012).
9. Platzer, A. *Logics of Dynamical Systems* in *LICS* (IEEE, 2012), 13–24. doi:10.1109/LICS.2012.13.
10. Platzer, A. *The Complete Proof Theory of Hybrid Systems* in *LICS* (IEEE, 2012), 541–550. doi:10.1109/LICS.2012.64.
11. Platzer, A. Differential Game Logic. *ACM Trans. Comput. Log.* **17,** 1:1–1:51 (2015).
12. Platzer, A. A Complete Uniform Substitution Calculus for Differential Dynamic Logic. *J. Autom. Reas.* doi:10.1007/s10817-016-9385-1 (2016).
13. Platzer, A. & Clarke, E. M. Computing Differential Invariants of Hybrid Systems as Fixedpoints. *Form. Methods Syst. Des.* **35.** Special issue for selected papers from CAV'08, 98–120 (2009).
14. Rice, H. G. Classes of recursively enumerable sets and their decision problems. *Trans. AMS* **74,** 358–366 (1953).
15. Seto, D., Krogh, B., Sha, L. & Chutinan, A. *The Simplex architecture for safe online control system upgrades* in *ACC* **6** (1998), 3504–3508.