Chapter 18
Axioms & Uniform Substitutions

Synopsis This chapter explores a succinct approach for soundly implementing rigorous reasoning for hybrid systems. Unlike previous chapters, this chapter is not concerned with identifying new reasoning principles for cyber-physical systems, but, rather, focuses on how they can best be implemented correctly. Uniform substitutions are identified as a simple concept based upon which differential dynamic logic proof systems can be implemented quite easily. Uniform substitutions uniformly instantiate predicate symbols by formulas. Since all reasoning can be reduced to finding the appropriate sequence of uniform substitutions, this makes it possible to implement theorem provers with a small soundness-critical core.

18.1 Introduction

The logic and reasoning principles for hybrid systems (Part I), differential equations (Part II), and hybrid games (Part III) identified in previous chapters are conducive to quite simple correctness arguments. Proof principles decouple the question of what a correct argument is from the question of how to find it. Soundness even of the biggest and most complicated proofs directly follows from the soundness of each of the proof steps. Every proof step uses one of a small set of dL axioms and proof rules, which can each be proved sound individually quite easily. The transfer of soundness was already rooted in Definition 6.2 on p. 179, which defined a proof rule to be sound iff the validity of all premises implies the validity of the conclusion. For axioms, Definition 5.1 on p. 146 defined an axiom to be sound iff all its instances are valid. Since proofs only consist of axioms composed with proof rules, this implies that the conclusion of every (completed) proof is valid.

The remaining challenge for soundness of proofs is to ensure that all axioms and proof rules are also implemented correctly in a theorem prover. The primary obstacle is that the reasoning principles identified so far were considered as *axiom schemata*, i.e., they stand for an infinite family of formulas of the same shape. That is easily said, but still needs some form of implementation. Moreover, a fair number of the

A. Platzer, *Logical Foundations of Cyber-Physical Systems*,
https://doi.org/10.1007/978-3-319-63588-0_18

axiom schemata have soundness-critical side conditions that need to be respected to guarantee soundness. That these soundness-critical side conditions cannot be elided is most obvious in the vacuous axiom schema from Lemma 5.11:

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

Of course, every use of axiom schema V needs to ensure that the same formula p is used in the precondition and the postcondition. But without checking that no free variable of p is written to in the hybrid program α, it would be quite unsound to conclude that p always holds after running HP α if p was true initially. After all, if α changes a variable that p reads, its truth-value may change. It is only thanks to this side condition that the following invalid formula is not provable by axiom V:

$$x \geq 0 \rightarrow [x' = -5]\, x \geq 0$$

The differential equation solution axiom schema from Lemma 5.3 has even more complicated side conditions:

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

The soundness-critical side conditions for axiom schema $[']$ are:

1. The variable t needs to be fresh and cannot have occurred already, because it is supposed to represent the independent variable for time.
2. The function of time $y(\cdot)$ needs to solve the differential equation $y(t)' = f(y(t))$ and needs to be defined at all times that the quantifier for t quantifies over, because the continuous dynamics of the differential equation can only be equivalently replaced by a discrete assignment when $y(\cdot)$ is the correct solution of the differential equation.
3. The solution $y(\cdot)$ needs to solve the symbolic initial value condition $y(0) = x$ for the variable x, because we usually do not have a specific numerical initial value when using axiom schema $[']$.
4. The solution $y(\cdot)$ needs to cover all solutions parametrically, e.g., when the solution has different shapes for different choices of the initial value x.
5. The postcondition $p(x)$ cannot have differential symbol x' as a free variable, because it receives the value $f(x)$ after the differential equation but retains the initial value after the discrete assignment to x.[1]

A correct implementation of axiom schema $[']$, thus, amounts to an algorithm accepting every formula of this shape after checking all the required side conditions. Fortunately, Part II already provided a substantially more elegant way of proving properties of differential equations by induction that also makes the solution axiom schema $[']$ superfluous [8], because it can be replaced by appropriate differential cuts to augment the evolution domain with the solution after a suitable differential ghost has shifted the dynamics into the time domain $t' = 1$ (Chap. 12). But the fact

[1] Of course, this is easily fixed by adding an assignment $x' := f(x)$ after the assignment to x.

remains that axiom schemata have a tendency to require a somewhat unwieldy set of side conditions that are soundness-critical and, thus, need to be enforced for every reasoning step. Compared to verification algorithms that do not even benefit from a similar logical foundation, it is still substantially easier to devise correct implementations of individual axiom schemata and then glue them together with correct implementations of proof rules. But this chapter will find a more straightforward way that is even easier to get correct.

The primary observation to make this happen comes from a shift in perspective that distinguishes between axioms and axiom schemata. An *axiom* is a single valid formula that is adopted as a basis for reasoning in a proof calculus. An *axiom schema* stands for an infinite family of formulas of the same shape (subject to the required side conditions) and, thus, needs to be implemented with an algorithm. Implementing an axiom is trivial, because an axiom is just a single formula in the object logic. The only downside is that the only formula that an axiom enables us to prove is literally that formula in verbatim, which we are rarely interested in proving.

Consequently, the missing element for a reasoning system that is based on axioms is a mechanism for instantiating them. Church's uniform substitutions [2] provide such a mechanism for first-order logic. Uniform substitutions make it possible to instantiate predicate symbols by formulas and check the required conditions to ensure that that instantiation is sound. Generalizing uniform substitutions from first-order logic to differential dynamic logic leads to the corresponding mechanism to implement flexible dL proving parsimoniously with uniform substitution as essentially the only proof rule [7, 8].

Differential dynamic logic provides sound reasoning principles. Uniform substitutions make it easy to implement them correctly. Uniform substitutions are the secret for simple sound hybrid systems provers such as KeYmaera X [3]. This chapter has a major impact, enabling the 1,700 lines of soundness-critical code in KeYmaera X compared to the 66,000 lines of soundness-critical code[2] in its predecessor KeYmaera [9], which implements a schematic sequent calculus for dL [5].

The most important learning goals of this chapter are:

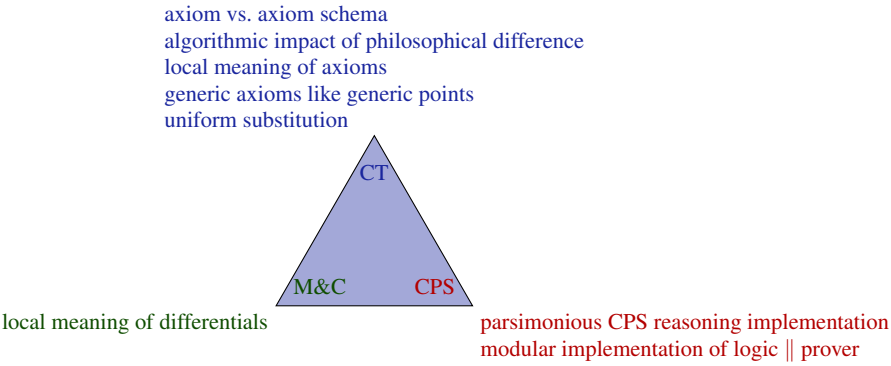
Modeling and Control: We will eventually see how the shift in perspective to axioms gives us an opportunity to reflect on the significance of the local meaning of differentials in hybrid systems.

Computational Thinking: This chapter investigates the relationship and fundamental difference of axioms versus axiom schemata. This philosophical distinction leads to a significant algorithmic impact on the style of implementing hybrid systems reasoning. This chapter explores the local meaning of axioms, which is the axiomatic counterpart of how generic points are understood as a nondegenerate generalization of concrete points in algebraic geometry. The fundamental concept of uniform substitutions will be explored, which makes it possible to use axioms as if they were axiom schemata without the need for any additional mechanisms or side condition checking. This purely axiomatic

[2] These numbers are to be taken with a grain of salt because the two provers were implemented in different programming languages.

reconsideration of the proof calculus for differential dynamic logic will lead to a new level of appreciation for what the axioms of differential dynamic logic already offered throughout this book without us noticing.

CPS Skills: We identify techniques for a parsimonious straightforward implementation of CPS reasoning. These techniques enable a modular implementation of the logic and the prover mostly independently in parallel, which reduces complexity and makes it easier to advance the reasoning techniques.



18.2 Axioms Versus Axiom Schemata

Recall the axiom [∪] for hybrid programs with a nondeterministic choice $\alpha \cup \beta$ from Lemma 5.1 on p. 144:

$$[\cup] \quad [\alpha \cup \beta]P \leftrightarrow [\alpha]P \wedge [\beta]P \tag{18.1}$$

The innocent way of reading (18.1) is as an axiom schema [∪]. An axiom schema is meant to stand for the infinite family of formulas that have the shape of that axiom schema, so α, β are schema variables or placeholders for arbitrary HPs and P is a placeholder for an arbitrary dL formula. The left-hand side of axiom schema [∪] applies for any dL formula of the form $[\alpha \cup \beta]P$, so to any box modality of any HP that begins with a nondeterministic choice as the top-level operator and has any HPs as subprograms and any dL formula as a postcondition. For example, the left-hand side of axiom schema [∪] fits dL formula $[x := x+1 \cup x' = x^2]x \geq 0$, which implies that the axiom schema [′] justifies the following equivalence:

$$[x := x+1 \cup x' = x^2]x \geq 0 \leftrightarrow [x := x+1]x \geq 0 \wedge [x' = x^2]x \geq 0 \tag{18.2}$$

Of course, this is not the only dL formula that needs to be recognized to be of the shape that axiom schema [∪] indicates. Here are a few more:

$$[x' = x^2 \cup x := x+1]\, x \geq 0 \leftrightarrow [x' = x^2]\, x \geq 0 \wedge [x := x+1]\, x \geq 0$$
$$[x' = 5 \cup x' = -x]\, x^2 \geq 5 \leftrightarrow [x' = 5]\, x^2 \geq 5 \wedge [x' = -x]\, x^2 \geq 5$$
$$[v := v+1; x' = v \cup x' = 2]\, x \geq 5 \leftrightarrow [v := v+1; x' = v]\, x \geq 5 \wedge [x' = 2]\, x \geq 5$$

A direct implementation of axiom schema [∪] consists of an algorithm that takes a dL formula as an input and decides whether that formula is of the form of schema [∪]. Of course, it is crucially important that literally the same postcondition is used for all three modalities of axiom schema [∪]. And it is important that the same HP is used in the left part of the nondeterministic choice $\alpha \cup \beta$ and in the first modality $[\alpha]$ on the right-hand side, and that the same HP is used in the right part of $\alpha \cup \beta$ that is also used in the second modality $[\beta]$ on the right-hand side.[3] Axiom schema [∪] does not even have any side conditions yet, but it already comes with a few tedious conditions to check (if implementing it in an imperative programming language) or match correctly (in a functional programming language with pattern matching).

A more conscious way of reading (18.1) is as an axiom [∪] that literally only refers to one dL formula:

$$[\alpha \cup \beta]P \leftrightarrow [\alpha]P \wedge [\beta]P \tag{18.3}$$

Of course, we will still have to make sure that (18.3) actually is a syntactically well-formed dL formula, which it is not presently. The only formula that such an axiom [∪] ever proves is (18.3). That alone is not so useful, but an axiom is easily implemented, just by copying the dL formula (18.3) from axiom [∪] into the prover.

Alonzo Church's seminal observation is that the sole operation it takes to make more use of an axiom is to provide a uniform substitution mechanism that replaces parts of formulas with other formulas [2]. The trick is to identify when such a replacement is sound. Of course, Church did not know about differential dynamic logic yet, so he settled for first-order logic. But with a sufficiently generalized notion of uniform substitutions for differential dynamic logic [8], we can prove the dL formula (18.3) from axiom [∪] and then use a uniform substitution to prove (18.2) from (18.3). All this takes is the uniform substitution that substitutes $x := x+1$ for α and substitutes $x' = x^2$ for β and simultaneously also substitutes $x \geq 0$ for P uniformly everywhere in (18.3).

Now, the one crucial missing piece is a precise definition of this uniform substitution mechanism. The other crucial element is a precise understanding of whether and what the uniform substitution mechanism needs to check to ensure that all its replacements preserve soundness. And the final missing element is the question of what precise form the syntactic expressions α, β, and P take in (18.1) if it is to be taken literally as an axiom. Then the same process needs to be repeated with an axiomatic reinterpretation of all other dL axioms to find out how they can all be read as axioms instead of as significantly more complicated axiom schemata.

[3] If the formula that is used in place of P has a modality, then the textual description of the places where these occur is, of course, slightly more complex, but they are still in the same places of the expression tree corresponding to the formula.

Admittedly, on a sheet of paper, it is more convenient to work with axiom schemata, because we are now already so well trained to pay attention to make no incorrect reasoning steps by checking all required side conditions. But for precision purposes in a formal verification tool it is substantially easier to work with axioms instead, because the uniform substitution mechanism only needs to be understood and implemented once and because the axioms can be implemented by copy-and-paste. And even when working on a sheet of paper it may be easier to just remember a single uniform substitution mechanism instead of a diverse list of side conditions.

18.3 What Axioms Want

If the axiom of nondeterministic choice $[\cup]$ is internalized as an axiom, not as an axiom schema, then what syntactic elements of differential dynamic logic do the parts of its formula (18.1) correspond to? Suddenly, α and β need to be concrete HPs in the syntax of dL as opposed to schematic variables or placeholders for concrete HPs. Likewise the postcondition P needs to be a concrete dL formula. In fact, revisiting the differential dynamic logic axiom schemata from Chap. 5, there are three different cases of postconditions:

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

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

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

The postcondition p of the vacuous axiom schema V cannot have any variable free that is bound by the HP α. But anything that is not written to by HP α can still be mentioned in p, which is the whole point of this axiom compared to Gödel's generalization proof rule G. In comparison, the postcondition $p(x)$ of the assignment axiom schema $[:=]$ should be allowed to mention variable x despite the fact that it is written to in the HP $x:=e$. That is why the postcondition $p(x)$ mentions x explicitly. The postcondition on the left-hand side of axiom schema $[:=]$ can have the argument x free in the same places that the formula $p(e)$ on the right-hand side of that axiom schema has term e. Its postcondition $p(x)$ can still mention other free variables besides x, because no other variable is written to in the discrete assignment $x:=e$. The postcondition P of the axiom schema of nondeterministic choice $[\cup]$, instead, can have any free variables without reservation, because the axiom is correct whether or not the HPs $\alpha\cup\beta$, α, or β modify the values of free variables of P.

Predicate Symbols

Predicate symbols explain all three cases of postconditions with one joint mechanism. The postcondition p in axiom V has a predicate symbol p with 0 arguments,

so it has no special permission to have its truth-value depend on any particular free variables. The postcondition $p(x)$ in axiom [:=] has a predicate symbol p with variable x as its only argument, so its truth-value can depend on the value of x since $[x:=e]$ binds no other variables, x is the only variable that needs explicit permission to be mentioned in the context $[x:=e]p(x)$. When reading the postcondition P in axiom [∪] as $p(\bar{x})$ for a predicate symbol p that receives the vector $\bar{x}$ of all variables as argument, so its truth-value can depend on the values of all variables, then all cases of postconditions are covered by corresponding predicate symbols that only differ in the number of their arguments.

It is conceptually easier to read the axioms [:=],[∪],V as axioms, so concrete dL formulas, with predicate symbols as postconditions instead of placeholders for formulas. The concrete dL formula $p(x)$ from axiom [:=] literally tells us that its truth-value depends on variable x and apparently nothing else. The formula p from axiom V directly indicates that its truth-value does not depend on the values of any variables. And the case $p(\bar{x})$, which is how we read P in axiom [∪], indicates that its truth-value may depend on the values of all variables $\bar{x}$. We no longer need to keep in mind what other dL formulas the respective postconditions might stand for, but we see the concrete dL formula explicitly.

Separately, we can then worry about what formulas are acceptable as drop-in replacements for predicate symbols. We can find out which replacements are fine once and for all, and independently of the particular axiom at hand. This separation of concerns is liberating because it enables us to understand the soundness of an axiom via the validity of its dL formula independently of the soundness of the mechanism that generalizes and replaces syntactic elements of the axioms with other concrete dL expressions. For example, the concrete instance (18.2) can be obtained from the concrete dL formula (18.3) of axiom [∪] by the uniform substitution

$$\sigma = \{\alpha \mapsto x:=x+1,\ \beta \mapsto x' = x^2,\ P \mapsto x \geq 0\}$$

This substitution σ substitutes HP $x:=x+1$ for α and HP $x' = x^2$ for β and dL formula $x \geq 0$ for P alias $p(\bar{x})$. Of course, this will require us to better understand the substitution process itself and the rôle of the HPs α and β. But let us first stay on the topic of how to interpret predicate symbols.

Unlike a formula such as $x^2 > 5$, which comes with a fixed interpretation of when it is *true*, namely exactly when the square of the value of x exceeds 5, a predicate symbol p does not have a fixed meaning, but is subject to our interpretation. That is what makes it a *symbol*, because it *stands for something*. Certainly, a predicate symbol can take different truth-values depending on its argument. So for example, depending on the value of its argument e, the formula $p(e)$ in axiom [:=] will be *true* or *false*. But if two terms e and $\tilde{e}$ are evaluated to the same real value, then $p(e)$ and $p(\tilde{e})$ will, of course, either both be *true* consistently, or both be *false* consistently. Likewise, the predicate symbol p with 0 arguments in axiom V may be either *true* or *false*. But since it does not take any arguments at all, its truth-value does not depend on the values of any variables, so is independent of the state, and will either be *true* consistently everywhere or *false* consistently everywhere. Indeed, if the assumption

p of axiom V holds then the arity 0 predicate symbol p is *true*, which makes it *true* everywhere, even after running HP α, because its truth-value visibly does not depend on the values of any variables. If, instead p is *false*, then the assumption of axiom V is not met, so its implication is trivially *true*.

Function Symbols

Predicate symbols capture the different cases of formulas in dL axioms. Similarly, in the assignment axiom [:=], the term e needs to be a concrete dL term, but one that can take on any value, because that is what a schema variable placeholder in the corresponding axiom schema [:=] would be able to do. A function symbol with 0 arguments plays that rôle, because a function symbol can be evaluated to any real value, but will then have the same value in all states since it has no variables in its 0 arguments, just as predicate symbols can evaluate to any truth-value.

The following concrete dL formula can, then, be used as assignment axiom [:=]

$$[x:=c()]p(x) \leftrightarrow p(c()) \tag{18.4}$$

with predicate symbol p of arity 1 and function symbol $c()$ of arity 0. For example, the concrete instance

$$[x:=x^2-1]x \geq 0 \leftrightarrow x^2-1 \geq 0 \tag{18.5}$$

can be obtained from (18.4) by the uniform substitution

$$\sigma = \{c() \mapsto x^2-1,\ p(\cdot) \mapsto (\cdot \geq 0)\} \tag{18.6}$$

This substitution σ substitutes the term x^2-1 for the arity 0 function symbol $c()$ and substitutes the greater-or-equal zero comparison formula for the arity 1 predicate symbol p. To indicate that every occurrence of the predicate symbol p of any argument is affected and substituted with the corresponding ≥ 0 comparison, the substitution substitutes $p(\cdot)$ with a dL formula in which the dot $\cdot$ marks where the argument goes in the resulting dL formula. So for any argument e, the formula $p(e)$ will be replaced with $\sigma(e) \geq 0$. Of course, the substitution σ will also need to be applied to the argument e of $p(e)$, not just to the predicate symbol p, which is why $\sigma(e) \geq 0$ is substituted for $p(e)$ and not just $e \geq 0$ when forming (18.5) from (18.4) by (18.6). The result of applying the substitution σ to e is denoted $\sigma(e)$ and will be defined properly later.

Program Constant Symbols

Finally, we return to the rôle that the HPs α and β play in axiom [$\cup$]. On the one hand, both need to be concrete HPs for axiom [$\cup$] to become a concrete dL formula. On the other hand, neither α nor β has a concrete specific behavior, because the

axiom [∪] works for whatever HPs α and β do. Consequently, the HPs we use for α and β in axiom [∪] are what we call *program constant symbols* and can have any arbitrary behavior. Just as predicate symbols do not have a fixed interpretation but might be *true* of any argument, and just as function symbols f do not have a fixed interpretation but might have any real value as a function of the argument's value, so program constant symbols do not have a fixed interpretation but might have any arbitrary behavior. Depending on its interpretation, a program constant symbol might possibly transition from any initial state to any final state, because its behavior is not described explicitly as it would be in the case of a specific differential equation or discrete assignment.

18.4 Differential Dynamic Logic with Interpretations

After having realized what syntactic elements dL axioms need so that they can be faithfully represented as concrete axioms instead of axiom schemata, the first thing we do is officially add those elements to the syntax of differential dynamic logic [8]. Of course, we could have added them right away when introducing hybrid programs in Chap. 3 and differential dynamic logic in Chap. 4, but that would have been a distraction, because we did not need them until now.

18.4.1 Syntax

Differential dynamic logic dL is as usual, except that *function symbols*, *predicate symbols*, and *program constant symbols* are added. Function symbols are usually written f,g,h, predicate symbols p,q,r, and program constant symbols are written a,b,c. Each function and predicate symbol expects a fixed number of terms as arguments, called its *arity*. When f is a function symbol of arity n, then $f(e_1,\ldots,e_n)$ is now also allowed as a term for any n terms $e_1,\ldots,e_n$. Likewise, when p is a predicate symbol of arity n, then $p(e_1,\ldots,e_n)$ is now a formula for any n terms $e_1,\ldots,e_n$. But $f(e_1,\ldots,e_{n-1})$ is not a term, because the function symbol f of arity n has not even received sufficiently many arguments. When we have a function that can, say, add two numbers that we pass as arguments, then we cannot just call this function with one argument or with seven arguments, but need to provide exactly two.

Function symbols are essentially a more liberal generalization of built-in term operators, such as $+$, which has arity 2, is written infix as e_1+e_2 instead of as $+(e_1,e_2)$, and always means addition. Function symbols can have a different number of arguments, but also always expect exactly the same number of arguments as indicated by their arity. Function symbols of arity 0 are also called *constant symbols*, because their value does not depend on any arguments. The use of a function symbol c of arity 0 is sometimes written as $c()$ with empty parentheses for emphasis. In fact, we already allowed rational numbers as constant symbols of arity 0 when originally

defining terms. The meaning of a rational number constant is, of course, also fixed. The meaning of the rational number constant 1 is always 1 and the meaning of the rational number constant $1/2$ is always the real number 0.5.

By contrast, function symbols are more general, because they are actually meant as symbols. That is, they do not have a fixed meaning once and for all time, but are symbolic, so their meaning is subject to interpretation. Similarly, predicate symbols are symbols so their meaning depends on our interpretation, and likewise for program constant symbols.

Definition 18.1 (Terms). A *term* e is defined by augmenting the grammar from Definition 2.2 on p. 42 with the following case (where $e_1,\dots,e_n$ are n terms and f is a function symbol of arity n):

$$e ::= f(e_1,\dots,e_n) \mid \dots$$

Definition 18.2 (Hybrid program). *Hybrid programs* are defined by augmenting the grammar from Definition 3.1 on p. 76 with the following case (where a is any program constant symbol):

$$\alpha,\beta ::= a \mid \dots$$

Definition 18.3 (dL formula). The *formulas of differential dynamic logic* (dL) are defined by augmenting the grammar from Definition 4.1 on p. 111 with the following case (where $e_1,\dots,e_n$ are terms and p is an arity n predicate symbol):

$$P ::= p(e_1,\dots,e_n) \mid \dots$$

For emphasis, we might call the resulting logic *differential dynamic logic with interpretations*, but continue to just call it dL, because we just neglected to consider these extensions until now, since they were not necessary for our understanding yet.

This extension of the syntax of dL makes it possible to phrase all axioms we saw before as axioms with concrete dL formulas (instead of as axiom schemata that represent their infinitely many instances subject to side conditions). For example, the axiom schemata we considered as motivating examples above turn into

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

$$[\cup]\ [a\cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \wedge [b]p(\bar{x})$$

$$\mathrm{V}\ p \rightarrow [a]p$$

18.4.2 Semantics

The semantics of function symbols, predicate symbols, and program constant symbols is actually easy, but it comes with a twist compared to all other definitions of semantics we saw anywhere else in this textbook. The whole point of function symbols, predicate symbols, and program constant symbols is that they are symbolic, so they do not come with a fixed interpretation. Consequently, unlike for the binary $+$ operator, which always means addition, the semantics of a term mentioning a function symbol f of arity 2 depends on how we interpret the symbol f, which may be addition or multiplication or any other reasonable function from two reals to a real.

In order to be able to evaluate to a real number any term in any state, we fix an *interpretation* I that assigns a (sufficiently smooth[4]) n-ary function $I(f) : \mathbb{R}^n \to \mathbb{R}$ to every function symbol f of arity n. Given such an interpretation I, we can easily evaluate every term in any state ω just by looking up in the interpretation I the corresponding function $I(f)$ for every function symbol f in the term and using the variable values from the state ω.

Definition 18.4 (Semantics of terms). The *value of term* e in state $\omega \in \mathscr{S}$ for interpretation I is a real number denoted $\omega[\![e]\!]$ and is defined by augmenting Definition 2.4 with the following case:

$$\omega[\![f(e_1,\ldots,e_n)]\!] = I(f)\big(\omega[\![e_1]\!],\ldots,\omega[\![e_n]\!]\big) \text{ if } f \text{ is a function symbol of arity } n$$

That is, in state ω, a function symbol application is evaluated to the result of the function $I(f)$ applied to the real values $\omega[\![e_i]\!]$ to which the respective argument terms e_i are evaluated in the state ω.

As predicate symbols have no fixed interpretation either, the interpretation I also assigns an n-ary relation $I(p) \subseteq \mathbb{R}^n$ to every predicate symbol p of arity n. With such an interpretation, the set of states in which a formula is true can be defined easily.

Definition 18.5 (dL semantics). The *semantics* of a dL formula P for interpretation I is the set of states $[\![P]\!] \subseteq \mathscr{S}$ in which P is true, and is defined by augmenting Definition 4.2 with the following case:

12. $[\![p(e_1,\ldots,e_n)]\!] = \big\{\omega : \big(\omega[\![e_1]\!],\ldots,\omega[\![e_n]\!]\big) \in I(p)\big\}$
    That is, a predicate symbol application is true in the set of states ω in which the arguments terms e_i are evaluated to a tuple of real numbers that is in the relation $I(p)$.

A formula P is *valid*, written $\models P$, iff it is true in all states of all interpretations I, i.e., $[\![P]\!] = \mathscr{S}$, so $\omega \in [\![P]\!]$ for all states ω and all interpretations I.

[4] Functions that are continuously differentiable are smooth enough for our purposes.

Finally, the interpretation I also assigns a reachability relation $I(a) \subseteq \mathscr{S} \times \mathscr{S}$ to every program constant symbol a. As usual, $(\omega, \nu) \in [\![a]\!]$ indicates that final state ν is reachable from initial state ω in the HP a.

Definition 18.6 (Transition semantics of HPs). Each HP α is interpreted semantically as a binary reachability relation $[\![\alpha]\!] \subseteq \mathscr{S} \times \mathscr{S}$ over states for each interpretation ω, and is defined by augmenting Definition 3.2 with the case:

7. $[\![a]\!] = I(a)$
   That is, the reachability relation for program constant symbol a is an arbitrary state transition relation determined by the interpretation I.

With this extension of the semantics, it is now easy to see that the dL formula in the V axiom is valid. In fact, this is the easiest possible proof of the soundness of the vacuous axiom V (Lemma 5.11).

Lemma 18.1 (V vacuous axiom). *The* vacuous axiom *is sound:*

$$\mathrm{V} \quad p \to [a]p$$

Proof. The truth of an arity 0 predicate symbol p just depends on the interpretation I but not on the state ω since p does not have any variables. Consequently, either p is interpreted to be *true* by I, in which case $[a]p$ is *true* as well, because if p holds in all states then it also holds in all states reachable after running HP a. Or p is interpreted to be *false* by I, in which case the assumption p is *false* and the implication $p \to [a]p$ is vacuously *true*. ∎

Likewise the equivalence of the dL formula in the assignment axiom [:=] is easily seen to be valid (Lemma 5.2).

Lemma 18.2 ([:=] assignment axiom). *The* assignment axiom *is sound:*

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

Proof. Predicate symbol p is *true* of x after assigning the new value $c()$ to x (so $[x:=c()]p(x)$) iff predicate symbol p is *true* of the new value $c()$ (so $p(c())$). ∎

18.5 Uniform Substitution

A *uniform substitution* σ substitutes function symbols with terms, predicate symbols with formulas, and program constant symbols with hybrid programs, and it does so uniformly, e.g., it uses the same HP as replacement for program constant symbol b in all places.[5] The result of applying the uniform substitution σ to dL

[5] Replacing the same program constant symbol b with different HPs in different places would be very illogical and break all structure there ever was. Let's don't ever be so silly!

formula ϕ is denoted $\sigma(\phi)$. Similarly, $\sigma(\theta)$ denotes the result of applying the uniform substitution σ to term θ and $\sigma(\alpha)$ denotes the result of applying the uniform substitution σ to HP α. They will all be defined rigorously in Sect. 18.5.3.

The substitution σ defines a term σf as a replacement for each arity 0 function symbol f. The substitution σ also defines a dL formula σp as a replacement for each arity 0 predicate symbol p. It also defines a hybrid program σa for each program constant symbol a. Applying the substitution σ will replace every occurrence of program constant symbol a uniformly with the HP σa and every occurrence of arity 0 function symbol f with σf and every occurrence of arity 0 predicate symbol p with the corresponding replacement σp. For function and predicate symbols with arguments, the reserved function symbol $\cdot$ is used as a placeholder to indicate where the argument goes. For an arity 1 function symbol f, the substitution defines a term whose occurrences of function symbol $\cdot$ indicate where the argument of f is placed. For an arity 1 predicate symbol p, the substitution defines a dL formula whose occurrences of function symbol $\cdot$ indicate where the argument of p goes.

The notation for describing a uniform substitution σ that substitutes term e_1 for arity 1 function symbol f and substitutes term e_2 for arity 0 function symbol c, and that substitutes dL formula ϕ_1 for arity 1 predicate symbol p and substitutes dL formula ϕ_2 for arity 0 predicate symbol q and substitutes hybrid program α for program constant symbol a is

$$\sigma = \{f(\cdot) \mapsto e_1, c \mapsto e_2, p(\cdot) \mapsto \phi_1, q \mapsto \phi_2, a \mapsto \alpha\} \tag{18.7}$$

The occurrences of reserved arity 0 function symbol $\cdot$ in the term e_1 and in the formula ϕ_1, respectively, indicate where the arguments of f and of p, respectively, go in the replacement. We have already seen examples of uniform substitutions in Sect. 18.3. The uniform substitution σ in (18.7) replaces arity 1 function symbol f and predicate symbol p, arity 0 function symbol c and predicate symbol q, and program constant symbol a but leaves all other symbols alone. The *domain* of substitution σ is the set of all symbols it replaces, so $\{f, c, p, q, a\}$ for (18.7).

18.5.1 Uniform Substitution Rule

Church's uniform substitution proof rule US says that the result $\sigma(\phi)$ of applying a uniform substitution σ to a valid formula ϕ is valid, too. Its generalization to differential dynamic logic is sound as well [8]. The intuition is that, if a formula ϕ is valid, so true in all states with any interpretation of its predicate, function, and program constant symbols, then it is also valid after substituting concrete formulas in for its predicate symbols, etc., because the predicate symbol very well may be interpreted to have the same truth-value as its substitute formula. The tricky part is the correct handling of arguments of the predicate symbols and of variables in the replacements, because variables may have different values in different subformulas.

Theorem 18.1 (Uniform substitution). *The proof rule US is sound:*

$$\text{US}\ \frac{\phi}{\sigma(\phi)}$$

So if formula ϕ has a proof, then its uniform substitution instance $\sigma(\phi)$ has a proof, too, just by applying the uniform substitution proof rule US. *The uniform substitution mechanism checks that it does not introduce a free variable in a context in which it is bound in* $\sigma(\phi)$. If the uniform substitution σ applied to ϕ were to introduce a free variable x into a context in which x has been bound, then $\sigma(\phi)$ is not defined, because it *clashes*, and the proof rule US is not applicable to ϕ.

Before proceeding with an exact definition of the uniform substitution mechanism constructing $\sigma(\phi)$ in Sect. 18.5.3, we explore a number of representative examples to gain intuition for the rôle of rule US in proving.

The formula $(\neg\neg p) \leftrightarrow p$, for example, is valid (in classical logic). When we pick any dL formula ψ, then also valid will be the formula that results from $(\neg\neg p) \leftrightarrow p$ by uniformly substituting all occurrences of arity 0 predicate symbol p with this formula ψ. For example, the uniform substitution $\sigma = \{p \mapsto [x' = x^2]\, x \geq 0\}$ proves

$$\text{US}\frac{(\neg\neg p) \leftrightarrow p}{(\neg\neg[x' = x^2]\, x \geq 0) \leftrightarrow [x' = x^2]\, x \geq 0}$$

Any other formula could have been used as a replacement for p (consistently everywhere) as well and rule US would have proved the result from $(\neg\neg p) \leftrightarrow p$.

Substitutions are more subtle when working, e.g., from the formula $(\forall x\, p) \leftrightarrow p$. This formula expresses for an arity 0 predicate symbol p that p is true for all x if and only if p is true in the current state, which makes apparent sense, because the arity 0 predicate symbol p quite visibly does not mention any variables that its truth-value would depend on. In fact, it is precisely this absence of the mention of x that the validity of the formula $(\forall x\, p) \leftrightarrow p$ depends on. We cannot possibly soundly replace p with $x \geq 0$, because that would lead to

$$\text{clash}\lightning\frac{(\forall x\, p) \leftrightarrow p}{\forall x\, (x \geq 0) \leftrightarrow x \geq 0}$$

which is unsound, because not all values of x are nonnegative (left) just because the present value of x is nonnegative (right) in the current state. Indeed, the uniform substitution mechanism will clash when applying $\sigma = \{p \mapsto x \geq 0\}$ to $(\forall x\, p) \leftrightarrow p$, because σ would introduce the free variable x in the replacement for p in a context $\forall x\, p$ in which x refers to a bound variable, such that the variable x in the replacements for the two occurrences of p would possibly refer to two different values. The requirement that the replacement for p does not have x as a free variable is quite consistent with our original reason why the premise $(\forall x\, p) \leftrightarrow p$ was valid at all.

Variables other than x can be mentioned free in the replacement for p, though, because they are not bound anywhere where p occurs. For example, the uniform substitution $\sigma = \{p \mapsto y \geq 0\}$ enables rule US to prove

$$\text{US}\frac{(\forall x\, p) \leftrightarrow p}{\forall x\,(y \geq 0) \leftrightarrow y \geq 0}$$

18.5.2 Examples

The primary, but not the only, use case of the uniform substitution proof rule US is that it makes it possible to instantiate axioms with specific dL formulas. The following examples will, thus, have an axiom as premise, which is proved in the dL calculus just by mentioning its name. The primary focus will be on demonstrating how uniform substitutions work, when they clash, and why that is soundness-critical.

How Uniform Substitutions Handle Arguments

Rule US proves, for example, (18.5) from (18.4) with uniform substitution (18.6):

$$\text{US}\frac{[x:=c()]p(x) \leftrightarrow p(c())}{[x:=x^2-1]x \geq 0 \leftrightarrow x^2-1 \geq 0}$$

Intuitively, this uniform substitution replaces all occurrences of function symbol $c()$ with x^2-1 while also replacing all occurrences of predicate symbol p with a greater-or-equal-to-zero comparison. Of course, in addition to substituting $(\cdot \geq 0)$ for $p(\cdot)$, the uniform substitution is also used on all arguments e of p in any subformula $p(e)$. So σ uniformly replaces every occurrence of $p(e)$ with $\sigma(e) \geq 0$. In particular, σ replaces $p(x)$ with $x \geq 0$ but $p(c())$ with $x^2-1 \geq 0$.

The uniform substitution $\sigma = \{c() \mapsto x^2-1,\ p(\cdot) \mapsto (\cdot \geq x)\}$, instead, clashes for the same formula, because the replacement for $p(\cdot)$ would introduce the free variable x in a context $[x:=x^2-1]\_$ in which x is bound:

$$\text{clash}\lightning\frac{[x:=c()]p(x) \leftrightarrow p(c())}{[x:=x^2-1]x \geq x \leftrightarrow x^2-1 \geq x} \tag{18.8}$$

It is crucial for soundness that this substitution clashes, because the premise is valid (axiom [:=]) but the conclusion is not, because the postcondition $x \geq x$ of the assignment is valid, but the right-hand side $x^2-1 \geq x$ is not. This makes sense, because all free occurrences of x in the postcondition are affected by the assignment $x:=x^2-1$, so the substitution $\{p(\cdot) \mapsto (\cdot \geq x)\}$ does not select *all* occurrences of x for the $\cdot$ placeholder. In contrast, the uniform substitution $\sigma = \{c() \mapsto x^2-1,\ p(\cdot) \mapsto (\cdot \geq \cdot)\}$ gives the perfectly acceptable result

$$\text{US}\frac{[x:=c()]p(x)\leftrightarrow p(c())}{[x:=x^2-1]x\geq x\leftrightarrow x^2-1\geq x^2-1}$$

Likewise, the uniform substitution $\sigma=\{c()\mapsto x^2-1,\ p(\cdot)\mapsto(2(\cdot)\geq\cdot)\}$ results in

$$\text{US}\frac{[x:=c()]p(x)\leftrightarrow p(c())}{[x:=x^2-1]2x\geq x\leftrightarrow 2(x^2-1)\geq x^2-1}$$

In comparison, the uniform substitution $\sigma=\{c()\mapsto x^2-1,\ p(\cdot)\mapsto(\cdot\geq y)\}$ is acceptable, because, even if the replacement for $p(\cdot)$ introduces the free variable y, it only introduces it in the context $[x:=x^2-1]\_$ in which y is not bound either:

$$\text{US}\frac{[x:=c()]p(x)\leftrightarrow p(c())}{[x:=x^2-1]x\geq y\leftrightarrow x^2-1\geq y}$$

Observe how the explicit argument x in the subformula $p(x)$ of the premise makes it possible for the substitute $x\geq y$ to mention x instead of placeholder $\cdot$ in its replacement $(\cdot\geq y)$. But as (18.8) demonstrated, even such a mention of x as an argument does not give license to use variable x anywhere else in the replacements. Of course, the argument x in $p(x)$ only indicates an explicit license for a possible dependence on x, not that $p(x)$ has to depend on x. For example, this uniform substitution $\sigma=\{c()\mapsto 2x+1,\ p(\cdot)\mapsto(y^2\geq y)\}$ does not use the $\cdot$ argument placeholder:

$$\text{US}\frac{[x:=c()]p(x)\leftrightarrow p(c())}{[x:=2x+1]y^2\geq y\leftrightarrow y^2\geq y}$$

Uniform substitutions can also have predicates in which the argument placeholder $\cdot$ appears in more deeply nested positions. For example, the uniform substitution $\sigma=\{c()\mapsto x^2,\ p(\cdot)\mapsto[(y:=\cdot+y)^*](\cdot\geq y)\}$ is acceptable, because it does not introduce any free variables in a context in which they are bound:

$$\text{US}\frac{[x:=c()]p(x)\leftrightarrow p(c())}{[x:=x^2][(y:=x+y)^*](x\geq y)\leftrightarrow[(y:=x^2+y)^*](x^2\geq y)}$$

How Uniform Substitutions Handle Constant Predicate Symbols

Without the original formula mentioning x as an argument in $p(x)$, the uniform substitution cannot use x in a context in which x is bound. For example, the uniform substitution $\sigma=\{a\mapsto x'=5,\ p\mapsto(x\leq 5)\}$ clashes, because the replacement for p introduces the free variable x into the context $[x'=5]\_$ in which x is bound, which is the context that results from applying σ to $[a]p$:

$$\text{clash}\lightning\frac{p\rightarrow[a]p}{x\leq 5\rightarrow[x'=5]x\leq 5}$$

It is crucial for soundness that this substitution clashes, because the premise is valid (axiom V), but the conclusion is not, because x does not stay below 5 forever when following the differential equation $x' = 5$. This is precisely what the side condition of axiom schema V from Lemma 5.11 prevents, too. But unlike axiom schemata, rule US does not need special purpose knowledge about how to prevent such incorrect uses for the particular case of axiom V. It provides a generic mechanism.

In contrast, the uniform substitution $\sigma = \{a \mapsto x' = 5,\ p \mapsto (y \leq 5)\}$ works fine, because it only introduces free variable y in the resulting context $[x' = 5]\_$ in which y is not bound anyhow:

$$\text{US}\frac{p \rightarrow [a]p}{y \leq 5 \rightarrow [x' = 5]y \leq 5}$$

The uniform substitution $\sigma = \{a \mapsto (v := v + 1; \{x' = v, v' = -b\}),\ p \mapsto (y \leq b)\}$ works fine, because its function and predicate symbols only introduce free variables y and b in a context in which they are possibly read but never written:

$$\text{US}\frac{p \rightarrow [a]p}{y \leq b \rightarrow [v := v + 1; \{x' = v, v' = -b\}]y \leq b}$$

Telling the respective good and bad cases of axiom instantiation attempts apart is what uniform substitutions achieve without having to provide any side conditions that are specific to the particular formulas or axiom schemata at hand. Uniform substitutions provide a uniform answer, once and for all, to the question of which instantiations of formulas are sound because they preserve validity.

How Uniform Substitutions Handle Program Constant Symbols

When working from the assignment axiom [:=] with its postcondition $p(x)$ or from the vacuous axiom V with its postcondition p, the uniform substitution rule US needs to check for capture of other variables, which is important for soundness. When working from the nondeterministic choice axiom [∪] with its postcondition $p(\bar{x})$, instead, then this postcondition has explicit permission to mention all variables $\bar{x}$, such that any dL formula can be accepted as replacement. The uniform substitution $\sigma = \{a \mapsto v := -cv,\ b \mapsto x'' = -g,\ p(\bar{x}) \mapsto 2gx \leq 2gH - v^2\}$ yields

$$\text{US}\frac{[a \cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \wedge [b]p(\bar{x})}{[v := \text{-}cv \cup x'' = \text{-}g]2gx \leq 2gH - v^2 \leftrightarrow [v := \text{-}cv]2gx \leq 2gH - v^2 \wedge [x'' = \text{-}g]2gx \leq 2gH - v^2}$$

As usual, $x'' = -g$ is short for $\{x' = v, v' = -g\}$.

18.5.3 Uniform Substitution Application

A uniform substitution can replace any number of function, predicate, or program constant symbols simultaneously. The notation $\sigma f(\cdot)$ denotes the replacement for $f(\cdot)$ according to σ, i.e., the value $\sigma f(\cdot)$ of function σ at $f(\cdot)$. By contrast, $\sigma(\phi)$ denotes the result of applying σ to ϕ which we defined now (likewise for $\sigma(\theta)$ and $\sigma(\alpha)$). The notation $f \in \sigma$ signifies that σ replaces function symbol f, i.e., $\sigma f(\cdot) \neq f(\cdot)$, so f is in the domain of σ. Likewise, the notation $p \in \sigma$ signifies that σ replaces predicate symbol p, and correspondingly $a \in \sigma$ means that σ replaces program constant symbol a.

$\sigma(x)$	$= x$	for variable $x \in \mathscr{V}$
$\sigma(f(e))$	$= (\sigma(f))(\sigma(e)) \stackrel{\text{def}}{=} \{\cdot \mapsto \sigma(e)\}(\sigma f(\cdot))$	for function symbol $f \in \sigma$
$\sigma(g(e))$	$= g(\sigma(e))$	for function symbol $g \notin \sigma$
$\sigma(e+\tilde{e})$	$= \sigma(e)+\sigma(\tilde{e})$	
$\sigma(e \cdot \tilde{e})$	$= \sigma(e) \cdot \sigma(\tilde{e})$	
$\sigma((e)')$	$= (\sigma(e))'$	if σ is $\mathscr{V}$-admissible for e
$\sigma(e \geq \tilde{e})$	$\equiv \sigma(e) \geq \sigma(\tilde{e})$	likewise for $>, =, <, \leq$
$\sigma(p(e))$	$\equiv (\sigma(p))(\sigma(e)) \stackrel{\text{def}}{\equiv} \{\cdot \mapsto \sigma(e)\}(\sigma p(\cdot))$	for predicate symbol $p \in \sigma$
$\sigma(q(e))$	$\equiv q(\sigma(e))$	for predicate symbol $q \notin \sigma$
$\sigma(\neg\phi)$	$\equiv \neg\sigma(\phi)$	
$\sigma(\phi \wedge \psi)$	$\equiv \sigma(\phi) \wedge \sigma(\psi)$	likewise for $\vee, \rightarrow, \leftrightarrow$
$\sigma(\forall x \phi)$	$\equiv \forall x \sigma(\phi)$	if σ is $\{x\}$-admissible for ϕ
$\sigma(\exists x \phi)$	$\equiv \exists x \sigma(\phi)$	if σ is $\{x\}$-admissible for ϕ
$\sigma([\alpha]\phi)$	$\equiv [\sigma(\alpha)]\sigma(\phi)$	if σ is $\mathrm{BV}(\sigma(\alpha))$-admissible for ϕ
$\sigma(\langle\alpha\rangle\phi)$	$\equiv \langle\sigma(\alpha)\rangle\sigma(\phi)$	if σ is $\mathrm{BV}(\sigma(\alpha))$-admissible for ϕ
$\sigma(a)$	$\equiv \sigma a$	for program constant symbol $a \in \sigma$
$\sigma(b)$	$\equiv b$	for program constant symbol $b \notin \sigma$
$\sigma(x := e)$	$\equiv x := \sigma(e)$	
$\sigma(x' = e \,\&\, Q)$	$\equiv x' = \sigma(e) \,\&\, \sigma(Q)$	if σ is $\{x, x'\}$-admissible for e, Q
$\sigma(?Q)$	$\equiv ?\sigma(Q)$	
$\sigma(\alpha \cup \beta)$	$\equiv \sigma(\alpha) \cup \sigma(\beta)$	
$\sigma(\alpha;\beta)$	$\equiv \sigma(\alpha);\sigma(\beta)$	if σ is $\mathrm{BV}(\sigma(\alpha))$-admissible for β
$\sigma(\alpha^*)$	$\equiv (\sigma(\alpha))^*$	if σ is $\mathrm{BV}(\sigma(\alpha))$-admissible for α

Fig. 18.1 Recursive application of uniform substitution σ

Figure 18.1 defines the result $\sigma(\phi)$ of applying to a dL formula ϕ the *uniform substitution* σ that uniformly replaces all occurrences of a function f by a term (instantiated with its respective argument of f) and all occurrences of a predicate p by a formula (instantiated with its argument) as well as of a program constant symbol a by a program. Each case in Fig. 18.1 applies the uniform substitution recursively.[6] In each case, the uniform substitution application mechanism checks that the substitution is admissible for the bound variables of the operator, i.e., σ will not introduce

[6] This makes the uniform substitution a *homomorphism*, because the substitution of an addition is the addition of substitutions: $\sigma(e+\tilde{e}) = \sigma(e)+\sigma(\tilde{e})$ and accordingly for all other operators.

free variables in the scope of an operator in which they are bound (which will be defined in Definition 18.7 below).

For example, for the case $\sigma(\forall x\, \phi)$, the set of bound variables that σ needs to be admissible for ϕ is $\{x\}$, because if σ were to introduce free variable x while forming $\sigma(\phi)$, then x would be incorrectly captured by quantifier $\forall x$. Suppose the substitution σ were to replace an arity 0 predicate symbol p that occurs in ϕ with the formula $x \geq 0$; then within the scope of the quantifier of $\forall x\, \phi$, this formula $x \geq 0$ refers to a different variable called x, namely the one bound by the universal quantifier $\forall x$ and no longer the free variable x. That is why such a uniform substitution is not defined, because it is not admissible. This is crucial for soundness, e.g., for the formula $p \leftrightarrow \forall x\, p$, because the substitution would otherwise replace p inconsistently with the same formula $x \geq 0$ but referring to different values of x in different places, since one of the newly introduced occurrences of x in the resulting $x \geq 0 \leftrightarrow \forall x\, (x \geq 0)$ is in the scope of a quantifier binding x. In the case of a modal formula $\sigma([\alpha]\phi)$, the bound variables that are taboo and cannot be introduced as free variables when forming the substituted postcondition $\sigma(\phi)$ are the bound variables $\mathrm{BV}(\sigma(\alpha))$ of the substituted HP $\sigma(\alpha)$. In the case of a differential equation $\sigma(x' = e \,\&\, Q)$, the bound variables $\{x, x'\}$ are taboo and cannot be introduced as free variables when forming $\sigma(e)$ or $\sigma(Q)$, since the differential equation changes the values of both.

Arguments are put in for the placeholder $\cdot$ recursively by uniform substitution $\{\cdot \mapsto \sigma(\theta)\}$ in Fig. 18.1, which is well-defined since it replaces the placeholder function symbol $\cdot$ of arity 0 by the readily substituted argument $\sigma(\theta)$. Recall the definition of the free variables $\mathrm{FV}(P)$ as well as the bound variables $\mathrm{BV}(P)$ of formula P from Sects. 5.6.5 and 5.6.6.

Definition 18.7 (Admissible uniform substitution). A uniform substitution σ is *U-admissible for formula ϕ* (or term θ or HP α, respectively) with respect to the variables $U \subseteq \mathscr{V}$ iff $\mathrm{FV}(\sigma|_{\Sigma(\phi)}) \cap U = \emptyset$, where $\sigma|_{\Sigma(\phi)}$ is the restriction of substitution σ that only replaces symbols that occur in ϕ, and $\mathrm{FV}(\sigma) = \bigcup_{f \in \sigma} \mathrm{FV}(\sigma f(\cdot)) \cup \bigcup_{p \in \sigma} \mathrm{FV}(\sigma p(\cdot))$ is the set of *free variables* that σ introduces for function or predicate symbols.
A uniform substitution σ is *admissible for* ϕ (or θ or α, respectively) iff the bound variables U of each operator of ϕ are not free in the substitution on its arguments, i.e., σ is U-admissible. These admissibility conditions are listed explicitly in Fig. 18.1, which defines the result $\sigma(\phi)$ of applying σ to ϕ. For each case in Fig. 18.1, the taboo set U whose U-admissibility is required of σ is exactly the set of variables that are bound by its top-level operator.

The substitution σ is said to *clash* and its result $\sigma(\phi)$ (or $\sigma(\theta)$ or $\sigma(\alpha)$) is not defined if σ is not admissible, in which case rule US is not applicable either. All the admissibility conditions in Fig. 18.1 are easily summarized:

If you bind a free variable, you go to logic jail!

Note that the free variables $\mathrm{FV}(\sigma)$ of a substitution σ are only defined as the union of the free variables of the replacements for its function symbols f and pred-

icate symbols p, not the program constant symbols, because programs may already read the full state and change it to a new state. Likewise, replacements of predicate symbols $p(\bar{x})$ with all variables $\bar{x}$ as arguments are disregarded in the free variable determination, because they apparently already have explicit permission to depend on the values of all variables and, thus, do not introduce any new free variables.

Finally, observe that σ is already U-admissible for formula ϕ if the sufficient condition $\mathrm{FV}(\sigma) \cap U = \emptyset$ holds. The only reason for Definition 18.7 to restrict the admissibility check to the restriction $\sigma|_{\Sigma(\phi)}$ of the substitution to the symbols that actually occur in the affected formula ϕ is that there is no need for the substitution to clash if σ introduces free variables for function or predicate symbols that do not even occur in ϕ. For example, $\sigma = \{p(\cdot) \mapsto (\cdot \leq y), q \mapsto (x \leq 5)\}$ is $\{x\}$-admissible for $\phi \overset{\text{def}}{=} (x > 2 \land p(y))$, because the dangerous predicate symbol q with its free variable x that would not be $\{x\}$-admissible does not even occur in ϕ, so the substitution is restricted to $\sigma|_{\Sigma(\phi)} = \{p(\cdot) \mapsto (\cdot \leq y)\}$, whose only free variable is y. Neither the original substitution σ nor its restriction $\sigma|_{\Sigma(\phi)}$ are $\{y\}$-admissible for ϕ, because both have y as a free variable. The original substitution σ also would not be $\{x\}$-admissible for $\psi \overset{\text{def}}{\equiv} (x > 2 \land p(y) \land q)$, because its replacement for the predicate symbol q that occurs in ψ has x as a free variable.

For example, this uniform substitution $\sigma = \{a \mapsto x' = 5,\ p \mapsto (y \leq 5)\}$ succeeds:

$$\text{US}\frac{p \to [a]p}{y \leq 5 \to [x' = 5]y \leq 5}$$

It uses the uniform substitution mechanism in Fig. 18.1 and also $y \notin \mathrm{BV}(x' = 5)$:

$$\begin{aligned}&\sigma(p \to [a]p) \equiv \sigma(p) \to \sigma([a]p) \equiv \sigma(p) \to [\sigma(a)]\sigma(p)\\ \equiv\ &\sigma p \to [\sigma a]\sigma p \equiv y \leq 5 \to [x' = 5]y \leq 5\end{aligned}$$

In addition to the previous examples, we consider a few very insightful ones. The uniform substitution $\sigma = \{p(\cdot) \mapsto (\cdot \geq 0),\ q \mapsto (y < 0)\}$ works fine, because it only introduces free variable y in the context $\forall x\_$ in which y is not bound:

$$\text{US}\frac{\forall x\,(p(x) \lor q) \leftrightarrow (\forall x\, p(x)) \lor q}{\forall x\,(x \geq 0 \lor y < 0) \leftrightarrow (\forall x\,(x \geq 0)) \lor y < 0}$$

The application of uniform substitution according to Fig. 18.1 is straightforward:

$$\begin{aligned}&\sigma(\forall x\,(p(x) \lor q) \leftrightarrow (\forall x\, p(x)) \lor q) \equiv \sigma(\forall x\,(p(x) \lor q)) \leftrightarrow \sigma((\forall x\, p(x)) \lor q)\\ \equiv\ &\forall x\,(\sigma(p(x) \lor q)) \leftrightarrow \sigma(\forall x\, p(x)) \lor \sigma(q) \equiv \forall x\,(\sigma(p(x)) \lor \sigma(q)) \leftrightarrow \forall x\,\sigma(p(x)) \lor \sigma(q)\\ \equiv\ &\forall x\,(x \geq 0 \lor y < 0) \leftrightarrow (\forall x\,(x \geq 0)) \lor y < 0\end{aligned}$$

This substitution uses that x is not free in the replacement for q.

In contrast, uniform substitution $\sigma = \{p(\cdot) \mapsto (\cdot \geq 0),\ q \mapsto (x < 0)\}$ clashes as its replacement for q introduces free variable x in a context $\forall x\_$ in which it is bound:

$$\text{clash}\lightning\ \frac{\forall x\,(p(x)\lor q)\leftrightarrow(\forall x\,p(x))\lor q}{\forall x\,(x\geq 0\lor x<0)\leftrightarrow(\forall x\,(x\geq 0))\lor x<0}$$

This is soundness-critical, because the left formula is valid (every number is either greater-or-equal or smaller than 0) but the right formula is not, because it is equivalent to $x<0$, which imposes a condition on the present value of x. The uniform substitution application $\sigma(\forall x\,(p(x)\lor q))$ clashes, because σ is not $\{x\}$-admissible for $p(x)\lor q$ on account of the replacement $x<0$ for q having free variable x which would already be bound by the $\forall x$ quantifier. Of course, this makes sense, because a disjunction can only be pulled outside the scope of a quantifier if it does not actually use the quantified variable. That is precisely what the premise expresses. Indeed, the premise can be proved from other quantifier axioms.

Observe that it is crucial for soundness that even an occurrence of $p(x)$ in a context where x is bound does not permit free variable x to be mentioned in the replacement except in the places of the $\cdot$ placeholder. For example, uniform substitution $\sigma=\{c()\mapsto 0,\ p(\cdot)\mapsto(\cdot\geq x)\}$ clashes when used on the assignment axiom [:=], because the replacement for $p(\cdot)$ would introduce the extra free variable x in a context $[x:=0]\_$ in which x is bound:

$$\text{clash}\lightning\ \frac{[x:=c()]p(x)\leftrightarrow p(c())}{[x:=0]x\geq x\leftrightarrow 0\geq x}$$

The premise is valid (axiom [:=]) but the conclusion is not, because the postcondition $x\geq x$ of the assignment is valid, but the right-hand side $0\geq x$ is not. The reason is that the free variable x in the replacement $(\cdot\geq x)$ for $p(\cdot)$ would refer to the variable bound by $x:=0$ in the substitute for $p(x)$ but would refer to a free variable x in the substitute for $p(c())$.

Uniform substitutions also need to pay attention when substituting in the argument. For example, $\sigma=\{c()\mapsto y^2,\ p(\cdot)\mapsto[(y:=\cdot+y)^*](\cdot\geq y)\}$ clashes when applied to the assignment axiom [:=] while substituting the replacement y^2 for $c()$ for the argument placeholder $\cdot$ in the replacement for $p(c())$, since that would introduce free variable y into a context $[(y:=\cdot+y)^*](\cdot\geq y)$ where it is bound:

$$\text{clash}\lightning\ \frac{[x:=c()]p(x)\leftrightarrow p(c())}{[x:=y^2][(y:=x+y)^*](x\geq y)\leftrightarrow[(y:=y^2+y)^*](y^2\geq y)}$$

This is, of course, crucial for soundness because the left loop always adds to y the *same* value in each round (the square of the initial value of y) while the right loop, instead, always adds to y the square of the *most recent* value of y.

18.5.4 Uniform Substitution Lemmas

The key to understanding why the rule US is sound is the uniform substitution lemma that relates the syntactic change that a uniform substitution makes to a cor-

responding semantic reinterpretation called *adjoint interpretation*. The idea is that instead of syntactically replacing a predicate symbol p with another formula when forming the result $\sigma(\phi)$ of a uniform substitution σ, one might just as well modify the interpretation of the predicate symbol p. The uniform substitute $\sigma(\phi)$ of a formula is true in state ω in an interpretation I iff the formula ϕ itself is true in ω in its adjoint interpretation $\sigma^*_\omega I$. The semantic modification of adjoint interpretations has the same effect as the syntactic uniform substitution but on the semantics.

For example, recall that to prove (18.5) from (18.4) we used US with substitution

$$\sigma = \{c() \mapsto x^2-1,\ p(\cdot) \mapsto (\cdot \geq 0)\} \tag{18.6*}$$

$$\text{US}\frac{[x:=c()]p(x) \leftrightarrow p(c())}{[x:=x^2-1]x \geq 0 \leftrightarrow x^2-1 \geq 0}$$

Instead of syntactically substituting $(\cdot \geq 0)$ for $p(\cdot)$ everywhere, we could have reinterpreted predicate symbol p in a different way, namely such that $\sigma^*_\omega I(p)$ holds true iff its argument is greater-or-equal 0. And instead of syntactically substituting x^2-1 for $c()$ everywhere, we could have reinterpreted function symbol $c()$ such that $\sigma^*_\omega I(c())$ has the value that x^2-1 has in state ω. In the so-modified adjoint interpretation $\sigma^*_\omega I$ the original $[x:=c()]p(x) \leftrightarrow p(c())$ now has exactly the same meaning that the substituted formula $[x:=x^2-1]x \geq 0 \leftrightarrow x^2-1 \geq 0$ has in I.

Since the exact details of this construction are inconsequential for the purposes of this textbook, we refer to previous work [8] for a precise construction of the adjoint interpretation $\sigma^*_\omega I$ for I, ω in this way. The only important point is that adjoint interpretations enable the following uniform substitution lemma, whose proof can be found in previous work [8, Lemma 24].

Lemma 18.3 (Uniform substitution for formulas). *The uniform substitution σ and its adjoint interpretation $\sigma^*_\omega I$ for I, ω have the same semantics for all* formulas *ϕ:*

$$\omega \in I[\![\sigma(\phi)]\!] \text{ iff } \omega \in \sigma^*_\omega I[\![\phi]\!]$$

18.5.5 Soundness

Equipped with the uniform substitution lemma, which equates the semantics of a uniform substitute with the semantics of the original in an adjoint interpretation, it is now easy to establish the soundness of proof rule US (Theorem 18.1). Of course, the uniform substitution proof rule US is only applicable if its uniform substitution is defined, so respects its admissibility conditions.

Theorem 18.1 (Uniform substitution). *The proof rule US is sound:*

$$\text{US}\ \frac{\phi}{\sigma(\phi)}$$

Proof. The proof [8] uses that truth of the substituted formula is equivalent to truth of the original formula in the adjoint interpretation to conclude that validity of the premise in all interpretations implies validity in the adjoint interpretation so validity of the conclusion. Let the premise ϕ of rule US be valid, i.e., $\omega \in I[\![\phi]\!]$ for all states ω and for all interpretations I of the program, predicate, and function symbols. To show that the conclusion is valid, consider any state ω and any interpretation I and show that $\omega \in I[\![\sigma(\phi)]\!]$. By Lemma 18.3, the uniformly substituted formula $\sigma(\phi)$ is true in state ω of interpretation I iff the original formula ϕ is true in state ω of the adjoint interpretation $\sigma^*_\omega I$ that has already been modified according to the substitution σ, that is $\omega \in I[\![\sigma(\phi)]\!]$ iff $\omega \in \sigma^*_\omega I[\![\phi]\!]$. Now $\omega \in \sigma^*_\omega I[\![\phi]\!]$ holds, because $\omega \in I[\![\phi]\!]$ for all states ω and interpretations I, including for state ω and interpretation $\sigma^*_\omega I$, by premise. □

The other missing ingredient for the uniform substitution proof rule US is the exact definition of the free and bound variables, which is needed in the definition of admissibility (Definition 18.7). Those were already reported in Sect. 5.6.6. The only addition is the definition of free and bound variables for the newly added function and predicate symbols as well as program constant symbols. For function and predicate symbols this is just a matter of asking the argument terms:

$$\begin{aligned}\text{FV}(f(e_1,\ldots,e_k)) &= \text{FV}(e_1)\cup\cdots\cup\text{FV}(e_k)\\ \text{FV}(p(e_1,\ldots,e_k)) &= \text{FV}(e_1)\cup\cdots\cup\text{FV}(e_k)\\ \text{BV}(p(e_1,\ldots,e_k)) &= \emptyset\end{aligned}$$

The interpretations of a program constant symbol a can read and write any variable from the set $\mathscr{V}$ of all variables but is not guaranteed to write any particular variable so has no must-bound variables:

$$\begin{aligned}\text{FV}(a) &= \mathscr{V}\\ \text{BV}(a) &= \mathscr{V}\\ \text{MBV}(a) &= \emptyset\end{aligned}$$

18.6 Axiomatic Proof Calculus for dL

A purely axiomatic formulation of the differential dynamic logic axiomatization [8] is shown in Fig. 18.2. The axioms listed in Fig. 18.2 are axioms, so concrete dL for-

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

$$[?]\ [?q]p \leftrightarrow (q \rightarrow p)$$

$$[\cup]\ [a \cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \wedge [b]p(\bar{x})$$

$$[;]\ [a;b]p(\bar{x}) \leftrightarrow [a][b]p(\bar{x})$$

$$[^*]\ [a^*]p(\bar{x}) \leftrightarrow p(\bar{x}) \wedge [a][a^*]p(\bar{x})$$

$$\langle\cdot\rangle\ \langle a\rangle p(\bar{x}) \leftrightarrow \neg[a]\neg p(\bar{x})$$

$$K\ [a](p(\bar{x}) \rightarrow q(\bar{x})) \rightarrow ([a]p(\bar{x}) \rightarrow [a]q(\bar{x}))$$

$$I\ [a^*]p(\bar{x}) \leftrightarrow p(\bar{x}) \wedge [a^*](p(\bar{x}) \rightarrow [a]p(\bar{x}))$$

$$V\ p \rightarrow [a]p$$

$$G\ \frac{p(\bar{x})}{[a]p(\bar{x})}$$

$$\forall\ \frac{p(x)}{\forall x\, p(x)}$$

$$MP\ \frac{p \rightarrow q \quad p}{q}$$

Fig. 18.2 Differential dynamic logic axioms and proof rules

mulas, and not axiom schemata that stand for an infinite collection of formulas. The axioms are *sound*, i.e., valid dL formulas. Besides the cases we already discussed so far, these axioms are formed by using program constant symbols a and b as concrete hybrid programs and by using $p(\bar{x})$ as a concrete formula for the postconditions that have no admissibility requirement. During uniform substitution with rule US, those program constant symbols a and b and the formulas $p(\bar{x})$ and $q(\bar{x})$ in the axioms can, in turn, be substituted with arbitrary HPs and dL formulas, respectively.

The only exception is the test axiom [?], which might have been phrased as either of the following two dL formulas:

$$[?q]p \leftrightarrow (q \rightarrow p) \tag{18.9}$$

$$[?q(\bar{x})]p(\bar{x}) \leftrightarrow (q(\bar{x}) \rightarrow p(\bar{x})) \tag{18.10}$$

It looks as if the second formulation (18.10) would be more flexible, because its explicit mention of the list of all variables in $p(\bar{x})$ and $q(\bar{x})$ make it obvious that the axiom can be instantiated with any arbitrary dL formulas for the test $?q(\bar{x})$ and postcondition $p(\bar{x})$. However, the first formulation (18.9) is sufficient, because any arbitrary dL formulas can already be substituted in for the arity 0 predicate symbols p and q as well, since no variables are bound anywhere in (18.10), so its intersection with any arbitrary set of free variables of any substitution will always be empty.

Soundness of the axioms and proof rules in Fig. 18.2 follows from soundness of the corresponding axiom schemata and proof rule schemata in Chap. 5 and Chap. 7 from Part I of this textbook. The concrete axioms in Fig. 18.2 are instances of the previous axiom schemata, even if their soundness would have been easier to prove directly [8]. Implementing the axioms of Fig. 18.2 in a theorem prover is now straightforward, because each axiom is just a single concrete dL formula that the

prover needs to remember. It can be shown that the uniform substitution proof rule US can prove all instances of these axioms that are required for completeness [8].

18.7 Differential Axioms

The axiomatic approach discussed in this chapter is not limited to logically internalizing the CPS reasoning principles from Part I but works equally well elsewhere, including the proof principles for differential equations from Part II. The key ingredient enabling such an approach for differential equations is the differential forms that we have already gotten to know in Part II. A purely axiomatic formulation of the differential equation axioms and axioms for differentials of dL [8] is shown in Fig. 18.3. These axioms are special instances of the axiom schemata from Part II, which explains their soundness.

DW $[x' = f(x)\,\&\,q(x)]p(x) \leftrightarrow [x' = f(x)\,\&\,q(x)](q(x) \rightarrow p(x))$

DI $\big([x' = f(x)\,\&\,q(x)]p(x) \leftrightarrow [?q(x)]p(x)\big) \leftarrow (q(x) \rightarrow [x' = f(x)\,\&\,q(x)](p(x))')$

DC $\big([x' = f(x)\,\&\,q(x)]p(x) \leftrightarrow [x' = f(x)\,\&\,q(x) \wedge r(x)]p(x)\big) \leftarrow [x' = f(x)\,\&\,q(x)]r(x)$

DE $[x' = f(x)\,\&\,q(x)]p(\bar{x}) \leftrightarrow [x' = f(x)\,\&\,q(x)][x' := f(x)]p(\bar{x})$

DG $[x' = f(x)\,\&\,q(x)]p(x) \leftrightarrow \exists y\,[x' = f(x), y' = a(x)\cdot y + b(x)\,\&\,q(x)]p(x)$

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

$+'$ $(f(\bar{x}) + g(\bar{x}))' = (f(\bar{x}))' + (g(\bar{x}))'$

$-'$ $(f(\bar{x}) - g(\bar{x}))' = (f(\bar{x}))' - (g(\bar{x}))'$

$\cdot'$ $(f(\bar{x}) \cdot g(\bar{x}))' = (f(\bar{x}))' \cdot g(\bar{x}) + f(\bar{x}) \cdot (g(\bar{x}))'$

$/'$ $(f(\bar{x})/g(\bar{x}))' = \big((f(\bar{x}))' \cdot g(\bar{x}) - f(\bar{x}) \cdot (g(\bar{x}))'\big)/g(\bar{x})^2$

c' $(c())' = 0$ (for numbers or constants $c()$)

x' $(x)' = x'$ (for variable $x \in \mathcal{V}$)

Fig. 18.3 Differential equation axioms and differential axioms

The structural advantages of uniform substitutions are exploited in the axioms listed in Fig. 18.3. Since the arity 1 function symbols a and b in the differential ghost axiom DG receive argument x, their respective replacements also have special permission to depend on x. Their uniform substitution replacements can, thus, also have free variable x and any other variable but *not* the new differential ghost y, because y is bound by $y' = a(x) \cdot y + b(x)$. The replacements for $a(x)$ and $b(x)$ in

axiom DG can, thus, overall mention any variable other than the differential ghost y. It is crucial for soundness (Chap. 12) that the replacements for $a(x)$ and $b(x)$ do not have free variable y, because the new differential equation $y' = a(x) \cdot y + b(x)$ is not otherwise guaranteed to have a solution of sufficient duration when $y' = a(x) \cdot y + b(x)$ is not actually linear since the replacements for $a(x)$ or $b(x)$ secretly depend on y. Unlike for another variable z, the axiom DG needs to provide special permission in the form $a(x)$ and $b(x)$ to depend on x, because x is bound by $x' = f(x)$.

Observe how much easier it is to establish the soundness of the concrete axiom DG with its concrete mentions of free variables compared to establishing what precise relationships of variable occurrences are soundly acceptable in the schematic instances of DG. The uniform substitution mechanism in the form of rule US takes care of these generalization and instantiation questions once and for all, as opposed to on a case-by-case basis for each axiom schema again.

Looking through the axioms in Fig. 18.3, it is also important that x' is not free in the postcondition $p(x)$ of the differential invariant axiom DI, because x' is guaranteed to equal $f(x)$ in $[x' = f(x) \& q(x)]p(x)$ but not in $[?q(x)]p(x)$. Indeed, uniform substitution maintains this during instantiation, because x' is bound by $x' = f(x) \& q(x)$ so cannot occur in the replacements for the postcondition $p(x)$ without special permission. This is unlike for axiom DW, where the postcondition $p(x)$ also disallows a mention of x' for simplicity even if it would have been perfectly sound, because all occurrences of $p(x)$ are in the scope of $[x' = f(x) \& q(x)]$.

Similarly, it is important for the solution axiom DS to not have x' in the postcondition $p(x)$, because otherwise an additional assignment $[x' := c()]p(x)$ would be needed instead of $p(x)$ on the right-hand side to propagate the effect that the differential equation $x' = c() \& q(x)$ has on x'. Of course, it is even more important for the replacement of the arity 0 constant symbol $c()$ in the differential equation to not have x as a free variable, because $x + c()t$ would not otherwise be the correct solution of the differential equation $x' = c()$. The constant differential equation axiom DS is weaker than the full solution axiom schema $[']$, because it only works for differential equations with constant (symbolic) right-hand side. But axiom DS can be used along with a differential ghost DG to introduce time $t' = 1$ and with differential cuts DC to introduce and then prove by DI the solutions of other solvable differential equations [8] similar to the approach discussed in Chap. 12.

The arity 0 function symbol $c()$ in axiom c' cannot be substituted with formulas that mention variables, because, similarly to a quantifier, the differential operator $(\ldots)'$ does not accept the introduction of variables. The reason why the differential operator $(\ldots)'$ does not allow any new variables to be introduced during uniform substitution is that the value of $(xy)'$ equals the value of $x'y + xy'$ and depends on x, x', y, y', which is why it is important to know all free variables of any $(\ldots)'$ term.

In particular, the arity 0 function symbol $c()$ in axiom c' can be replaced by constant terms like $5 \cdot 2$ or $5 + b()$ for an arity 0 (constant) function symbol such as $b()$ for braking force, but not by a term like $5 + x$ with a new variable whose differential would indeed depend on the value of variables x and x'.

This is to be contrasted with axiom $+'$ whose occurrence of $f(\bar{x})$ and $g(\bar{x})$ can be replaced by any arbitrary terms, because they already mention all variables $\bar{x}$,

so no new variables remain to be introduced during uniform substitution. Indeed, differentials act as specified in axioms $+'$,$-'$,$\cdot'$,$/'$ on the arithmetic operations for any terms $f(\bar{x}), g(\bar{x})$, but the differential 0 as specified in axiom c' only applies to terms that are actually constant and cannot have any free variables. Uniform substitutions make it very easy to distinguish between the two cases just by the syntactic expressions used in the respective concrete axiom formulas.

18.8 Summary

The main insight of this chapter is that uniform substitutions provide a simple and modular way of implementing differential dynamic logic reasoning for hybrid systems. Based on a straightforward recursive implementation of uniform substitution, the soundness-critical part of a theorem prover reduces to mere copy-and-paste of the concrete formulas adopted as axioms. The resulting proof calculus continues to be sound and complete relative to any differentially expressive logic [8], including first-order logic of differential equations [5, 6] and discrete dynamic logic [6].

Theorem 18.2 (Axiomatization of dL). *The uniform substitution* dL *calculus listed in Figs. 18.2 and 18.3 is a* sound and complete axiomatization *of hybrid systems relative to* any *differentially expressive logic L, i.e., every valid* dL *formula is provable in the* dL *calculus from L tautologies.*

This succinct approach explains the small soundness-critical core of the uniform substitution prover KeYmaera X [3] and why it was relatively easy to cross-verify it [1] both in Isabelle/HOL [4] and in Coq [10]. In fact, with a minor generalization of the set of symbols that uniform substitutions can instantiate, it is easy to derive the contextual equivalence rewriting rules (Lemma 6.2) from uniform substitution [8] as well, which are featured prominently to apply axioms in context. The Appendix explores that all other proof rules of dL are not schematic but *axiomatic proof rules* consisting of concrete dL formulas instantiated by uniform substitutions [8].

18.9 Appendix: Uniform Substitution of Rules and Proofs

Uniform substitutions are not limited to be used on axioms, but can also be used on proof rules

$$\frac{\phi_1 \quad \ldots \quad \phi_n}{\psi}$$

or entire proofs that conclude ψ from the premises ϕ_1 to ϕ_n (likewise for sequents). We just need to use the same uniform substitution on the premises that we use for the conclusion. By Lemma 18.3, using the same uniform substitution everywhere semantically corresponds to fixing and using the same interpretation I everywhere.

An inference or proof rule is *locally sound* iff its conclusion is valid in any interpretation I in which all its premises are valid. All locally sound proof rules are sound, because if all premises are valid in all interpretations, then local soundness makes the conclusion valid in each of the interpretations. But locally sound proof rules can also be soundly substituted uniformly, which preserves local soundness.

Theorem 18.3 (Uniform substitution of rules). *All uniform substitution instances (with $FV(\sigma)=\emptyset$) of locally sound inferences are* locally sound*:*

$$\frac{\phi_1 \quad \ldots \quad \phi_n}{\psi} \text{ locally sound} \qquad \text{implies} \qquad \frac{\sigma(\phi_1) \quad \ldots \quad \sigma(\phi_n)}{\sigma(\psi)} \text{ locally sound}$$

The idea behind the proof of Theorem 18.3 [8] is that, by Lemma 18.3, the truth of the right premises $\sigma(\phi_i)$ in a state ω and interpretation I is equivalent to the truth of the corresponding left premises ϕ_i in ω and adjoint interpretation $\sigma^*_\omega I$. By local soundness of the left inference, if all premises ϕ_i are valid in interpretation $\sigma^*_\omega I$, then so is its conclusion ψ, which, by Lemma 18.3, implies that the substituted conclusion $\sigma(\psi)$ is valid in I. The assumption that $\mathrm{FV}(\sigma)=\emptyset$ is used to ensure that the same argument works in one adjoint interpretation $\sigma^*_\omega I$ regardless of the state ω. If $n=0$ so that ψ has a proof, then this theorem also holds when $\mathrm{FV}(\sigma)\neq\emptyset$, since soundness and local soundness are equivalent notions for $n=0$ premises.

Theorem 18.3 explains how all proof rules of dL (except US) are *axiomatic proof rules* that are merely pairs of concrete dL formulas. For example, generalization rule

$$\mathrm{G}\ \frac{p(\bar{x})}{[a]p(\bar{x})}$$

is a pair of concrete dL formulas. Rule G can be instantiated with Theorem 18.3 to:

$$\frac{x^2\geq 0}{[x:=x+1;(x'=x\cup x'=-2)]x^2\geq 0}$$

using the uniform substitution

$$\sigma=\{a\mapsto x:=x+1;(x'=x\cup x'=-2),\ p(\bar{x})\mapsto x^2\geq 0\}$$

All of a sudden, the only proof rule that needs an implementation as an algorithm is the uniform substitution mechanism itself that is used in rule US and Theorem 18.3. All other axioms and axiomatic proof rules are just concrete data.

Exercises

18.1. Give the result of applying the uniform substitution rule US with substitution $\sigma=\{a\mapsto\{x''=-g\,\&\,x\geq 0\},\ b\mapsto ?(x=0);v:=-cv,\ p(\bar{x})\mapsto 2gx\leq 2gH-v^2\}$ to

the following formulas, respectively:

$$
\begin{aligned}
[a \cup b]p(\bar{x}) &\leftrightarrow [a]p(\bar{x}) \wedge [b]p(\bar{x}) \\
[a;b]p(\bar{x}) &\leftrightarrow [a][b]p(\bar{x}) \\
[a^*]p(\bar{x}) &\leftrightarrow p(\bar{x}) \wedge [a][a^*]p(\bar{x}) \\
\langle a \rangle p(\bar{x}) &\leftrightarrow \neg[a]\neg p(\bar{x}) \\
[a^*]p(\bar{x}) &\leftrightarrow p(\bar{x}) \wedge [a^*](p(\bar{x}) \rightarrow [a]p(\bar{x}))
\end{aligned}
$$

18.2 (Clash or not). The uniform substitution proof rule US checks that the substitution σ has no replacements that would introduce a free variable in a context where that variable is bound. List the conclusion that rule US produces when being applied with the given substitution on the following examples or explain why and how US clashes, and explain whether it is soundness-critical that US clashes:

$$
\begin{aligned}
&[x := c()]p(x) \leftrightarrow p(c()) && \sigma = \{c() \mapsto 0,\ p(\cdot) \mapsto (\cdot = x)\} \\
&[x := c()]p(x) \leftrightarrow p(c()) && \sigma = \{c() \mapsto y{+}1,\ p(\cdot) \mapsto [y := 1; (y := \cdot)^*]y \leq 1\} \\
&[x' = c()]p(x) \leftrightarrow \forall t \geq 0\, [x := x + t \cdot c()]p(x) && \sigma = \{c() \mapsto -x,\ p(\cdot) \mapsto (\cdot \geq 0)\}
\end{aligned}
$$

18.3 (Make it clash). Let p an arity 0 predicate symbol. Give a uniform substitution σ for which it is necessary for soundness that US clashes when being applied to

$$p \rightarrow [a]p$$

Can you also give a uniform substitution that clashes when applied to the following?

$$[a;b]p(\bar{x}) \leftrightarrow [a][b]p(\bar{x})$$

18.4. Give the result of applying uniform substitution rule US with substitution $\sigma = \{c() \mapsto x \cdot y^2 + 1,\ p(\cdot) \mapsto (y + \cdot \geq z)\}$ on the following formulas or explain why and how US clashes:

$$
\begin{aligned}
&[u := c()]p(u) \leftrightarrow p(c()) \\
&[x := c()]p(x) \leftrightarrow p(c()) \\
&[y := c()]p(y) \leftrightarrow p(c()) \\
&[z := c()]p(z) \leftrightarrow p(c()) \\
&[u := c()]p(u) \leftrightarrow \forall u\, (u = c() \rightarrow p(u)) \\
&[x := c()]p(x) \leftrightarrow \forall x\, (x = c() \rightarrow p(x)) \\
&[y := c()]p(y) \leftrightarrow \forall y\, (y = c() \rightarrow p(y)) \\
&[z := c()]p(z) \leftrightarrow \forall z\, (z = c() \rightarrow p(z))
\end{aligned}
$$

If σ clashes, give a "similar" uniform substitution that would not clash.

18.5. What is the result of applying rule US to the [:=] axiom with the substitution $\sigma = \{c() \mapsto x + y,\ p(\cdot) \mapsto [z := \cdot + 1; (z := z + \cdot)^* \cdot + 1 \geq 0]\}$? For each of the follow-

ing formulas, either say which uniform substitution proves it by rule US from axiom [:=], or explain why no such uniform substitution exists.

$$[x:=-x]x^2 \geq 2x \leftrightarrow (-x)^2 \geq 2(-x)$$
$$[x:=y+1][(z:=z+x)^*]x^2 \geq z \leftrightarrow [(z:=z+y+1)^*](y+1)^2 \geq z$$
$$[x:=2x][(z:=z+x)^*]x^2 \geq z \leftrightarrow [(z:=z+2x)^*](2x)^2 \geq z$$
$$[x:=2x][(z:=z+x;z:=z+x)^*]x^2 \geq z \leftrightarrow [(z:=z+2x;z:=z+x)^*](2x)^2 \geq z$$
$$[x:=z+1][(z:=z+x)^*]x^2 \geq z \leftrightarrow [(z:=z+z+1)^*](z+1)^2 \geq z$$
$$[x:=z][x'=2x]x \geq 0 \leftrightarrow [z'=2z]z \geq 0$$
$$[x:=z+1][x'=2x]x \geq 0 \leftrightarrow [z'=2z]z \geq 0$$

18.6 (Local soundness). Theorem 18.3 shows that locally sound proof rules can be uniformly substituted. Show that all proof rules of dL except US are locally sound.

18.7 (Renaming). Uniform substitutions are perfect for substituting formulas for predicate symbols, terms for function symbols, and programs for program constant symbols. Yet, no matter how many uniform substitutions we try to use on the assignment axiom [:=], it will always be the variable x that it assigns to. The only other axioms that even mention variable names are the differential variable axiom x', the quantifier generalization rule $\forall$, and the differential equation axioms (which resolve this question by generalizing to systems of differential equations, however).

In order to prove instances of these axioms with other variable names, it would be helpful to have a proof rule for renaming. Renaming can be done in at least two different ways. Uniform renaming renames a variable x to a variable y uniformly everywhere. Bound renaming only renames one bound occurrence of a variable x to y (and, of course, consistently renames occurrences of x within the scope of this bound occurrence to y), which can be used to prove $\forall x\, p(x) \leftrightarrow \forall y\, p(y)$. Uniform renaming, for example, proves

$$\text{UR}\frac{x \geq 0 \land \forall x\,(x^2 \geq 0) \rightarrow [x:=x+1]x > 0}{y \geq 0 \land \forall y\,(y^2 \geq 0) \rightarrow [y:=y+1]y > 0}$$

Bound renaming, instead, proves

$$\text{BR}\frac{x \geq 0 \land \forall x\,(x^2 \geq 0) \rightarrow [x:=x+1]x > 0}{x \geq 0 \land \forall x\,(x^2 \geq 0) \rightarrow [y:=x+1]y > 0}$$

Give a precise construction defining both styles of renaming proof rules and carefully identify all requirements for soundness. If you are up for a challenge, prove both rules sound.

References

[1] Brandon Bohrer, Vincent Rahli, Ivana Vukotic, Marcus Völp, and André Platzer. Formally verified differential dynamic logic. In: *Certified Programs and Proofs - 6th ACM SIGPLAN Conference, CPP 2017, Paris, France, January 16-17, 2017*. Ed. by Yves Bertot and Viktor Vafeiadis. New York: ACM, 2017, 208–221. DOI: 10.1145/3018610.3018616.

[2] Alonzo Church. *Introduction to Mathematical Logic*. Princeton: Princeton University Press, 1956.

[3] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. KeYmaera X: an axiomatic tactical theorem prover for hybrid systems. In: *CADE*. Ed. by Amy Felty and Aart Middeldorp. Vol. 9195. LNCS. Berlin: Springer, 2015, 527–538. DOI: 10.1007/978-3-319-21401-6_36.

[4] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Vol. 2283. LNCS. Berlin: Springer, 2002.

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

[6] André Platzer. The complete proof theory of hybrid systems. In: *LICS*. Los Alamitos: IEEE, 2012, 541–550. DOI: 10.1109/LICS.2012.64.

[7] André Platzer. A uniform substitution calculus for differential dynamic logic. In: *CADE*. Ed. by Amy Felty and Aart Middeldorp. Vol. 9195. LNCS. Berlin: Springer, 2015, 467–481. DOI: 10.1007/978-3-319-21401-6_32.

[8] André Platzer. A complete uniform substitution calculus for differential dynamic logic. *J. Autom. Reas.* **59**(2) (2017), 219–265. DOI: 10.1007/s10817-016-9385-1.

[9] André Platzer and Jan-David Quesel. KeYmaera: a hybrid theorem prover for hybrid systems. In: *IJCAR*. Ed. by Alessandro Armando, Peter Baumgartner, and Gilles Dowek. Vol. 5195. LNCS. Berlin: Springer, 2008, 171–178. DOI: 10.1007/978-3-540-71070-7_15.

[10] The Coq development team. *The Coq proof assistant reference manual*. Version 8.0. LogiCal Project. 2004.