A Uniform Substitution Calculus for Differential Dynamic Logic

André Platzer(✉)

Computer Science Department, Carnegie Mellon University, Pittsburgh, USA
aplatzer@cs.cmu.edu

Abstract. This paper introduces a new proof calculus for *differential dynamic logic* (d$\mathcal{L}$) that is entirely based on *uniform substitution*, a proof rule that substitutes a formula for a predicate symbol everywhere. Uniform substitutions make it possible to rely on *axioms* rather than axiom schemata, substantially simplifying implementations. Instead of subtle schema variables and soundness-critical side conditions on the occurrence patterns of variables, the resulting calculus adopts only a finite number of ordinary d$\mathcal{L}$ formulas as axioms. The static semantics of differential dynamic logic is captured exclusively in uniform substitutions and bound variable renamings as opposed to being spread in delicate ways across the prover implementation. In addition to sound uniform substitutions, this paper introduces *differential forms* for differential dynamic logic that make it possible to internalize differential invariants, differential substitutions, and derivations as first-class axioms in d$\mathcal{L}$.

1 Introduction

Differential dynamic logic (d$\mathcal{L}$) [4,6] is a logic for proving correctness properties of hybrid systems. It has a sound and complete proof calculus relative to differential equations [4,6] and a sound and complete proof calculus relative to discrete systems [6]. Both sequent calculi [4] and Hilbert-type axiomatizations [6] have been presented for d$\mathcal{L}$ but only the former has been implemented. The implementation of d$\mathcal{L}$'s sequent calculus in KeYmaera makes it straightforward for users to prove properties of hybrid systems, because it provides rules performing natural decompositions for each operator. The downside is that the implementation of the rule schemata and their side conditions on occurrence constraints and relations of reading and writing of variables as well as rule applications in context is nontrivial and inflexible in KeYmaera.

The goal of this paper is to identify how to make it straightforward to implement the axioms and proof rules of differential dynamic logic by writing down a finite list of *axioms* (concrete formulas, not axiom schemata that represent an infinite list of axioms subject to sophisticated soundness-critical schema variable matching implementations). They require multiple axioms to be combined with

All proofs are in a companion report [9]. This material is based upon work supported by the National Science Foundation by NSF CAREER Award CNS-1054246.

A.P. Felty and A. Middeldorp (Eds.): CADE-25, LNAI 9195, pp. 467–481, 2015.
DOI: 10.1007/978-3-319-21401-6_32

one another to obtain the effect that a user would want for proving a hybrid system conjecture. This paper argues that this is still a net win for hybrid systems, because a substantially simpler prover core is easier to implement correctly, and the need to combine multiple axioms to obtain user-level proof steps can be achieved equally well by appropriate tactics, which are not soundness-critical.

To achieve this goal, this paper follows observations for differential game logic [8] that highlight the significance and elegance of *uniform substitutions*, a classical proof rule for first-order logic [2, §35,40]. Uniform substitutions uniformly instantiate predicate and function symbols with formulas and terms, respectively, as functions of their arguments. In the presence of the nontrivial binding structure that nondeterminism and differential equations of hybrid programs induce for the dynamic modalities of differential dynamic logic, flexible but sound uniform substitutions become more complex for $\mathsf{d}\mathcal{L}$, but can still be read off elegantly from its static semantics. In fact, $\mathsf{d}\mathcal{L}$'s static semantics is solely captured[1] in the implementation of uniform substitution (and bound variable renaming), thereby leading to a completely modular proof calculus.

This paper introduces a static and dynamic semantics for *differential-form* $\mathsf{d}\mathcal{L}$, proves coincidence lemmas and uniform substitution lemmas, culminating in a soundness proof for uniform substitutions (Sect. 3). It exploits the new *differential forms* that this paper adds to $\mathsf{d}\mathcal{L}$ for internalizing differential invariants [5], differential cuts [5,7], differential ghosts [7], differential substitutions, total differentials and Lie-derivations [5,7] as first-class citizens in $\mathsf{d}\mathcal{L}$, culminating in entirely modular axioms for differential equations and a superbly modular soundness proof (Sect. 4). This approach is to be contrasted with earlier approaches for differential invariants that were based on complex built-in rules [5,7]. The relationship to related work from previous presentations of differential dynamic logic [4,6] continues to apply except that $\mathsf{d}\mathcal{L}$ now internalizes differential equation reasoning axiomatically via differential forms.

2 Differential-Form Differential Dynamic Logic

2.1 Syntax

Formulas and hybrid programs (HPs) of $\mathsf{d}\mathcal{L}$ are defined by simultaneous induction based on the following definition of terms. Similar simultaneous inductions are used throughout the proofs for $\mathsf{d}\mathcal{L}$. The set of all *variables* is $\mathcal{V}$. For any $V \subseteq \mathcal{V}$ is $V' \overset{\text{def}}{=} \{x' : x \in V\}$ the set of *differential symbols* x' for the variables in V. Function symbols are written f, g, h, predicate symbols p, q, r, and variables $x, y, z \in \mathcal{V}$ with differential symbols $x', y', z' \in \mathcal{V}'$. Program constants are a, b, c.

Definition 1 (Terms). Terms *are defined by this grammar (with* $\theta, \eta, \theta_1, \ldots, \theta_k$ *as terms,* $x \in \mathcal{V}$ *as variable,* $x' \in \mathcal{V}'$ *differential symbol, and* f *function symbol):*

$$\theta, \eta \ ::= \ x \mid x' \mid f(\theta_1, \ldots, \theta_k) \mid \theta + \eta \mid \theta \cdot \eta \mid (\theta)'$$

[1] This approach is dual to other successful ways of solving the intricacies and subtleties of substitutions [1,3] by imposing occurrence side conditions on axiom schemata and proof rules, which is what uniform substitutions can get rid of.

Number literals such as 0,1 are allowed as function symbols without arguments that are always interpreted as the numbers they denote. Beyond differential symbols x', *differential-form* $\mathsf{d}\mathcal{L}$ allows *differentials* $(\theta)'$ of terms θ as terms for the purpose of axiomatically internalizing reasoning about differential equations.

Definition 2 (Hybrid program). Hybrid programs *(HPs) are defined by the following grammar (with α, β as HPs, program constant a, variable x, term θ possibly containing x, and formula ψ of first-order logic of real arithmetic):*

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

Assignments $x := \theta$ of θ to variable x, *tests* $?\psi$ of the formula ψ in the current state, *differential equations* $x' = \theta \,\&\, \psi$ restricted to the evolution domain constraint ψ, *nondeterministic choices* $\alpha \cup \beta$, *sequential compositions* $\alpha; \beta$, and *nondeterministic repetition* α^* are as usual in $\mathsf{d}\mathcal{L}$ [4,6]. The effect of the *differential assignment* $x' := \theta$ to differential symbol x' is similar to the effect of the assignment $x := \theta$ to variable x, except that it changes the value of the differential symbol x' around instead of the value of x. It is not to be confused with the differential equation $x' = \theta$, which will follow said differential equation continuously for an arbitrary amount of time. The differential assignment $x' := \theta$, instead, only assigns the value of θ to the differential symbol x' discretely once at an instant of time. Program constants a are uninterpreted, i.e. their behavior depends on the interpretation in the same way that the values of function symbols f and predicate symbols p depends on their interpretation.

Definition 3 ($\mathsf{d}\mathcal{L}$ formula). *The* formulas of (differential-form) differential dynamic logic *($\mathsf{d}\mathcal{L}$) are defined by the grammar (with $\mathsf{d}\mathcal{L}$ formulas ϕ, ψ, terms $\theta, \eta, \theta_1, \ldots, \theta_k$, predicate symbol p, quantifier symbol C, variable x, HP α):*

$$\phi, \psi \ ::= \ \theta \geq \eta \mid p(\theta_1, \ldots, \theta_k) \mid C(\phi) \mid \neg\phi \mid \phi \wedge \psi \mid \forall x\, \phi \mid \exists x\, \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi$$

Operators $>, \leq, <, \vee, \rightarrow, \leftrightarrow$ are definable, e.g., $\phi \rightarrow \psi$ as $\neg(\phi \wedge \neg\psi)$. Likewise $[\alpha]\phi$ is equivalent to $\neg\langle\alpha\rangle\neg\phi$ and $\forall x\, \phi$ equivalent to $\neg\exists x\, \neg\phi$. The modal formula $[\alpha]\phi$ expresses that ϕ holds after all runs of α, while the dual $\langle\alpha\rangle\phi$ expresses that there is a run of α after which ϕ holds. *Quantifier symbols* C (with formula ϕ as argument), i.e. higher-order predicate symbols that bind all variables of ϕ, are unnecessary but internalize contextual congruence reasoning efficiently.

2.2 Dynamic Semantics

A state is a mapping from variables $\mathcal{V}$ and differential symbols $\mathcal{V}'$ to $\mathbb{R}$. The set of states is denoted $\mathcal{S}$. Let ν_x^r denote the state that agrees with state ν except for the value of variable x, which is changed to $r \in \mathbb{R}$, and accordingly for $\nu_{x'}^r$. The interpretation of a function symbol f with arity n (i.e. with n arguments) is a smooth function $I(f) : \mathbb{R}^n \to \mathbb{R}$ of n arguments.

Definition 4 (Semantics of terms). *For each interpretation I, the* semantics of a term θ *in a state $\nu \in \mathcal{S}$ is its value in $\mathbb{R}$. It is defined inductively as follows*

1. $[\![x]\!]^I\nu = \nu(x)$ *for variable* $x \in \mathcal{V}$
2. $[\![x']\!]^I\nu = \nu(x')$ *for differential symbol* $x' \in \mathcal{V}'$
3. $[\![f(\theta_1,\ldots,\theta_k)]\!]^I\nu = I(f)\big([\![\theta_1]\!]^I\nu,\ldots,[\![\theta_k]\!]^I\nu\big)$ *for function symbol* f
4. $[\![\theta+\eta]\!]^I\nu = [\![\theta]\!]^I\nu + [\![\eta]\!]^I\nu$
5. $[\![\theta\cdot\eta]\!]^I\nu = [\![\theta]\!]^I\nu \cdot [\![\eta]\!]^I\nu$
6. $[\![(\theta)']\!]^I\nu = \sum\limits_x \nu(x')\dfrac{\partial[\![\theta]\!]^I}{\partial x}(\nu) = \sum\limits_x \nu(x')\dfrac{\partial[\![\theta]\!]^I\nu^X_x}{\partial X}$

Time-derivatives are undefined in an isolated state ν. The clou is that differentials can still be given a local semantics: $[\![(\theta)']\!]^I\nu$ is the sum of all (analytic) spatial partial derivatives of the value of θ by all variables x (or rather their values X) multiplied by the corresponding tangent described by the value $\nu(x')$ of differential symbol x'. That sum over all variables $x \in \mathcal{V}$ has finite support, because θ only mentions finitely many variables x and the partial derivative by variables x that do not occur in θ is 0. The spatial derivatives exist since $[\![\theta]\!]^I\nu$ is a composition of smooth functions, so smooth. Thus, the semantics of $[\![(\theta)']\!]^I\nu$ is the *differential*[2] of (the value of) θ, hence a differential one-form giving a real value for each tangent vector (i.e. vector field) described by the values $\nu(x')$. The values $\nu(x')$ of the differential symbols x' describe an arbitrary tangent vector or vector field. Along the flow of (the vector field of a) differential equation, though, the value of the differential $(\theta)'$ coincides with the analytic time-derivative of θ (Lemma 8). The interpretation of predicate symbol p with arity n is an n-ary relation $I(p) \subseteq \mathbb{R}^n$. The interpretation of quantifier symbol C is a functional $I(C)$ mapping subsets $M \subseteq \mathcal{S}$ to subsets $I(C)(M) \subseteq \mathcal{S}$.

Definition 5 (d$\mathcal{L}$ semantics). *The* semantics of a d$\mathcal{L}$ formula ϕ*, for each interpretation* I *with a corresponding set of states* $\mathcal{S}$*, is the subset* $[\![\phi]\!]^I \subseteq \mathcal{S}$ *of states in which* ϕ *is true. It is defined inductively as follows*

1. $[\![\theta \geq \eta]\!]^I = \{\nu \in \mathcal{S} : [\![\theta]\!]^I\nu \geq [\![\eta]\!]^I\nu\}$
2. $[\![p(\theta_1,\ldots,\theta_k)]\!]^I = \{\nu \in \mathcal{S} : ([\![\theta_1]\!]^I\nu,\ldots,[\![\theta_k]\!]^I\nu) \in I(p)\}$
3. $[\![C(\phi)]\!]^I = I(C)\big([\![\phi]\!]^I\big)$ *for quantifier symbol* C
4. $[\![\neg\phi]\!]^I = ([\![\phi]\!]^I)^{\complement} = \mathcal{S} \setminus [\![\phi]\!]^I$
5. $[\![\phi \wedge \psi]\!]^I = [\![\phi]\!]^I \cap [\![\psi]\!]^I$
6. $[\![\exists x\, \phi]\!]^I = \{\nu \in \mathcal{S} : \nu^r_x \in [\![\phi]\!]^I$ *for some* $r \in \mathbb{R}\}$
7. $[\![\langle\alpha\rangle\phi]\!]^I = [\![\alpha]\!]^I \circ [\![\phi]\!]^I = \{\nu : \omega \in [\![\phi]\!]^I$ *for some* ω *such that* $(\nu,\omega) \in [\![\alpha]\!]^I\}$
8. $[\![[\alpha]\phi]\!]^I = [\![\neg\langle\alpha\rangle\neg\phi]\!]^I = \{\nu : \omega \in [\![\phi]\!]^I$ *for all* ω *such that* $(\nu,\omega) \in [\![\alpha]\!]^I\}$

A d$\mathcal{L}$ *formula* ϕ *is* valid *in* I*, written* $I \models \phi$*, iff* $[\![\phi]\!]^I = \mathcal{S}$*, i.e.* $\nu \in [\![\phi]\!]^I$ *for all* ν*. Formula* ϕ *is* valid*, written* $\models \phi$*, iff* $I \models \phi$ *for all interpretations* I*.*

The interpretation of a program constant a is a state-transition relation $I(a) \subseteq \mathcal{S} \times \mathcal{S}$, where $(\nu,\omega) \in I(a)$ iff a can run from initial state ν to final state ω.

[2] A slight abuse of notation rewrites the differential as $[\![(\theta)']\!]^I = d[\![\theta]\!]^I = \sum_{i=1}^n \frac{\partial[\![\theta]\!]^I}{\partial x^i}dx^i$ when $x^1,\ldots,x^n$ are the variables in θ and their differentials dx^i form the basis of the cotangent space, which, when evaluated at a point ν whose values $\nu(x')$ determine the tangent vector alias vector field, coincides with Definition 4.

Definition 6 (Transition semantics of HPs). *For each interpretation I, each HP α is interpreted semantically as a binary transition relation $[\![\alpha]\!]^I \subseteq \mathcal{S}\times\mathcal{S}$ on states, defined inductively by*

1. $[\![a]\!]^I = I(a)$ *for program constants* a
2. $[\![x := \theta]\!]^I = \{(\nu, \nu^r_x) : r = [\![\theta]\!]^I\nu\} = \{(\nu,\omega) : \omega = \nu \text{ except } [\![x]\!]^I\omega = [\![\theta]\!]^I\nu\}$
3. $[\![x' := \theta]\!]^I = \{(\nu, \nu^r_{x'}) : r = [\![\theta]\!]^I\nu\} = \{(\nu,\omega) : \omega = \nu \text{ except } [\![x']\!]^I\omega = [\![\theta]\!]^I\nu\}$
4. $[\![?\psi]\!]^I = \{(\nu,\nu) : \nu \in [\![\psi]\!]^I\}$
5. $[\![x' = \theta\,\&\,\psi]\!]^I = \{(\nu,\omega) : I, \varphi, \models x' = \theta \wedge \psi$, *i.e.* $\varphi(\zeta) \in [\![x' = \theta \wedge \psi]\!]^I$ *for all* $0 \le \zeta \le r$, *for some function* $\varphi : [0, r] \to \mathcal{S}$ *of some duration* r *for which all* $\varphi(\zeta)(x') = \frac{\mathrm{d}\varphi(t)(x)}{\mathrm{d}t}(\zeta)$ *exist and* $\nu = \varphi(0)$ *on* $\{x'\}^{\complement}$ *and* $\omega = \varphi(r)\}$*; i.e., φ solves the differential equation and satisfies ψ at all times. In case $r = 0$, the only condition is that $\nu = \omega$ on $\{x'\}^{\complement}$ and $\omega(x') = [\![\theta]\!]^I\omega$ and $\omega \in [\![\psi]\!]^I$.*
6. $[\![\alpha \cup \beta]\!]^I = [\![\alpha]\!]^I \cup [\![\beta]\!]^I$
7. $[\![\alpha;\beta]\!]^I = [\![\alpha]\!]^I \circ [\![\beta]\!]^I = \{(\nu,\omega) : (\nu,\mu) \in [\![\alpha]\!]^I, (\mu,\omega) \in [\![\beta]\!]^I\}$
8. $[\![\alpha^*]\!]^I = ([\![\alpha]\!]^I)^* = \bigcup\limits_{n\in\mathbb{N}} [\![\alpha^n]\!]^I$ *with* $\alpha^{n+1} \equiv \alpha^n;\alpha$ *and* $\alpha^0 \equiv ?true$

where ρ^* denotes the reflexive transitive closure of relation ρ.

The initial values $\nu(x')$ of differential symbols x' do *not* influence the behavior of $(\nu,\omega) \in [\![x' = \theta\,\&\,\psi]\!]^I$, because they may not be compatible with the time-derivatives for the differential equation, e.g. in $x' := 1; x' = 2$, with a x' mismatch.

Functions and predicates are interpreted by I and are only influenced indirectly by ν through the values of their arguments. So $p(e) \to [x := x + 1]p(e)$ is valid if x is not in e since the change in x does not change whether $p(e)$ is true (Lemma 2). By contrast $p(x) \to [x := x + 1]p(x)$ is invalid, since it is false when $I(p) = \{d : d \le 5\}$ and $\nu(x) = 4.5$. If the semantics of p were to depend on the state ν, then there would be no discernible relationship between the truth-values of p in different states, so not even $p \to [x := x + 1]p$ would be valid.

2.3 Static Semantics

The static semantics of $\mathsf{d}\mathcal{L}$ and HPs defines some aspects of their behavior that can be read off directly from their syntactic structure without running their programs or evaluating their dynamical effects. The most important aspects of the static semantics concern free or bound occurrences of variables. Bound variables x 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 $[x := 1 \cup ?true]$.

The notions of free and bound variables are defined by simultaneous induction in the subsequent definitions: free variables for terms ($\mathrm{FV}(\theta)$), formulas ($\mathrm{FV}(\phi)$), and HPs ($\mathrm{FV}(\alpha)$), as well as bound variables for formulas ($\mathrm{BV}(\phi)$) and for HPs ($\mathrm{BV}(\alpha)$). For HPs, there will be a need to distinguish must-bound variables ($\mathrm{MBV}(\alpha)$) that are bound/written to on all executions of α from (may-)bound variables ($\mathrm{BV}(\alpha)$) which are bound on some (not necessarily all) execution paths of α, such as in $[x := 1 \cup (x := 0; y := x + 1)]$, which has bound variables $\{x, y\}$ but must-bound variables only $\{x\}$, because y is not written to in the first choice.

Definition 7 (Bound variable). *The set* $BV(\phi) \subseteq \mathcal{V} \cup \mathcal{V}'$ *of* bound variables *of* $\mathsf{d}\mathcal{L}$ *formula* ϕ *is defined inductively as*

$$\begin{aligned} BV(\theta \geq \eta) = BV(p(\theta_1, \ldots, \theta_k)) &= \emptyset \\ BV(C(\phi)) &= \mathcal{V} \cup \mathcal{V}' \\ BV(\neg\phi) &= BV(\phi) \\ BV(\phi \wedge \psi) &= BV(\phi) \cup BV(\psi) \\ BV(\forall x\, \phi) = BV(\exists x\, \phi) &= \{x\} \cup BV(\phi) \\ BV([\alpha]\phi) = BV(\langle\alpha\rangle\phi) &= BV(\alpha) \cup BV(\phi) \end{aligned}$$

Definition 8 (Free variable). *The set* $FV(\theta) \subseteq \mathcal{V} \cup \mathcal{V}'$ *of* free variables *of term* θ*, i.e. those that occur in* θ*, is defined inductively as*

$$\begin{aligned} FV(x) &= \{x\} \\ FV(x') &= \{x'\} \\ FV(f(\theta_1, \ldots, \theta_k)) &= FV(\theta_1) \cup \cdots \cup FV(\theta_k) \\ FV(\theta + \eta) = FV(\theta \cdot \eta) &= FV(\theta) \cup FV(\eta) \\ FV((\theta)') &= FV(\theta) \cup FV(\theta)' \end{aligned}$$

The set $FV(\phi)$ *of* free variables *of* $\mathsf{d}\mathcal{L}$ *formula* ϕ*, i.e. all those that occur in* ϕ *outside the scope of quantifiers or modalities binding it, is defined inductively as*

$$\begin{aligned} FV(\theta \geq \eta) &= FV(\theta) \cup FV(\eta) \\ FV(p(\theta_1, \ldots, \theta_k)) &= FV(\theta_1) \cup \cdots \cup FV(\theta_k) \\ FV(C(\phi)) &= \mathcal{V} \cup \mathcal{V}' \\ FV(\neg\phi) &= FV(\phi) \\ FV(\phi \wedge \psi) &= FV(\phi) \cup FV(\psi) \\ FV(\forall x\, \phi) = FV(\exists x\, \phi) &= FV(\phi) \setminus \{x\} \\ FV([\alpha]\phi) = FV(\langle\alpha\rangle\phi) &= FV(\alpha) \cup (FV(\phi) \setminus MBV(\alpha)) \end{aligned}$$

Soundness requires that $\mathrm{FV}([\alpha]\phi)$ is not defined as $\mathrm{FV}(\alpha) \cup (\mathrm{FV}(\phi) \setminus \mathrm{BV}(\alpha))$, otherwise $[x := 1 \cup y := 2]x \geq 1$ would have no free variables, but its truth-value depends on the initial value of x, demanding $\mathrm{FV}([x := 1 \cup y := 2]x \geq 1) = \{x\}$.

The static semantics defines which variables are free so may be read ($\mathrm{FV}(\alpha)$), which are bound ($\mathrm{BV}(\alpha)$) so may be written to somewhere in α, and which are must-bound ($\mathrm{MBV}(\alpha)$) so must be written to on all execution paths of α.

Definition 9 (Bound variable). *The set* $BV(\alpha) \subseteq \mathcal{V} \cup \mathcal{V}'$ *of* bound variables *of HP* α*, i.e. all those that may potentially be written to, is defined inductively:*

$$\begin{aligned} BV(a) &= \mathcal{V} \cup \mathcal{V}' && \text{for program constant } a \\ BV(x := \theta) &= \{x\} \\ BV(x' := \theta) &= \{x'\} \end{aligned}$$

$$\begin{aligned} BV(?\psi) &= \emptyset \\ BV(x' = \theta \,\&\, \psi) &= \{x, x'\} \\ BV(\alpha \cup \beta) = BV(\alpha;\beta) &= BV(\alpha) \cup BV(\beta) \\ BV(\alpha^*) &= BV(\alpha) \end{aligned}$$

Definition 10 (Must-bound variable). *The set* $MBV(\alpha) \subseteq BV(\alpha) \subseteq \mathcal{V} \cup \mathcal{V}'$ *of* must-bound variables *of HP* α*, i.e. all those that must be written to on all paths of* α*, is defined inductively as*

$$\begin{aligned} MBV(a) &= \emptyset && \text{for program constant } a \\ MBV(\alpha) &= BV(\alpha) && \text{for other atomic HPs } \alpha \\ MBV(\alpha \cup \beta) &= MBV(\alpha) \cap MBV(\beta) \\ MBV(\alpha;\beta) &= MBV(\alpha) \cup MBV(\beta) \\ MBV(\alpha^*) &= \emptyset \end{aligned}$$

Definition 11 (Free variable). *The set* $FV(\alpha) \subseteq \mathcal{V} \cup \mathcal{V}'$ *of* free variables *of HP* α*, i.e. all those that may potentially be read, is defined inductively as*

$$\begin{aligned} FV(a) &= \mathcal{V} \cup \mathcal{V}' && \text{for program constant } a \\ FV(x := \theta) = FV(x' := \theta) &= FV(\theta) \\ FV(?\psi) &= FV(\psi) \\ FV(x' = \theta \,\&\, \psi) &= \{x\} \cup FV(\theta) \cup FV(\psi) \\ FV(\alpha \cup \beta) &= FV(\alpha) \cup FV(\beta) \\ FV(\alpha;\beta) &= FV(\alpha) \cup (FV(\beta) \setminus MBV(\alpha)) \\ FV(\alpha^*) &= FV(\alpha) \end{aligned}$$

Unlike x, the left-hand side x' of differential equations is not added to the free variables of $\mathrm{FV}(x' = \theta \,\&\, \psi)$, because its behavior does not depend on the initial value of differential symbols x', only the initial value of x. Free and bound variables are the set of all variables $\mathcal{V}$ and differential symbols $\mathcal{V}'$ for program constants a, because their effect depends on the interpretation I, so may read and write all $\mathrm{FV}(a) = \mathrm{BV}(a) = \mathcal{V} \cup \mathcal{V}'$ but not on all paths $\mathrm{MBV}(a) = \emptyset$. Subsequent results about free and bound variables are, thus, vacuously true when program constants occur. Corresponding observations hold for quantifier symbols.

The static semantics defines which variables are readable or writable. There may not be any run of α in which a variable is read or written to. If $x \notin \mathrm{FV}(\alpha)$, though, then α cannot read the value of x. If $x \notin \mathrm{BV}(\alpha)$, it cannot write to x.

The *signature*, i.e. set of function, predicate, quantifier symbols, and program constants in ϕ is denoted by $\Sigma(\phi)$ (accordingly for terms and programs). It is defined like $\mathrm{FV}(\phi)$ except that all occurrences are free. Variables in $\mathcal{V} \cup \mathcal{V}'$ are interpreted by state ν. The symbols in $\Sigma(\phi)$ are interpreted by interpretation I.

2.4 Correctness of Static Semantics

The following result reflects that HPs have bounded effect: for a variable x to be modified during a run of α, x needs the be a bound variable in *HP* α, i.e.

$x \in \mathrm{BV}(\alpha)$, so that α can write to x. The converse is not true, because α may bind a variable x, e.g. by having an assignment to x, that never actually changes the value of x, such as $x := x$ or because the assignment can never be executed.

Lemma 1 (Bound effect lemma). *If $(\nu, \omega) \in [\![\alpha]\!]^I$, then $\nu = \omega$ on $BV(\alpha)^{\complement}$.*

Similarly, only $\mathrm{BV}(\phi)$ change their value during the evaluation of formulas.

The value of a term only depends on the values of its free variables. When evaluating a term θ in two states ν, $\tilde{\nu}$ that differ widely but agree on the free variables $\mathrm{FV}(\theta)$ of θ, the values of θ in both states coincide. Accordingly for different interpretations I, J that agree on the symbols $\Sigma(\theta)$ that occur in θ. Recall that all proofs and additional examples are in a companion report [9].

Lemma 2 (Coincidence lemma). *If $\nu = \tilde{\nu}$ on $FV(\theta)$ and $I = J$ on $\Sigma(\theta)$, then $[\![\theta]\!]^I \nu = [\![\theta]\!]^J \tilde{\nu}$.*

By a more subtle argument, the values of $\mathsf{d}\mathcal{L}$ formulas also only depend on the values of their free variables. When evaluating $\mathsf{d}\mathcal{L}$ formula ϕ in two states ν, $\tilde{\nu}$ that differ but agree on the free variables $\mathrm{FV}(\phi)$ of ϕ, the (truth) values of ϕ in both states coincide. Lemmas 3 and 4 are proved by simultaneous induction.

Lemma 3 (Coincidence lemma). *If $\nu = \tilde{\nu}$ on $FV(\phi)$ and $I = J$ on $\Sigma(\phi)$, then $\nu \in [\![\phi]\!]^I$ iff $\tilde{\nu} \in [\![\phi]\!]^J$.*

In a sense, the runs of an HP α also only depend on the values of its free variables, because its behavior cannot depend on the values of variables that it never reads. That is, if $\nu = \tilde{\nu}$ on $\mathrm{FV}(\alpha)$ and $(\nu, \omega) \in [\![\alpha]\!]^I$, then there is an $\tilde{\omega}$ such that $(\tilde{\nu}, \tilde{\omega}) \in [\![\alpha]\!]^J$ and ω and $\tilde{\omega}$ agree in some sense. There is a subtlety, though. The resulting states ω and $\tilde{\omega}$ will only continue to agree on $\mathrm{FV}(\alpha)$ and the variables that are bound on the particular path that α took for the transition $(\nu, \omega) \in [\![\alpha]\!]^I$. On variables z that are neither free (so the initial states ν and $\tilde{\nu}$ have not been assumed to coincide) nor bound on the particular path that α took, ω and $\tilde{\omega}$ may continue to disagree, because z has not been written to. Yet, ω and $\tilde{\omega}$ agree on the variables that are bound on *all* paths of α, rather than somewhere in α. That is on the must-bound variables of α. If initial states agree on (at least) all free variables $\mathrm{FV}(\alpha)$ that HP α may read, then the final states agree on those as well as on all variables that α must write, i.e. on $\mathrm{MBV}(\alpha)$.

Lemma 4 (Coincidence lemma). *If $\nu = \tilde{\nu}$ on $V \supseteq FV(\alpha)$ and $I = J$ on $\Sigma(\alpha)$ and $(\nu, \omega) \in [\![\alpha]\!]^I$, then there is an $\tilde{\omega}$ such that $(\tilde{\nu}, \tilde{\omega}) \in [\![\alpha]\!]^J$ and $\omega = \tilde{\omega}$ on $V \cup MBV(\alpha)$.*

3 Uniform Substitutions

The uniform substitution rule US_1 from first-order logic [2, §35,40] substitutes *all* occurrences of predicate $p(\cdot)$ by a formula $\psi(\cdot)$, i.e. it replaces all occurrences of $p(\theta)$, for any (vectorial) term θ, by the corresponding $\psi(\theta)$ simultaneously:

$$(\mathrm{US}_1)\ \frac{\phi}{\phi^{\psi(\cdot)}_{p(\cdot)}} \qquad (\mathrm{US})\ \frac{\phi}{\sigma(\phi)}$$

Rule US_1 [8] requires all relevant substitutions of $\psi(\theta)$ for $p(\theta)$ to be admissible and requires that no $p(\theta)$ occurs in the scope of a quantifier or modality binding a variable of $\psi(\theta)$ other than the occurrences in θ; see [2, §35,40].

This section considers a constructive definition of this proof rule that is more general: US. The $\mathsf{d}\mathcal{L}$ calculus uses uniform substitutions that affect terms, formulas, and programs. A *uniform substitution* σ is a mapping from expressions of the form $f(\cdot)$ to terms $\sigma f(\cdot)$, from $p(\cdot)$ to formulas $\sigma p(\cdot)$, from $C(\_)$ to formulas $\sigma C(\_)$, and from program constants a to HPs σa. Vectorial extensions are accordingly for uniform substitutions of other arities $k \geq 0$. Here $\cdot$ is a reserved function symbol of arity zero and $\_$ a reserved quantifier symbol of arity zero. Figure 1 defines the result $\sigma(\phi)$ of applying to a $\mathsf{d}\mathcal{L}$ formula ϕ the *uniform substitution* σ that uniformly replaces all occurrences of function f by a (instantiated) term and all occurrences of predicate p or quantifier C by a (instantiated) formula as well as of program constant a by a program. 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 according to Fig. 1 (likewise for $\sigma(\theta)$ and $\sigma(\alpha)$). The notation $f \in \sigma$ signifies that σ replaces f, i.e. $\sigma f(\cdot) \neq f(\cdot)$. Finally, σ is a total function when augmented with $\sigma g(\cdot) = g(\cdot)$ for all $g \notin \sigma$. Accordingly for predicate symbols, quantifiers, and program constants.

Definition 12 (Admissible uniform substitution). *The uniform substitution σ is U-*admissible *for ϕ (or θ or α, respectively) with respect to the set $U \subseteq \mathcal{V} \cup \mathcal{V}'$ iff $FV(\sigma|_{\Sigma(\phi)}) \cap U = \emptyset$, where $\sigma|_{\Sigma(\phi)}$ is the restriction of σ that only replaces symbols that occur in ϕ and $FV(\sigma) = \bigcup_{f \in \sigma} FV(\sigma f(\cdot)) \cup \bigcup_{p \in \sigma} FV(\sigma p(\cdot))$ are the* free variables *that σ introduces. The uniform substitution σ is* admissible *for ϕ (or θ or α, respectively) iff all admissibility conditions during its application according to Fig. 1 hold, which check that the bound variables U of each operator are not free in the substitution on its arguments, i.e. σ is U-admissible. Otherwise the substitution clashes so its result $\sigma(\phi)$ ($\sigma(\theta)$ or $\sigma(\alpha)$) is not defined.*

US is only applicable if σ is admissible for ϕ. In all subsequent results, all applications of uniform substitutions are required to be defined (no clash).

3.1 Correctness of Uniform Substitutions

Let I_p^R denote the interpretation that agrees with interpretation I except for the interpretation of predicate symbol p, which is changed to $R \subseteq \mathbb{R}$. Accordingly for predicate symbols of other arities, for function symbols f, and quantifiers C.

Corollary 1 (Substitution adjoints). *The* adjoint interpretation $\sigma_\nu^* I$ *to substitution σ for I, ν is the interpretation that agrees with I except that for each function symbol $f \in \sigma$, predicate symbol $p \in \sigma$, quantifier $C \in \sigma$, and program constant $a \in \sigma$:*

$$
\begin{array}{rll}
\sigma(x) = x & & \text{for variable } x \in \mathcal{V} \\
\sigma(x') = x' & & \text{for differential symbol } x' \in \mathcal{V}' \\
\sigma(f(\theta)) = (\sigma(f))(\sigma(\theta)) \overset{\text{def}}{=} \{\cdot \mapsto \sigma(\theta)\}(\sigma f(\cdot)) & & \text{for function symbol } f \in \sigma \\
\sigma(g(\theta)) = g(\sigma(\theta)) & & \text{for function symbol } g \notin \sigma \\
\sigma(\theta + \eta) = \sigma(\theta) + \sigma(\eta) & & \\
\sigma(\theta \cdot \eta) = \sigma(\theta) \cdot \sigma(\eta) & & \\
\sigma((\theta)') = (\sigma(\theta))' & & \text{if } \sigma\ \mathcal{V} \cup \mathcal{V}'\text{-admissible for } \theta \\
\hline
\sigma(\theta \geq \eta) \equiv \sigma(\theta) \geq \sigma(\eta) & & \\
\sigma(p(\theta)) \equiv (\sigma(p))(\sigma(\theta)) \overset{\text{def}}{\equiv} \{\cdot \mapsto \sigma(\theta)\}(\sigma p(\cdot)) & & \text{for predicate symbol } p \in \sigma \\
\sigma(q(\theta)) \equiv q(\sigma(\theta)) & & \text{for predicate symbol } q \notin \sigma \\
\sigma(C(\phi)) \equiv \sigma(C)(\sigma(\phi)) \overset{\text{def}}{\equiv} \{\_ \mapsto \sigma(\phi)\}(\sigma C(\_)) & & \text{if } \sigma\ \mathcal{V} \cup \mathcal{V}'\text{-admissible for } \phi,\ C \in \sigma \\
\sigma(C(\phi)) \equiv C(\sigma(\phi)) & & \text{if } \sigma\ \mathcal{V} \cup \mathcal{V}'\text{-admissible for } \phi,\ C \notin \sigma \\
\sigma(\neg\phi) \equiv \neg\sigma(\phi) & & \\
\sigma(\phi \wedge \psi) \equiv \sigma(\phi) \wedge \sigma(\psi) & & \\
\sigma(\forall x\, \phi) = \forall x\, \sigma(\phi) & & \text{if } \sigma\ \{x\}\text{-admissible for } \phi \\
\sigma(\exists x\, \phi) = \exists x\, \sigma(\phi) & & \text{if } \sigma\ \{x\}\text{-admissible for } \phi \\
\sigma([\alpha]\phi) = [\sigma(\alpha)]\sigma(\phi) & & \text{if } \sigma\ \mathrm{BV}(\sigma(\alpha))\text{-admissible for } \phi \\
\sigma(\langle\alpha\rangle\phi) = \langle\sigma(\alpha)\rangle\sigma(\phi) & & \text{if } \sigma\ \mathrm{BV}(\sigma(\alpha))\text{-admissible for } \phi \\
\hline
\sigma(a) \equiv \sigma a & & \text{for program constant } a \in \sigma \\
\sigma(b) \equiv b & & \text{for program constant } b \notin \sigma \\
\sigma(x := \theta) \equiv x := \sigma(\theta) & & \\
\sigma(x' := \theta) \equiv x' := \sigma(\theta) & & \\
\sigma(x' = \theta\,\&\,\psi) \equiv x' = \sigma(\theta)\,\&\,\sigma(\psi) & & \text{if } \sigma\ \{x, x'\}\text{-admissible for } \theta, \psi \\
\sigma(?\psi) \equiv ?\sigma(\psi) & & \\
\sigma(\alpha \cup \beta) \equiv \sigma(\alpha) \cup \sigma(\beta) & & \\
\sigma(\alpha; \beta) \equiv \sigma(\alpha); \sigma(\beta) & & \text{if } \sigma\ \mathrm{BV}(\sigma(\alpha))\text{-admissible for } \beta \\
\sigma(\alpha^*) \equiv (\sigma(\alpha))^* & & \text{if } \sigma\ \mathrm{BV}(\sigma(\alpha))\text{-admissible for } \alpha
\end{array}
$$

Fig. 1. Recursive application of uniform substitution σ

$$
\begin{aligned}
&\sigma^*_\nu I(f) : \mathbb{R} \to \mathbb{R};\ d \mapsto [\![\sigma f(\cdot)]\!]^{I^d_\cdot}\nu \\
&\sigma^*_\nu I(p) = \{d \in \mathbb{R} \ :\ \nu \in [\![\sigma p(\cdot)]\!]^{I^d_\cdot}\} \\
&\sigma^*_\nu I(C) : \wp(\mathbb{R}) \to \wp(\mathbb{R});\ R \mapsto [\![\sigma C(\_)]\!]^{I^R_\_} \\
&\sigma^*_\nu I(a) = [\![\sigma a]\!]^I
\end{aligned}
$$

If $\nu = \omega$ on $FV(\sigma)$, then $\sigma^*_\nu I = \sigma^*_\omega I$. If σ is U-admissible for ϕ (or θ or α, respectively) and $\nu = \omega$ on $U^{\complement}$, then

$$
\begin{aligned}
&[\![\theta]\!]^{\sigma^*_\nu I} = [\![\theta]\!]^{\sigma^*_\omega I} \text{ i.e. } [\![\theta]\!]^{\sigma^*_\nu I}\mu = [\![\theta]\!]^{\sigma^*_\omega I}\mu \text{ for all } \mu \\
&[\![\phi]\!]^{\sigma^*_\nu I} = [\![\phi]\!]^{\sigma^*_\omega I} \\
&[\![\alpha]\!]^{\sigma^*_\nu I} = [\![\alpha]\!]^{\sigma^*_\omega I}
\end{aligned}
$$

Substituting equals for equals is sound by the compositional semantics of $\mathsf{d}\mathcal{L}$. The more general uniform substitutions are still sound, because interpretations

of uniform substitutes correspond to interpretations of their adjoints. The semantic modification of adjoint interpretations has the same effect as the syntactic uniform substitution, proved by simultaneous induction.

Lemma 5 (Uniform substitution lemma). *The uniform substitution σ and its adjoint interpretation $\sigma^*_\nu I, \nu$ to σ for I, ν have the same* term *semantics:*

$$[\![\sigma(\theta)]\!]^I\nu = [\![\theta]\!]^{\sigma^*_\nu I}\nu$$

Lemma 6 (Uniform substitution lemma). *The uniform substitution σ and its adjoint interpretation $\sigma^*_\nu I, \nu$ to σ for I, ν have the same* formula *semantics:*

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

Lemma 7 (Uniform substitution lemma). *The uniform substitution σ and its adjoint interpretation $\sigma^*_\nu I, \nu$ to σ for I, ν have the same* program *semantics:*

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

3.2 Soundness

The uniform substitution lemmas are the key insights for the soundness of US. US is only applicable if the uniform substitution is defined (does not clash).

Theorem 1 (Soundness of uniform substitution). *US is sound and so is its special case US_1. That is, if their premise is valid, then so is their conclusion.*

4 Differential Dynamic Logic Axioms

Proof rules and axioms for a Hilbert-type axiomatization of $\mathsf{d}\mathcal{L}$ from prior work [6] are shown in Fig. 2, except that, thanks to rule US, axioms and rules now comprise the finite list of $\mathsf{d}\mathcal{L}$ formulas in Fig. 2 as opposed to an infinite collection of axioms from a finite list of axiom schemata along with schema variables, side conditions, and implicit instantiation rules. Soundness of the axioms in Fig. 2 follows from the soundness of corresponding axiom schemata [6], but would be easier to prove standalone, because it is a finite list of formulas without the need to prove soundness for all their instantiations. The rules in Fig. 2 are *axiomatic rules*, i.e. pairs of concrete formulas instantiated by US. Further, $\bar{x}$ is the vector of all relevant variables, which is finite-dimensional, or, in practice, considered as a built-in vectorial term. Proofs in the uniform substitution $\mathsf{d}\mathcal{L}$ calculus use US (and bound renaming such as $\forall x\, p(x) \leftrightarrow \forall y\, p(y)$) to instantiate the axioms from Fig. 2 to the required form. CT,CQ,CE are congruence rules, which are included for efficiency to use axioms in any context even if not needed for completeness.

Real Quantifiers. Besides (decidable) real arithmetic (whose use is denoted $\mathbb{R}$), complete axioms for first-order logic can be adopted to express universal instantiation $\forall$i, distributivity $\forall\,\rightarrow$, and vacuous quantification $\mathrm{V}_\forall$.

$$\begin{array}{rl} (\forall\mathrm{i}) & (\forall x\, p(x)) \rightarrow p(f) \\ (\forall\rightarrow) & \forall x\, (p(x) \rightarrow q(x)) \rightarrow (\forall x\, p(x) \rightarrow \forall x\, q(x)) \\ (\mathrm{V}_\forall) & p \rightarrow \forall x\, p \end{array}$$

$$\langle\cdot\rangle\ \langle a\rangle p(\bar{x}) \leftrightarrow \neg[a]\neg p(\bar{x})$$
$$[:=]\ [x := f]p(x) \leftrightarrow p(f)$$
$$[?]\ [?q]p \leftrightarrow (q \rightarrow p)$$
$$[\cup]\ [a \cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \wedge [b]p(\bar{x})$$
$$[;]\ [a;b]p(\bar{x}) \leftrightarrow [a][b]p(\bar{x})$$
$$[^*]\ [a^*]p(\bar{x}) \leftrightarrow p(\bar{x}) \wedge [a][a^*]p(\bar{x})$$
$$\mathrm{K}\ [a](p(\bar{x}) \rightarrow q(\bar{x})) \rightarrow ([a]p(\bar{x}) \rightarrow [a]q(\bar{x}))$$
$$\mathrm{I}\ [a^*](p(\bar{x}) \rightarrow [a]p(\bar{x})) \rightarrow (p(\bar{x}) \rightarrow [a^*]p(\bar{x}))$$
$$\mathrm{V}\ p \rightarrow [a]p$$

$$\mathrm{G}\ \frac{p(\bar{x})}{[a]p(\bar{x})} \qquad \forall\ \frac{p(x)}{\forall x\, p(x)} \qquad \mathrm{MP}\ \frac{p \rightarrow q \quad p}{q}$$
$$\mathrm{CT}\ \frac{f(\bar{x}) = g(\bar{x})}{c(f(\bar{x})) = c(g(\bar{x}))} \qquad \mathrm{CQ}\ \frac{f(\bar{x}) = g(\bar{x})}{p(f(\bar{x})) \leftrightarrow p(g(\bar{x}))} \qquad \mathrm{CE}\ \frac{p(\bar{x}) \leftrightarrow q(\bar{x})}{C(p(\bar{x})) \leftrightarrow C(q(\bar{x}))}$$

Fig. 2. Differential dynamic logic axioms and proof rules

The Significance of Clashes. US clashes for substitutions that introduce a free variable into a bound context. Even an occurrence of $p(x)$ in a context where x is bound does not allow mentioning x in the replacement except in the $\cdot$ places:

$$\mathrm{clash}\lightning\ \frac{[x := f]p(x) \leftrightarrow p(f)}{[x := x+1]x \neq x \leftrightarrow x+1 \neq x} \qquad \sigma = \{f \mapsto x+1, p(\cdot) \mapsto (\cdot \neq x)\}$$

US can directly handle even nontrivial binding structures, though, e.g. from [:=] with the substitution $\sigma = \{f \mapsto x^2, p(\cdot) \mapsto [(z := \cdot + z)^*; z := \cdot + yz]y \geq \cdot\}$:

$$\mathrm{US}\ \frac{[x := f]p(x) \leftrightarrow p(f)}{[x := x^2][(z := x+z)^*; z := x+yz]y \geq x \leftrightarrow [(z := x^2+z)^*; z := x^2+yz]y \geq x^2}$$

5 Differential Equations and Differential Axioms

Section 4 leverages the first-order features of $\mathsf{d}\mathcal{L}$ and US to obtain a finite list of axioms without side-conditions. They lack axioms for differential equations, though. Classical calculi for $\mathsf{d}\mathcal{L}$ have axioms for replacing differential equations with a quantifier for time $t \geq 0$ and an assignment for their solutions $\bar{x}(t)$ [4,6]. Besides being limited to simple differential equations, such axioms have the inherent side-condition "if $\bar{x}(t)$ is a solution of the differential equation $x' = \theta$ with symbolic initial value x". Such a side-condition is more difficult than occurrence and read/write conditions, but equally soundness-critical. This section leverages US and the new differential forms in $\mathsf{d}\mathcal{L}$ to obtain a logically internalized version of differential invariants and related proof rules for differential equations [5,7] as axioms (without schema variables and free of side-conditions). These axioms can prove properties of more general "unsolvable" differential equations. They can also prove all properties of differential equations that can be proved with solutions [7] while guaranteeing correctness of the solution as part of the proof.

5.1 Differentials: Invariants, Cuts, Effects, and Ghosts

Figure 3 shows axioms for proving properties of differential equations (DW–DS) as well as axioms for differential substitutions ($[':=]$), and differential axioms for differentials ($+', \cdot', \circ'$). Axioms identifying $(x)' = x'$ for variables $x \in \mathcal{V}$ and $(f)' = 0$ for functions f and number literals of arity 0 are used implicitly. Some axioms use reverse implications $(\phi \leftarrow \psi) \equiv (\psi \rightarrow \phi)$ for emphasis.

$$
\begin{array}{ll}
\text{DW} & [x' = f(x) \,\&\, q(x)]q(x) \\
\text{DC} & \big([x' = f(x) \,\&\, q(x)]p(x) \leftrightarrow [x' = f(x) \,\&\, q(x) \wedge r(x)]p(x)\big) \leftarrow [x' = f(x) \,\&\, q(x)]r(x) \\
\text{DE} & [x' = f(x) \,\&\, q(x)]p(x, x') \leftrightarrow [x' = f(x) \,\&\, q(x)][x' := f(x)]p(x, x') \\
\text{DI} & [x' = f(x) \,\&\, q(x)]p(x) \leftarrow \big(q(x) \rightarrow p(x) \wedge [x' = f(x) \,\&\, q(x)](p(x))'\big) \\
\text{DG} & [x' = f(x) \,\&\, q(x)]p(x) \leftrightarrow \exists y\, [x' = f(x), y' = a(x)y + b(x) \,\&\, q(x)]p(x) \\
\text{DS} & [x' = f \,\&\, q(x)]p(x) \leftrightarrow \forall t{\geq}0\, \big((\forall 0{\leq}s{\leq}t\, q(x + fs)) \rightarrow [x := x + ft]p(x)\big) \\
[':=] & [x' := f]p(x') \leftrightarrow p(f) \\
+' & (f(\bar{x}) + g(\bar{x}))' \;=\; (f(\bar{x}))' + (g(\bar{x}))' \\
\cdot' & (f(\bar{x}) \cdot g(\bar{x}))' \;=\; (f(\bar{x}))' \cdot g(\bar{x}) + f(\bar{x}) \cdot (g(\bar{x}))' \\
\circ' & [y := g(x)][y' := 1]\big((f(g(x)))' = (f(y))' \cdot (g(x))'\big)
\end{array}
$$

Fig. 3. Differential equation axioms and differential axioms

Differential weakening axiom DW internalizes that differential equations can never leave their evolution domain $q(x)$. DW implies[3] $[x' = f(x) \,\&\, q(x)]p(x) \leftrightarrow [x' = f(x) \,\&\, q(x)](q(x) \rightarrow p(x))$ also called DW, whose (right) assumption is best proved by G. The *differential cut* axiom DC is a cut for differential equations. It internalizes that differential equations staying in $r(x)$ stay in $p(x)$ iff $p(x)$ always holds after the differential equation that is restricted to the smaller evolution domain $\&\, q(x) \wedge r(x)$. DC is a differential variant of modal modus ponens K.

Differential effect axiom DE internalizes that the effect on differential symbols along a differential equation is a differential assignment assigning the right-hand side $f(x)$ to the left-hand side x'. Axiom DI internalizes *differential invariants*, i.e. that a differential equation stays in $p(x)$ if it starts in $p(x)$ and if its differential $(p(x))'$ always holds after the differential equation $x' = f(x) \,\&\, q(x)$. The differential equation also vacuously stays in $p(x)$ if it starts outside $q(x)$, since it is stuck then. The (right) assumption of DI is best proved by DE to select the appropriate vector field $x' = f(x)$ for the differential $(p(x))'$ and a subsequent DW,G to make the evolution domain constraint $q(x)$ available as an assumption. For simplicity, this paper focuses on atomic postconditions for which $(\theta \geq \eta)' \equiv (\theta > \eta)' \equiv (\theta)' \geq (\eta)'$ and $(\theta = \eta)' \equiv (\theta \neq \eta)' \equiv (\theta)' = (\eta)'$, etc. Axiom DG internalizes *differential ghosts*, i.e. that additional differential equations can be added if

[3] $[x' = f(x) \,\&\, q(x)](q(x) \rightarrow p(x)) \;\rightarrow\; [x' = f(x) \,\&\, q(x)]p(x)$ derives by K from DW. The converse $[x' = f(x) \,\&\, q(x)]p(x) \rightarrow [x' = f(x) \,\&\, q(x)](q(x) \rightarrow p(x))$ derives by K since G derives $[x' = f(x) \,\&\, q(x)]\big(p(x) \rightarrow (q(x) \rightarrow p(x))\big)$.

their solution exists long enough. Axiom DS solves differential equations with the help of DG,DC. Vectorial generalizations to systems of differential equations are possible for the axioms in Fig. 3.

The following proof proves a property of a differential equation using differential invariants without having to solve that differential equation. One use of US is shown explicitly, other uses of US are similar for DI,DE,[′ :=] instances.

$$\dfrac{\dfrac{\dfrac{*}{x^3\cdot x + x\cdot x^3 \geq 0}\,\mathbb{R}}{[x' := x^3]x'\cdot x + x\cdot x' \geq 0}\,[':=]}{[x' = x^3][x' := x^3]x'\cdot x + x\cdot x' \geq 0}\,\mathrm{G} \qquad \dfrac{\dfrac{\dfrac{\dfrac{*}{(f(\bar{x})\cdot g(\bar{x}))' = (f(\bar{x}))'\cdot g(\bar{x}) + f(\bar{x})\cdot(g(\bar{x}))'}\,{}'}{(x\cdot x)' = (x)'\cdot x + x\cdot(x)'}\,\mathrm{US}}{(x\cdot x)' = x'\cdot x + x\cdot x'}}{(x\cdot x)' \geq 0 \leftrightarrow x'\cdot x + x\cdot x' \geq 0}\,\mathrm{CQ}$$

$$\dfrac{(x\cdot x \geq 1)' \leftrightarrow x'\cdot x + x\cdot x' \geq 0}{[x' = x^3][x' := x^3](x\cdot x \geq 1)'}\,\mathrm{CE}$$

$$\dfrac{[x' = x^3][x' := x^3](x\cdot x \geq 1)'}{[x' = x^3](x\cdot x \geq 1)'}\,\mathrm{DE}$$

$$\dfrac{[x' = x^3](x\cdot x \geq 1)'}{x\cdot x \geq 1 \rightarrow [x' = x^3]x\cdot x \geq 1}\,\mathrm{DI}$$

Previous calculi [5,7] collapse this proof into a single proof step with complicated built-in operator implementations that silently perform the same reasoning in an opaque way. The approach presented here combines separate axioms to achieve the same effect in a modular way, because they have individual responsibilities internalizing separate logical reasoning principles in *differential-form* $\mathsf{d}\mathcal{L}$.

5.2 Differential Substitution Lemmas

The key insight for the soundness of DI is that the analytic time-derivative of the value of a term η along a differential equation $x' = \theta \,\&\, \psi$ agrees with the values of its differential $(\eta)'$ along the vector field of that differential equation.

Lemma 8 (Differential lemma). *If $I, \varphi \models x' = \theta \wedge \psi$ holds for some flow $\varphi : [0, r] \to \mathcal{S}$ of any duration $r > 0$, then for all $0 \leq \zeta \leq r$:*

$$[\![(\eta)']\!]^I\varphi(\zeta) = \frac{\mathsf{d}[\![\eta]\!]^I\varphi(t)}{\mathsf{d}t}(\zeta)$$

The key insight for the soundness of differential effects DE is that differential assignments mimicking the differential equation are vacuous along that differential equation. The differential substitution resulting from a subsequent use of [′:=] is crucial to relay the values of the time-derivatives of the state variables x along a differential equation by way of their corresponding differential symbol x'. In combination, this makes it possible to soundly substitute the right-hand side of a differential equation for its left-hand side in a proof.

Lemma 9 (Differential assignment). *If $I, \varphi \models x' = \theta \wedge \psi$ holds for some flow $\varphi : [0, r] \to \mathcal{S}$ of any duration $r \geq 0$, then*

$$I, \varphi \models \phi \leftrightarrow [x' := \theta]\phi$$

The final insights for differential invariant reasoning for differential equations are syntactic ways of computing differentials, which can be internalized as axioms $(+', \cdot', \circ')$, since differentials are syntactically represented in differential-form $\mathsf{d}\mathcal{L}$.

Lemma 10 (Derivations). *The following equations of differentials are valid:*

$$(f)' = 0 \qquad \text{for arity 0 functions/numbers } f \tag{1}$$
$$(x)' = x' \qquad \text{for variables } x \in \mathcal{V} \tag{2}$$
$$(\theta + \eta)' = (\theta)' + (\eta)' \tag{3}$$
$$(\theta \cdot \eta)' = (\theta)' \cdot \eta + \theta \cdot (\eta)' \tag{4}$$
$$[y := \theta]\,[y' := 1]\big((f(\theta))' = (f(y))' \cdot (\theta)'\big) \qquad \text{for } y, y' \notin \theta \tag{5}$$

5.3 Soundness

Theorem 2 (Soundness). *The* $\mathsf{d}\mathcal{L}$ *axioms and proof rules in Figs. 2 and 3 are sound, i.e. the axioms are valid formulas and the conclusion of a rule is valid if its premises are. All US instances of the proof rules (with* $FV(\sigma) = \emptyset$*) are sound.*

6 Conclusions

With differential forms for local reasoning about differential equations, uniform substitutions lead to a simple and modular proof calculus for differential dynamic logic that is entirely based on axioms and axiomatic rules, instead of soundness-critical schema variables with side-conditions in axiom schemata. The US calculus is straightforward to implement and enables flexible reasoning with axioms by contextual equivalence. Efficiency can be regained by tactics that combine multiple axioms and rebalance the proof to obtain short proof search branches. Contextual equivalence rewriting for implications is possible when adding monotone quantifiers C whose substitution instances limit $\_$ to positive polarity.

Acknowledgment. I thank the anonymous reviewers for their helpful feedback.

References

1. Church, A.: A formulation of the simple theory of types. J. Symb. Log. **5**(2), 56–68 (1940)
2. Church, A.: Introduction to Mathematical Logic, vol. I. Princeton University Press, Princeton (1956)
3. Henkin, L.: Banishing the rule of substitution for functional variables. J. Symb. Log. **18**(3), 201–208 (1953)
4. Platzer, A.: Differential dynamic logic for hybrid systems. J. Autom. Reas. **41**(2), 143–189 (2008)
5. Platzer, A.: Differential-algebraic dynamic logic for differential-algebraic programs. J. Log. Comput. **20**(1), 309–352 (2010)
6. Platzer, A.: The complete proof theory of hybrid systems. In: LICS, pp. 541–550. IEEE (2012)
7. Platzer, A.: The structure of differential invariants and differential cut elimination. Log. Meth. Comput. Sci. **8**(4), 1–38 (2012)
8. Platzer, A.: Differential game logic. CoRR abs/1408.1980 (2014)
9. Platzer, A.: A uniform substitution calculus for differential dynamic logic. CoRR abs/1503.01981 (2015)