A Real-Analytic Approach to Differential-Algebraic Dynamic Logic

Jonathan Hellwig(✉) and André Platzer

Karlsruhe Institute of Technology, Karlsruhe, Germany
{jonathan.hellwig, platzer}@kit.edu

Abstract. This paper introduces a novel axiomatic proof calculus for differential-algebraic dynamic logic (dAL). The calculus enables deductive verification and sound transformation of differential-algebraic programs (DAPs), which generalize differential-algebraic equations, while remaining compatible with differential dynamic logic (dL) for hybrid programs. One central contribution is the ghost switching axiom which establishes precise conditions to decompose multi-modal DAPs into equivalent hybrid systems with ordinary differential equations. The applicability of the calculus is demonstrated through a formal equivalence proof, showing the reduction of the Euclidean pendulum from a differential-algebraic formulation to an equivalent system of ordinary differential equations.

Keywords: Differential dynamic logic · Differential-algebraic equations · Proof calculus · Theorem proving

1 Introduction

Hybrid systems with explicit (vectorial) ordinary differential equations (ODEs) $x' = f(x)$ are important models for cyber-physical systems [13,3,8] and have been axiomatized in differential dynamic logic (dL) [21,25,23,26], which is the basis of the theorem prover KeYmaera X [10] that was used to prove correctness properties of aircraft, train and robotics systems [7,14,17]. This paper extends (modern differential-form) differential dynamic logic [25] with *differential-algebraic programs (DAPs)*. DAPs allow the relation of x and its time-derivative x' to be defined by a formula of first-order real arithmetic. In particular, DAPs include ordinary equations given by implicit equations such as $f(x', x) = 0$. Beyond that, DAPs also support composition using logical operators such as conjunctions $\wedge$ and disjunctions $\vee$, enabling modular and flexible specification of system behavior. This gives rise to easy combinations and extensions of systems.

Consider a model of an electric circuit consisting of a resistor r, a capacitor c and a source s. Each component of the circuit can be given by a local equation:

$$v_r = R\, i_r, \quad i_c = C\, v_c', \quad v_s = 0,$$

where $v_\bullet$ and $i_\bullet$ are the voltage and the current of each component, and R and C are constants. Assuming these three elements are connected in series, Kirchhoff's

C. W. Barrett and U. Waldmann (Eds.): CADE-30, LNCS 15943, pp. 696–714, 2025.
DOI: 10.1007/978-3-031-99984-0_36

laws contribute two connection constraints $i_r = i_c$ and $v_r + v_c = v_s$. Putting everything together, the electric circuit is described by the single real-arithmetic formula

$$v_r = Ri_r \wedge i_c = Cv_c' \wedge v_s = 0 \wedge i_r = i_c \wedge v_r + v_c = v_s.$$

The complete model emerges by simply joining each component's constitutive equation with the corresponding connection constraints. This illustrates the key advantage of differential-algebraic programs: complex systems are built compositionally from independent components. For that reason, differential-algebraic equations are the mathematical foundation for the popular modeling tool *Modelica* [9]. However, the DAPs presented in this paper extend beyond differential algebraic equations by allowing inequality and quantified constraints such as $x' \leq f(x)$ and $\exists y(g(x,y) = 0 \wedge x' = f(x))$. This added expressiveness supports modelling of uncertainty and of dynamics with non-polynomial right-hand sides.

Yet, the expressive power and flexibility of DAPs comes with added complexities in numerical and symbolic analysis. DAPs may have hidden constraint that are only revealed when differentiating one of their algebraic constraints. Consider the *Euclidean pendulum*[1] [18, p. 13] — a classical example of a system of differential-algebraic equations. Its dynamics are given by the formula

$$x' = v \wedge v' = \lambda x \wedge y' = w \wedge w' = \lambda y - g \wedge x^2 + y^2 = 1,$$

where x, y are the pendulum's position coordinates, and v, w are the velocities. The scalar λ is a Lagrange multiplier implicitly constrained by the holonomic condition $x^2 + y^2 = 1$, and g denotes the gravitational constant. Suppose we wish to analyze the pendulum's behavior with respect to some safety property. Although x and y are confined to the unit circle, their time derivatives $x' = v$ and $y' = w$ are governed by the equations $v' = \lambda x$ and $w' = \lambda y - g$, so the evolution of x and y depends on the unknown multiplier λ. This multiplier is not a free parameter of the system: solutions exist only if λ satisfies the hidden constraint

$$\lambda = gy - (v^2 + w^2).$$

That relation is not explicit in the original model. It is only uncovered after repeatedly differentiating the algebraic constraint $x^2 + y^2 = 1$ and performing algebraic manipulations. Without this hidden constraint, we cannot directly analyze the behavior of x and y, because λ would remain undetermined. While such derivations are feasible for small examples, carrying them out by hand is tedious and error-prone, and doing so outside a formal logical framework forfeits any end-to-end guarantee of correctness. Because hidden constraints are a fundamental feature of DAPs, we need systematic and sound principles to uncover them, and a logic expressive enough to state and prove the resulting properties.

To meet this need, this paper introduces the *differential-form differential-algebraic dynamic logic* (dAL). This new logic gives native support to hybrid

[1] Its equations follow from the Lagrangian $\mathcal{L}(x, y, \lambda) = gx - \frac{1}{2}(v^2+w^2) + \frac{\lambda}{2}\big(x^2+y^2-1\big)$.

programs whose continuous behavior is specified by DAPs and provides an axiomatic proof calculus for reasoning about their dynamic properties. This calculus fulfills two roles. First, it guarantees correctness-preserving transformations of DAPs. Second, it enables rigorous proofs of system properties such as safety and reachability.

A first precursor, *differential-algebraic dynamic logic* (DAL) [22,23], already extended the original, non-differential-form variant of differential dynamic logic. The differential-form logic dAL presented in this paper builds on DAL and changes it in several essential ways. The main contributions of this paper are:

1. *Logical foundations for model transformations*: This calculus establishes a rigorous basis for differential and algebraic transformations of DAPs — an indispensable capability for practical work with DAPs that was not available in DAL. It also lays the groundwork for future integration with larger-scale modelling frameworks.
2. *Axioms for non-normalized differential constraints*: The prior DAL proof calculus required a syntactic normalization of DAPs. The new axioms remove that prerequisite: they apply directly and modularly to *non-normalized* formulas, streamlining proofs and simplifying implementation.
3. *Ghost switching axiom*: We introduce a new *ghost switching axiom* that gives exact criteria for when a multi-mode DAP can be split into a hybrid systems containing simpler, single-mode components. This axiom formally justifies a transformation that has previously been used without proof in the literature[28,15].
4. *Differential-form reasoning*: Our proof calculus employs differential forms, linking it to existing differential-invariant completeness results [27]. The same completeness guarantee now applies to every DAP that can be reduced to a polynomial ordinary differential equation.
5. *Local Hilbert-type proof calculus*: Compared to the prior DAL sequent calculus, our formulation represents a significant step toward a uniform substitution calculus for dAL. Uniform substitution externalizes side-condition checking, simplifying proof manipulation and theorem prover implementations [25].

2 Differential-algebraic Dynamic Logic

This section briefly introduces the syntax and semantics of (differential-form) differential-algebraic dynamic logic, with a focus on its continuous fragment. For a full treatment of the discrete fragment, the interested reader is referred to the foundational work on dL [25,26].

2.1 Syntax

The set $\mathcal{V}$ contains all variables and for each variable $x \in \mathcal{V}$, there is a differential variable $x' \in \mathcal{V}$. By $\mathcal{V}' \subseteq \mathcal{V}$ we denote the set of all differential variables. A

differential variable x' represents the instantaneous rate of change of x, much like a derivative in calculus. However, rather than being a function of time explicitly it is treated as an independent variable within the system. This abstraction allows differentials to be represented algebraically, making it possible to manipulate them syntactically.

Definition 1 (Syntax of terms and formulas [25,26]). *The language of* dAL terms *is generated by the following grammar, where x is a variable, c is a rational constant, and f is a function symbol:*

$$e, g ::= x \mid c \mid f(e_1, \ldots, e_k) \mid e \cdot g \mid e + g \mid (e)'.$$

The language of dAL formulas *is generated by the following grammar, where x is a variable, P a predicate symbol, and α a differential-algebraic program (Def. 2):*

$$F, G ::= e \leq g \mid P(e_1, \ldots, e_k) \mid F \wedge G \mid \neg F \mid \forall x F \mid [\alpha]F.$$

Remark 1. The usual logical and modal operators such as $<, \geq, >, \neq, \vee, \rightarrow, \exists$ and $\langle\cdot\rangle$ are *definable* in terms of more primitive constructs. For instance, disjunction can be defined as $A \vee B \equiv \neg(\neg A \wedge \neg B)$, and the diamond modality as $\langle\alpha\rangle P \equiv \neg[\alpha]\neg P$. Throughout this work, we adopt vectorial notation: $\bar{x} = (x_1, \ldots, x_n)$ denotes a tuple of variables. For vector-valued function symbols $f(\bar{x}) = (f_1(\bar{x}), \ldots, f_n(\bar{x}))$, the expression $\bar{x} = f(\bar{x})$ abbreviates the conjunction $\bigwedge_{i=1}^{n} x_i = f_i(x_1, \ldots, x_n)$. The square of the Euclidean norm is defined as $\|\bar{x}\|^2 \equiv \sum_{i=1}^{n} x_i^2$.

Remark 2. For a variable x and formula F we write $x \in F$ to mean that x is a free variable of that formula [25, Def. 7].

Definition 2 (Syntax of programs [25,26]). *The language of* dAL *programs is generated by the following grammar, where x is a variable, $\bar{x}$ is a tuple of variables, e is a term, and F is a formula of first-order real arithmetic:*

$$\alpha, \beta ::= x := e \mid ?F \mid \{\bar{x} : F\} \mid \alpha; \beta \mid \alpha \cup \beta \mid \alpha^*.$$

Remark 3. In this syntax, an assignment is denoted by $x := e$, a test by $?F$, a nondeterministic choice by $\alpha \cup \beta$, and a nondeterministic loop by α^*. For any $n \in \mathbb{N}$, the notation α^n denotes the n-fold sequential composition of α.

Remark 4. Unlike ordinary differential equations, where the bound variables are syntactically evident form $x' = f(x)$, DAPs may involve arbitrary first-order real arithmetic formulas. For this reason, the notation used here explicitly indicates which variables evolve in the DAPs: in $\{\bar{x} : F\}$, the bound variables are given by the tuple $\bar{x}$.

2.2 Semantics

A dAL state is a function $\omega : \mathcal{V} \to \mathbb{R}$ that assigns a real value to each variable in the set of all variables $\mathcal{V}$. The set of states is denoted by $\mathcal{S}$. Given a subset

$X \subseteq \mathcal{V}$, we write $X^{\complement} = \mathcal{V} \setminus X$ for its complement. For a variable x and a real value $r \in \mathbb{R}$, we write ω_x^r to denote the state obtained from ω by replacing its value at x with r. For a state $\omega \in \mathcal{S}$ and a tuple $\bar{x}$, we write $\omega(\bar{x})$ as a shorthand for $(\omega(x_1), \ldots, \omega(x_n))$. Finally, the partial derivative of a differentiable function h with respect to x is written as $\frac{\partial h}{\partial x}$.

Definition 3 (Semantics of terms [25]). *The* semantics $\omega[\![e]\!] \in \mathbb{R}$ *of a* dAL *term e in state $\omega \in \mathcal{S}$ is inductively as follows:*

1. $\omega[\![x]\!] = \omega(x)$ *for a variable* $x \in \mathcal{V}$,
2. $\omega[\![f(e_1, \ldots, e_k)]\!] = f(\omega[\![e_1]\!], \ldots, \omega[\![e_k]\!])$ *for a function* $f : \mathbb{R}^k \to \mathbb{R}$,
3. $\omega[\![e + g]\!] = \omega[\![e]\!] + \omega[\![g]\!]$,
4. $\omega[\![e \cdot g]\!] = \omega[\![e]\!] \cdot \omega[\![g]\!]$,
5. $\omega[\![(e)']\!] = \sum_{x \in \mathcal{V}} \omega(x') \frac{\partial \omega[\![e]\!]}{\partial x}$.

The differential $(e)'$ characterizes how the value of e changes locally in response to variations in its free variables x. Specifically, it expresses this dependence as a function of the corresponding differential symbols x', which represent the rate of change of x. We note that each term e can only have finitely many free variables. Therefore, the sum in the definition of $(e)'$ is always finite.

Definition 4 (Semantics of formulas [25]). *The* semantics of a dAL formula *F is the subset $[\![F]\!] \subseteq \mathcal{S}$ of states in which F is true and defined as:*

1. $[\![e \leq g]\!] = \{\omega \in \mathcal{S} \mid \omega[\![e]\!] \leq \omega[\![g]\!]\}$,
2. $[\![P(e_1, \ldots, e_n)]\!] = \{\omega \in \mathcal{S} \mid (\omega[\![e_1]\!], \ldots, \omega[\![e_n]\!]) \in [\![P]\!]\}$,
3. $[\![\neg F]\!] = \mathcal{S} \setminus [\![F]\!]$,
4. $[\![F \wedge G]\!] = [\![F]\!] \cap [\![G]\!]$,
5. $[\![\forall x F]\!] = \{\omega \in \mathcal{S} \mid \forall r \in \mathbb{R} : \omega_x^r \in [\![F]\!]\}$,
6. $[\![[\alpha]F]\!] = \{\omega \in \mathcal{S} \mid \forall (\omega, \nu) \in [\![\alpha]\!] : \nu \in [\![F]\!]\}$.

The *box modality* $[\alpha]F$ is used to express *safety properties*, statements that hold after all possible executions of the system. If $[\alpha]F$ is true in a given state, it guarantees that no matter how the system evolves according to the program α, the condition F will always hold. The *diamond modality* $\langle\alpha\rangle F$, on the other hand, is used to express reachability properties, statements asserting that some execution of the system can lead to a desired state. If $\langle\alpha\rangle F$ holds, then there exists at least one execution of the program α that reaches a state where F is true.

Definition 5 (Real-analytic function). *On an open set $U \subset \mathbb{R}$ the function $f : U \to \mathbb{R}$ is said to be* real-analytic*, if for every $t_0 \in U$ there exist $\varepsilon > 0$ and coefficients $a_k \in \mathbb{R}$ such that for all $t \in (t_0 - \varepsilon, t_0 + \varepsilon)$*

$$f(t) = \sum_{k=0}^{\infty} a_k (t - t_0)^k.$$

Definition 6 (Flow). *A mapping $\varphi : (a, b) \to \mathcal{S}$ is called a* flow *in $\bar{x}$, if $\varphi(\cdot)(\bar{x})$ is real-analytic on (a, b) and satisfies the following two properties:*

1. $\varphi(t)(\bar{x}') = \frac{d\varphi(t)(\bar{x})}{dt}(t)$ *for all* $t \in (a,b)$,
2. $\varphi(t) = \varphi(0)$ *on* $(\bar{x} \cup \bar{x}')^{\complement}$ *for all* $t \in (a,b)$.

We write $\mathcal{A}_{\bar{x}}(a,b)$ *to denote the set of flows in* $\bar{x}$ *defined on the interval* (a,b). *For a formula* F *and an interval* I, *we also write* $\varphi(I) \subseteq [\![F]\!]$ *to denote* $\varphi(t) \in [\![F]\!]$ *for all* $t \in I$.

Remark 5. There always exists a flow $\varphi \in \mathcal{A}_{\bar{x}}(-\varepsilon,\varepsilon)$ that satisfies a given state $\omega \in \mathcal{S}$ at time 0. Define $\varphi : (-\varepsilon,\varepsilon) \to \mathcal{S}$, $\varphi(t)(\bar{x}) = \omega(\bar{x}')t + \omega(\bar{x})$. We can verify that $\varphi(0)(\bar{x}) = \omega(\bar{x})$ and $\varphi(0)(\bar{x}') = \frac{d\varphi(t)(\bar{x})}{dt}(0) = \omega(\bar{x}')$.

Definition 7 (Semantics of programs). *The* semantics of a dAL program α *is the transition relation* $[\![\alpha]\!] \subseteq \mathcal{S} \times \mathcal{S}$ *and is defined as follows:*

$$
\begin{aligned}
&[\![\alpha \cup \beta]\!] = [\![\alpha]\!] \cup [\![\beta]\!],\\
&[\![\alpha;\beta]\!] = [\![\alpha]\!] \circ [\![\beta]\!],\\
&[\![x := e]\!] = \{(\omega, \omega_x^r) \mid r = \omega[\![e]\!]\},\\
&[\![?F]\!] = \{(\omega,\omega) \mid \omega \in [\![F]\!]\},\\
&[\![\alpha^*]\!] = \textstyle\bigcup_{n\in\mathbb{N}} [\![\alpha^n]\!],\\
&[\![\{\bar{x} : F\}]\!] = \{(\omega,\nu) \mid \textit{There exist } T \geq 0, \varepsilon > 0 \textit{ and a flow } \varphi \in \mathcal{A}_{\bar{x}}(-\varepsilon, T+\varepsilon),\\
&\quad \textit{with } \varphi([0,T]) \subseteq [\![F]\!] \textit{ and } \omega = \varphi(0) \textit{ and } \varphi(T) = \nu\}.
\end{aligned}
$$

Remark 6. The semantics of DAPs require flows to be real-analytic functions on an open neighborhood of $[0,T]$. In contrast, the differential constraint F only needs to hold on the smaller closed set $[0,T]$. This property is crucial for the soundness of the proof calculus in Section 3.

Example 1. To better understand the open-neighborhood assumption on flows, consider the following two DAPs:

$$\alpha \equiv \{x : x' = -1 \wedge x \geq 0\} \text{ and } \beta \equiv \{x, y : x' = -1 \wedge y^2 = x \wedge x \geq 0\}.$$

Fix the initial state $\omega(x) = 1$, $\omega(x') = -1$, $\omega(y) = 1$, and $\omega(y') = -1/2$. For some $\varepsilon > 0$, a flow $\varphi_\alpha \in \mathcal{A}_x(-\varepsilon, 1+\varepsilon)$ of system α is given by $\varphi_\alpha(t)(x) = 1 - t$. Since $\varphi_\alpha(\cdot)(x)$ is a polynomial, it extends analytically beyond the interval [0,1], even though the differential constraint only holds on [0,1]. For system β, a flow $\varphi_\beta \in \mathcal{A}_{x,y}(-\varepsilon, 1+\varepsilon)$ is given by $\varphi_\beta(t)(x) = 1 - t$ and $\varphi_\beta(t)(y) = \sqrt{1-t}$ for any value $0 \leq T < 1$. However, $\varphi_\beta(\cdot)(y)$ does not extend to $[0,1]$, because its first derivative exhibits finite-time blow-up (see Fig. 1):

$$\lim_{t\to 1} \frac{d\varphi_\beta(t)(x)}{dt} = \lim_{t\to 1} -\frac{1}{2\sqrt{1-t}} = -\infty.$$

While for both system α and β there exists a flow that satisfies their respective constraint each closed subinterval $I \subset [0,1]$, only system α has a flow that also satisfies it on $[0,1]$. More generally, Def. 6 ensures that φ remains well-behaved in a small neighborhood of $[0,T]$ and ensures all derivatives are bounded on $[0,T]$.

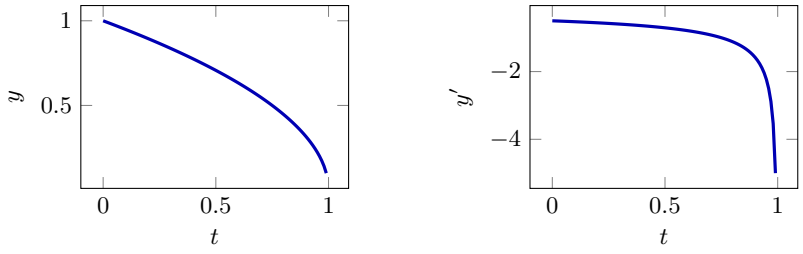

Fig. 1. The behavior $\varphi_\beta(\cdot)$ at y and y' as $t \to 1$.

Differential-algebraic programs are strictly more expressive than ODE systems in dL. Below, we show three key examples:

1. **Modeling Uncertainty**: Differential-algebraic programs can represent systems with continuous uncertainty by incorporating disturbances. For instance, the system

$$\{x, w : x' = f(x, w) \land w^- \leq w \leq w^+\}$$

   describes a dynamical system where the evolution of x is governed by $f(x, w)$ with a disturbance w, constrained within the range $[w^-, w^+]$.
2. **Implicit System Characterization**: Differential-algebraic programs can define systems implicitly using existential quantification. Consider:

$$\{x : \exists y(x' = y \land y^2 = x \land x > 0)\}$$

   Here, the derivative of x is characterized by the square root of x, as $x' = y$ must hold for some y satisfying $y^2 = x$. This implicitly defines the nonpolynomial differential equation with nonunique solutions:

$$x' = \pm\sqrt{x}, \quad \text{for } x > 0.$$

3. **Multimodal Systems**: These programs can naturally represent multimodal continuous programs. For example, the system

$$\{x : (x' = f(x) \land x \leq 0) \lor (x' = f(x) \land x \geq 0)\}$$

   models an evolution in the two domains $x \leq 0$ and $x \geq 0$. In Section 3, we see that such programs can be decomposed into a loop with a nondeterministic choice between the two modes, provided that the handoff points are carefully managed. Note that there is no arbitrary switching between these two modes, because flows are real-analytic.

Remark 7. All ODE solutions in the dL sense are real-analytic (By the Cauchy-Kovalevskaya Theorem [16, Theorem 1.7.1], all ODE solution with real-analytic right-hand side are real-analytic). Thus, the dAL DAP semantics generalize the

dL ODE semantics, allowing for a broader class of systems. In particular, the expressive term language of DAPs in dAL permits non-Lipschitz differential constraints (See 2 above). Non-Lipschitzian systems exhibit an implicit nondeterminism, such as spontaneous movement from rest [2]. To avoid unintended implicit behavior, we restrict flows to be real-analytic. When nondeterminism is needed, it can be explicitly introduced using discrete program constructs. For instance, a flow of the system $\{x : x' = f(x) \vee x' = g(x)\}$ never switches between two distinct right-hand sides. Because the flows of both modes are real-analtyic, the Identity Theorem for real-analytic functions [16, Cor. 1.2.6 p. 14] guarantees that any two flows agreeing on an open time interval must agree along their whole trajectories. If switching is desired, it must be made explicit with discrete assignments:

$$\big(x' := f(x); \{x : x' = f(x)\} \cup x' := g(x); \{x : x' = g(x)\}\big)^*.$$

Choosing flows to be real-analytic strikes the balance between flexible modeling and control over system behavior while still allowing for a powerful proof calculus.

3 Axiomatic Proof Calculus

In this section, we introduce a proof calculus for DAPs from Section 2. The complete proofs of the statements in this section are presented in [12].

3.1 Progress, Exit and Entry formulas

For reasoning about DAPs and the soundness of our proof calculus, it is crucial to understand how DAPs evolve locally and under what conditions they transition between modes. For discrete-time programs, the notion of *next step* is unambiguous: given a current state, there is a distinct next state at the subsequent time step. In the loop $(x := -x)^*$, a state with $x > 0$ is followed by a state with $x < 0$; each state has a distinct successor state. In contrast, continuous-time programs differ fundamentally, because the state evolves on real intervals of time. There is no smallest positive time instance, so there is no notion of next state — one can speak only of states that are arbitrarily close in the future.

Example 2. Let us consider the differential-algebraic program

$$\{x : x' = 1 \wedge -1 < x \wedge x \leq 1\},$$

started in the initial state $\omega \in \mathcal{S}$. The unique flow of this system is given by

$$\varphi(t)(x) = \omega(x) + t$$

as long as $-1 < \varphi(t)(x) \leq 1$ is satisfied. We consider three cases for the initial condition:

(i) Strictly inside the domain: If $-1 < \omega(x) < 1$, there always exists a positive time increment $\varepsilon > 0$ such that $\omega(x)+\varepsilon < 1$. Hence, the system can advance by every such ε, and the flow may continue until $\omega(x)$ reaches 1.

(ii) On the boundary: If $\omega(x) = 1$, the inequality $\varphi(t)(x) \leq 1$ is already tight. Any positive advance would violate it, so the only admissible evolution is the *blocked* flow $\varphi(t)(x) = \omega(x) + t$ for $t = 0$.

(iii) Outside the domain: If $\omega(x) > 1$, the initial state breaks the constraint $\varphi(t)(x) \leq 1$ — no execution is possible.

This example illustrates that there exist states in which the system can evolve and others where further progress is impossible. This distinction motivates the following definition:

Definition 8 (Local progress formula [27, p. 24]). *Let $\bar{y}$ and $\bar{y}'$ be two tuples of variables that do not occur in the formula F. The* local progress formula *of the system $\{\bar{x} : F\}$ is defined as*

$$\langle\{\bar{x} : F\}\rangle\bigcirc \;\equiv\; \exists\bar{y}\exists\bar{y}'\,(\bar{x} = \bar{y} \wedge \bar{x}' = \bar{y}' \wedge \langle\{\bar{x} : F \vee (\bar{x} = \bar{y} \wedge \bar{x}' = \bar{y}')\}\rangle(\bar{x} \neq \bar{y}))\,.$$

The formula expresses that, from the current state, the system can evolve to a different state—that is, make local progress. The right disjunct in the diamond modality includes all the points on the boundary of F that may not be strictly contained in F. In this way, the formula captures all states from which there exists a positive time step such that the state advances into F.

Example 3 (continued). Continuing Example 2, the progress formula $\langle\{x : x' = 1 \wedge x \leq 1\}\rangle\bigcirc$ precisely captures the concept we previously discussed: the characterization of states in which the system can locally evolve. In this particular instance, the progress formula is equivalent to $x' = 1 \wedge -1 < x \wedge x < 1$.

Equipped with the notion of progress formulas, we now define *local exit formulas*, which identify the state from which a flow may no longer continue within a mode, and *local entry formulas*, which mark the states from which a mode may start. Together, these formulas make it possible to describe the exact conditions under which a system leaves one mode and enters the next. The following definition is inspired by the approach in prior work [11].

Definition 9 (Local exit and entry formulas). *Let $F^- \equiv [\bar{x}' := -\bar{x}']F$ denote the reverse-flow formula of a formula F. The* local exit formula $\langle\{\bar{x} : F\}\rangle{\uparrow}\bigcirc$ *and the* local entry formula $\langle\{\bar{x} : F\}\rangle{\downarrow}\bigcirc$ *of the system $\{\bar{x} : F\}$ are defined as*

$$\langle\{\bar{x} : F\}\rangle{\uparrow}\bigcirc \equiv (\langle\{\bar{x} : \neg F\}\rangle\bigcirc \wedge F) \vee (\langle\{\bar{x} : F^-\}\rangle\bigcirc^- \wedge \neg F),$$
$$\langle\{\bar{x} : F\}\rangle{\downarrow}\bigcirc \equiv (\langle\{\bar{x} : F\}\rangle\bigcirc \wedge \neg F) \vee (\langle\{\bar{x} : \neg F^-\}\rangle\bigcirc^- \wedge F).$$

Remark 8. A state ω satisfies the *exit formula*, if and only if, one of two situations occurs:

(i) *Forward flow*: Starting from ω, there exists a flow of positive duration that satisfies the constraint F initially, but enter its negation $\neg F$ immediately.

(ii) *Reverse flow*: Alternatively, beginning in ω, there exists a time-reverse flow of positive duration that starts in the negation $\neg F$ and enters the constraint F .

Including the reverse-flow disjunct may seem surprising at first, but it is essential when the system's constraint is *open*. Open constraints do not contain their boundary, so no forward flow can reach the boundary while staying in the constraint. By considering a reverse flow starting form the negation, we can capture the exit points on the boundary.

Example 4 (continued). We now apply the notions of *entry formula* and *exit*

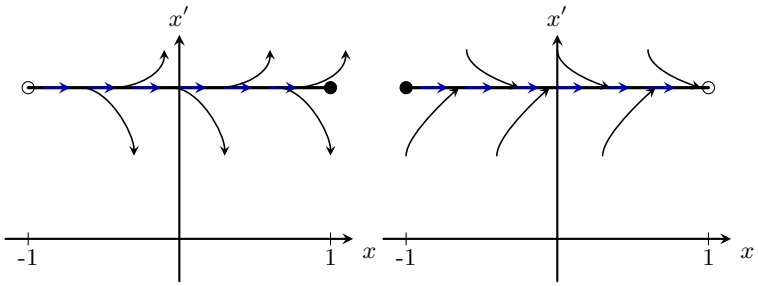

Fig. 2. Entry and exit formula for $\{x : x' = 1 \wedge -1 < x \wedge x \leq 1\}$. Left: The exit formula includes all points from which a trajectory can locally leave the constraint region (black arrows). The excluded left endpoint is shown as an open circle, the included right endpoint as a filled circle. Right: The entry formula includes all points where trajectories can locally enter the region.

formula to the differential-algebraic program

$$\{x : x' = 1 \wedge -1 < x \wedge x \leq 1\}$$

from Example 2. *Exit formula:* A state satisfies the exit formula, if there exists a flow of positive duration that makes progress into the constraint's negation or a reverse flow can progress from the negation into the constraint. For the present system this reduces to

$$\langle \{x : x' = 1 \wedge -1 < x \wedge x \leq 1\} \rangle \uparrow \bigcirc \;\leftrightarrow\; x' = 1 \wedge -1 < x \wedge x \leq 1.$$

The left-hand inequality remains strict, because each flow starting in the domain immediately evolves away from the left boundary, i.e., it is no exit point (See Fig. 2 left). *Entry formula:* In contrast to the exit formula, a flow may enter into the constraint through its left open boundary, even though it is not part of the constraint (See Fig. 2 right). Consequently, both disjuncts from Def. 9 contribute to the entry formula. It reduces to

$$\langle \{x : x' = 1 \wedge -1 < x \wedge x \leq 1\} \rangle \downarrow \bigcirc \;\leftrightarrow\; x' = 1 \wedge -1 \leq x \wedge x < 1.$$

A central lemma, ensuring the correctness of the definition of exit and entry formulas, is the following generalization of the intermediate value theorem:

Lemma 1. *Let $T \geq 0, \varepsilon > 0$, $\varphi \in \mathcal{A}_{\bar{x}}(-\varepsilon, T+\varepsilon)$ be a flow and F be a semi-algebraic formula. Then, if we have $\varphi(0) \in [\![\neg F]\!]$ and $\varphi(T) \in [\![F]\!]$, there exists a $\xi \in [0,T]$ such that $\varphi(\xi) \in [\![\langle\{\bar{x} : F\}\rangle\downarrow\bigcirc]\!]$ and $\varphi(\xi) \in [\![\langle\{\bar{x} : \neg F\}\rangle\uparrow\bigcirc]\!]$.*

This lemma formalizes our intuition of entry and exit formulas: when a system's trajectory transitions from a state not satisfying F to one satisfying F, then there must be an intermediate point simultaneously satisfying the entry formula $\langle\{\bar{x} : F\}\rangle\downarrow\bigcirc$ and the exit formula $\langle\{\bar{x} : \neg F\}\rangle\uparrow\bigcirc$. Now, we finally have the necessary tools to formally define when two modes of a DAP are consistent:

Definition 10 (Mode consistency). *Let F and G be two formulas of first-order real arithmetic. The* mode consistency formula $F \rightsquigarrow_{\bar{x}} G$ *of F and G is defined as*

$$F \rightsquigarrow_{\bar{x}} G \equiv (\langle\{\bar{x} : F\}\rangle\uparrow\bigcirc \wedge \langle\{\bar{x} : G\}\rangle\downarrow\bigcirc) \rightarrow (F \leftrightarrow G).$$

Intuitively, the formula $F \rightsquigarrow_{\bar{x}} G$ states that if a state satisfies both the exit formula of F and the entry formula of G, then both constraints must agree at that point. If this condition is satisfied along every trajectory of a system, the disjunctive DAP $\{\bar{x} : F \vee G\}$ can be safely decomposed into a two distinct DAPs without losing any of its original behavior. The role of this definition for ensuring correct decompositions is illustrated in the following example:

Example 5. Consider the unrestricted system $\alpha \equiv \{x : x' = 1\}$. *Inconsistent decomposition:* One might try to decompose the system into the loop

$$\beta \equiv (\{x : x' = 1 \wedge x < 0\} \cup \{x : x' = 1 \wedge x \geq 0\})^*.$$

The two modes meet at the single boundary state $x' = 1 \wedge x = 0$. That state is precisely the conjunction of the exit formula of the first mode and the entry formula of the second mode. Because the first mode excludes $x = 0$, it creates an implicit infinitesimal barrier, preventing a continuous transition between the two modes. In the original system α, the value of x evolves continuously from negative to positive values without restriction. Hence, α has distinct behavior from β and the consistency condition fails. *Consistent decomposition:* Including the boundary in both modes resolves the issue:

$$\gamma \equiv (\{x : x' = 1 \wedge x \leq 0\} \cup \{x : x' = 1 \wedge x \geq 0\})^*.$$

With the notion of mode consistency formally defined, we now present an axiomatic proof calculus for dAL, which allows rigorous reasoning about mode transitions and model transformations:

Theorem 1 (Soundness). *The following formulas are sound axioms of* dAL*, i.e., they are valid in all states:*

DW $[\{\bar{x}:F\}]F$

C $[\{\bar{x}:F\wedge G\}]P \leftrightarrow [\{\bar{x}:G\wedge F\}]P$

DE $[\{\bar{x}:e=0\}](e)'=0 \quad (\bar{x}'\notin e)$

DX $[\{\bar{x}:F\}]P \rightarrow [?F]P$

DI $(F\rightarrow[\{\bar{x}:F\}](P)')\rightarrow([?F]P\rightarrow[\{\bar{x}:F\}]P) \quad (\bar{x}'\notin P)$

DR $[\{\bar{x}:F\}]P\rightarrow\forall\bar{y}\forall\bar{y}'[\{\bar{x},\bar{y}:F\wedge G\}]P \quad (\bar{y},\bar{y}'\notin P,F)$

AG $[\{\bar{x}:F\}]G(h(\bar{x},\bar{x}'))\rightarrow$
$(\forall\bar{y}\forall\bar{y}'[\{\bar{x},\bar{y}:F\wedge G(\bar{y})\}]P\rightarrow[\{\bar{x}:F\}]P) \quad (\bar{y},\bar{y}'\notin P,F,G(\cdot),h(\cdot,\cdot))$

BDG $[\{\bar{x},\bar{y}:F\wedge\bar{y}'=g(\bar{x},\bar{x}',\bar{y})\}]\ \|\bar{y}\|^2\leq h(\bar{x},\bar{x}')\rightarrow$
$([\{\bar{x}:F\}]P\leftrightarrow[\{\bar{x},\bar{y}:F\wedge\bar{y}'=g(\bar{x},\bar{x}',\bar{y})\}]P) \quad (\bar{y},\bar{y}'\notin P,F,h(\cdot,\cdot))$

GS $[\{\bar{x}:F\vee G\}](F\rightsquigarrow_{\bar{x}}G\wedge G\rightsquigarrow_{\bar{x}}F)\rightarrow$
$((F\vee G\rightarrow[(\{\bar{x}:F\}\cup\{\bar{x}:G\})^*]P)\rightarrow[\{\bar{x}:F\vee G\}]P)$

The axioms for DAPs in Theorem 1 capture essential properties of their evolution. The *Differential Weakening (DW)* axiom ensures that the formulas representing differential-algebraic program remain valid throughout the system's execution. The *Commutativity (C)* axiom allows syntactic exchange of conjunctive formulas. The *Differential Effect (DE)* axiom justifies that the differential $(e)'=0$ of any algebraic equation $e=0$ also holds when following $e=0$ since equations are closed under differentials. Thereby, the DE axiom also enforces consistency between a variable x and its derivative x', ensuring that x evolves in accordance with the specified dynamics. The *Differential Skip (DX)* axiom guarantees the existence of a trivial solution with zero duration whenever a formula F is initially satisfied, meaning the system may remain in place (see Remark 5). The *Differential Invariant (DI)* axiom provides a key condition for proving properties under the box modality. For instance, if $x\geq 0$ initially and $x'\geq 0$ throughout, then $x\geq 0$ at all future times, allowing reasoning about system invariants. The *Differential Refinement (DR)* axiom captures a simple principle: adding additional constraints only restricts the system's behavior. The *Algebraic Ghost (AG)* axiom generalizes differential cuts [25,22] from dL to encode observable system behavior. This is particularly useful when applying the DE axiom to derivatives. Since the side condition prevents DE from being directly applied to differential variables, we can introduce a new dummy variable $y=x'$ to track the behavior of x', allowing the reasoning to proceed. The *Bounded Differential Ghost (BDG)* axiom allows expressing new differential relationships between variables as long as the new variable y has bounds expressible in terms of prior variables. The *Ghost Switching (GS)* axiom allows modularly decomposing multi-mode

differential-algebraic programs into a loop over a non-deterministic choice. This decomposition exploits analyticity and is *only* sound if the mode consistency formula is preserved along the evolution of the system. The key property, that the GS axiom needs for soundness is that a real-analytic function can only switch finitely many times in and out of a semi-algebraic set in bounded time. This is captured in the following lemma:

Lemma 2. *Let $T \geq 0$ and $\varepsilon > 0$. Further, let $\varphi \in \mathcal{A}_{\bar{x}}(-\varepsilon, T+\varepsilon)$ be a flow and F be a semi-algebraic formula. Let $(\tau_k)_{k\in\mathbb{N}} \subseteq [0,T]$ be a strictly monotone sequence with $\varphi(\tau_k) \in [\![F]\!]$ for all $k \in \mathbb{N}$. Then, there exists a $k_0 \in \mathbb{N}$ such that $\varphi(\tau) \in [\![F]\!]$ for all $\tau \geq \tau_{k_0}$ if the sequence is increasing and for all $\tau \leq \tau_{k_0}$ if the sequence is decreasing.*

The intuition behind this lemma is that if there exists a sequence of distinct time points where the flow satisfies F, then eventually, the system must satisfy F continuously from some point onward. Otherwise, the flow would have to switch infinitely many times in and out of F, which is incompatible with the regularity of real-analytic functions and semi-algebraic sets.

Remark 9. The formula $[\{\bar{x}:F\}][\{\bar{x}:F\}]P \leftarrow [\{\bar{x}:F\}]P$ is not a valid axiom of dAL, even though a corresponding axiom holds in dL. Consider, for instance:

$$\{x,t:(x' = t \wedge t \leq 0 \vee x' = -t \wedge t \geq 0) \wedge t' = 1\}.$$

This system exhibits a discontinuity in the second derivative of x at $t = 0$, which prevents the existence of a real-analytic flow from $t < 0$ to $t > 0$, because real-analytic functions are smooth. However, it is possible to construct two separate flows—one for $t \leq 0$ and one for $t \geq 0$—that agree in both x and x'. This shows a mode transition can occur via a composition of two flows, even though a single real-analtyic flow may not exist.

Some derived axioms are quite useful in practice:

Theorem 2. *The following are derived axioms of* dAL*:*

AR $\quad \forall\bar{x}\forall\bar{x}'(F \to G) \to ([\{\bar{x}:G\}]P \to [\{\bar{x}:F\}]P)$

DC $\quad [\{\bar{x}:F\}]G \to ([\{\bar{x}:F \wedge G\}]P \leftrightarrow [\{\bar{x}:F\}]P)$

∧DE $\quad [\{\bar{x}:F \wedge e = 0\}]P \leftrightarrow [\{\bar{x}:F \wedge e = 0 \wedge (e)' = 0\}]P \quad (\bar{x}' \notin e)$

The *Algebraic Refinement (AR)* axiom is for purely algebraic transformations. If F implies G, then all ways of following F also satisfy P if all ways of following the more general G do. For instance, AR allows rewriting $[\{x,y:x' = 1 \wedge y' = x'\}]P$ to $[\{x,y:x' = 1 \wedge y' = 1\}]P$. The *Differential Cut (DC)* axiom is one of the standard differential axioms of dL [25], allowing a DAP to be equivalently augmented with an invariant property G. It can be derived by AG and does not introduce a new variable each time. The *Conjunctive Differential Effect (∧DE)* axiom is useful to augment the system with its differential properties. For examples, we can transform the system $[\{x:y' = 2y \wedge x = y\}]P$ to $[\{x:y' = 2y \wedge x = 5y \wedge x' = 5y'\}]P$, deriving the differential properties of x from its algebraic relationship to y.

Example 6. Based on the derived ∧DE axiom, one might be tempted to assume the following equivalence holds:

$$[\{\bar{x} : F \vee e = 0\}]P \leftrightarrow [\{\bar{x} : F \vee (e = 0 \wedge (e)' = 0)\}]P.$$

However, this equivalence in not valid in general. To see why, consider the following differential-algebraic systems:

$$\alpha \equiv \{x : x' = 1\} \equiv \{x : (x' = 1 \wedge x \neq 0) \vee (x' = 1 \wedge x = 0)\},$$
$$\beta \equiv \{x : (x' = 1 \wedge x \neq 0) \vee (x' = 1 \wedge x = 0 \wedge x' = 0)\}.$$

In system α, x evolves according to $x' = 1$, allowing continuous progression from $x < 0$ to $x > 0$ without obstruction. In contrast, system β imposes an additional constraint at $x = 0$, requiring $x' = 0$ at that point. This restriction causes the system to get stuck at the transition between modes, as x' cannot jump from $x' = 1$ to $x' = 0$. Consequently, the behavior of the DAPs α and β is not equivalent.

3.2 Completeness

Prior work has shown that the natural number are definable using differential equations [19]. As a consequence, Gödel's incompleteness theorem applies to dAL's proof calculus. Nevertheless, there are completeness results for differential invariants in dL [27]. Since our proof calculus builds on top of dL's proof calculus, it inherits these completeness results for systems which are reducible to polynomial ordinary differential equations.

4 Applications

In this section, we demonstrate the usefulness of our proof calculus through two examples. Both examples focus on the *index reduction* of DAPs. The *differentiation index* of a differential-algebraic equation is the minimum number of times it must be differentiated in order to uniquely determine the derivative of each state variable in terms of the others [4, p. 179, Def. 4]. In the first example, we apply the AR and DE axioms in a straightforward manner, successively differentiating the algebraic constraints to obtain a reduced system. In the second example, a state-dependent singularity in the differentiated constraint necessitates careful case distinction using the GS axiom.

Example 7. We now address the Euclidean pendulum from the introduction, showing how our dAL calculus can be used to eliminate the unknown multiplier λ. Let $\bar{x} = (x, y, v, w)$ denote the state variables, and define the differential-algebraic constraint

$$D(\lambda) \equiv x' = v \wedge v' = \lambda x \wedge y' = w \wedge w' = \lambda y - g \wedge x^2 + y^2 = 1.$$

Our goal is to simplify the dAL-formula $[\{\bar{x}, \lambda : D(\lambda)\}]P$, which states that a post-condition P holds after every evolution of the system. Since dAL only defines the box modality, we use an arbitrary predicate P as a placeholder to reason about system equivalence. For simplicity, we focus only on one part of the equivalence, as the reverse is analogous. The derivations are presented as inference chains, read bottom-up. In the following, we write $\mathbb{R}$ to denote proof steps justified by real-arithmetic, and $\mathbb{F}$ for steps justified by first-order logical reasoning.

The multiplier λ appears explicitly in the constraint, but its time-derivative is unspecified. To uncover its implicit dynamics, we differentiate the holonomic constraint $x^2 + y^2 = 1$ twice, thereby obtaining an explicit expression for λ.

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash [\{\bar{x}, \lambda : D(\lambda) \land xv + yw = 0 \land \lambda = gy - (v^2 + w^2)\}]P}{\vdash [\{\bar{x}, \lambda : D(\lambda) \land xv + yw = 0 \land x'v + xv' + y'w + yw' = 0\}]P}\,\mathbb{R},\text{AR}}{\vdash [\{\bar{x}, \lambda : D(\lambda) \land xv + yw = 0\}]P}\,\land\text{DE}}{\vdash [\{\bar{x}, \lambda : D(\lambda) \land 2xx' + 2yy' = 0\}]P}\,\mathbb{R},\text{AR}}{\vdash [\{\bar{x}, \lambda : D(\lambda)\}]P}\,\land\text{DE}$$

Finally, we use algebraic refinement and differential refinement to remove λ from the formula with $h(x, y, v, w) = gy - (v^2 + w^2)$:

$$\dfrac{\dfrac{\vdash [\{\bar{x} : D(gy - (v^2 + w^2)) \land xv + yw = 0\}]P}{\vdash \forall\lambda\forall\lambda'[\{\bar{x}, \lambda : D(h(x, y, v, w)) \land xv + yw = 0 \land \lambda = h(x, y, v, w)\}]P}\,\text{DR}}{\vdash [\{\bar{x}, \lambda : D(\lambda) \land xv + yw = 0 \land \lambda = gy - (v^2 + w^2)\}]P}\,\mathbb{F},\text{AR}$$

The open goal contains the transformed system, in which the multiplier λ has been eliminated. The result is a fully specified ODE, enabling the application of completeness results for differential invariants [27].

Note that we did not need to use the GS axiom in the previous example, because we were able to find a globally defined explicit expression for λ. We see that this is not always the case in our next example:

Example 8. Consider the system:

$$\{x, y : x' = y \land x^2 + y^2 = 1\}.$$

The variable x evolves depending on the value of y, and both variables are constrained to a circle of radius one. Our goal is to derive an explicit expression for y'. We start by differentiating the circle constraint using the ∧DE axiom and do some equivalence transformations with the AR axiom:

$$\dfrac{\dfrac{\vdash [\{x, y : x' = y \land x^2 + y^2 = 1 \land (x + y')y = 0\}]P}{\vdash [\{x, y : x' = y \land x^2 + y^2 = 1 \land xx' + y'y = 0\}]P}\,\mathbb{R},\text{AR}}{\vdash [\{x, y : x' = y \land x^2 + y^2 = 1\}]P}\,\mathbb{R},\text{AR},\land\text{DE}$$

Now, we observe that there are two cases for the constraint: $x + y' = 0$ or $y = 0$. Next, we define

$$F(x, y) \equiv x' = 0 \land x^2 = 1 \land y = 0,$$
$$G(x, y) \equiv x' = y \land x^2 + y^2 = 1 \land y' = -x.$$

Here, $F(x,y)$ represents the special case where x is at its extrema (± 1) and remains constant, while $G(x,y)$ describes the general evolution of the system along the unit circle. We split the system into a disjunction of two modes using the AR axiom:

$$\mathbb{R},\ \text{AR}\ \frac{\mathbb{R},\ \text{AR}\ \dfrac{\vdash [\{x,y : F(x,y) \lor G(x,y)\}]P}{\vdash [\{x,y : x' = y \land x^2 + y^2 = 1 \land (y' = -x \lor y = 0)\}]P}}{\vdash [\{x,y : x' = y \land x^2 + y^2 = 1 \land (x + y')y = 0\}]P}$$

Finally, applying the GS axiom we have:

$$\text{GS}\ \frac{\begin{array}{l}\vdash [\{\{x,y : F(x,y) \lor G(x,y)\}\}](F(x,y) \rightsquigarrow_{\bar{x}} G(x,y) \land G(x,y) \rightsquigarrow_{\bar{x}} F(x,y)) \\ \vdash [(\{x,y : F(x,y)\} \cup \{x,y : G(x,y)\})^*]P\end{array}}{\vdash [\{x,y : (x' = 0 \land x^2 = 1 \land y = 0) \lor (x' = y \land x^2 + y^2 = 1 \land y' = -x)\}]P}$$

The top goal closes by simplification of progress formulas and real arithmetic (See [12]). The second goal contains a decomposed version of the original system, formulated as a loop over a nondeterministic choice of two simplified systems. This representation shows how singularities in the differentiated constraints give rise to distinct modes of evolution. To further illustrate these dynamics, Fig. 3 provides a visualization of the system in phase space, showcasing its trajectory and behavior over time.

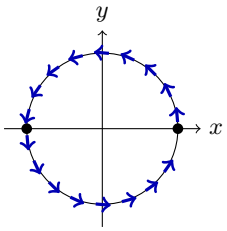

Fig. 3. Phase portrait on the unit circle with two stationary modes.

Remark 10. The pattern observed in the previous example can be formulated in a more general way: Given a semi-explicit system of the form

$$\{\bar{x}, \bar{y} : \bar{x}' = f(\bar{x}, \bar{y}) \land g(\bar{x}, \bar{y}) = 0\},$$

we can transform it using the $\land$DE axiom into the equivalent system:

$$\{\bar{x}, \bar{y} : \bar{x}' = f(\bar{x}, \bar{y}) \land g(\bar{x}, \bar{y}) = 0 \land \frac{\partial g}{\partial x}(\bar{x}, \bar{y}) f(\bar{x}) + \frac{\partial g}{\partial y}(\bar{x}, \bar{y}) y' = 0\}.$$

This transformation introduces an implicit constraint on $\bar{y}'$, but solving for $\bar{y}'$ is only possible if $\frac{\partial g}{\partial y}(\bar{x}, \bar{y})$ is non-singular. When this condition fails, the system encounters a singularity, highlighting the necessity of the GS axiom to distinguish between different cases.

5 Related Work

Differential dynamic logic dL has established the logical foundations for reasoning about hybrid systems [24,25]. However, dL does not support reasoning about differential-algebraic equations, which are natural for modeling many physical systems. To fill this gap, an extension of dL to differential-algebraic equation systems was proposed previously in a sequent calculus formulation [22]. However, each rule of this formulation requires a special normalization of differential-algebraic systems. In contrast, our calculus is given by a small set of axioms and does not require such special transformation. Additionally, a challenge in reasoning about differential-algebraic systems is understanding the conditions under which systems can be correctly decomposed. For our calculus, we exploit progress formulas, which have been used for reasoning about invariant properties of differential equations [27,11]. Contributions by Mattsson [18], Pantelides [20], and Benveniste [1] have addressed the challenge of consistently initializing differential-algebraic systems, primarily in the context of simulation. Other works, such as [5,6], focus on the minimization of polynomial ODE systems using bisimulation techniques, also with a focus on simulation. These approaches rely on model transformation techniques aimed at simulation and are not grounded in rigorous logical reasoning principles. In contrast, dAL enables transformations within an axiomatic proof calculus and goes beyond mere model simplifications as it enables the specification and verification of dynamic properties such as safety and liveness.

6 Conclusion

This paper introduced a sound axiomatic proof calculus for (differential-form) differential-algebraic dynamic logic dAL, providing a firm logical foundation for reasoning about differential-algebraic programs. This allows more flexible specification of continuous-time models, along with a framework for their rigorous transformations to obtain logically equivalent forms. The Euclidean pendulum example demonstrated how dAL supports the specification of systems with undetermined dynamics and their reduction to ordinary differential equations. Future work will explore how the sound logical reductions of dAL support automatic model transformation.

Acknowledgements

This work was funded by an Alexander von Humboldt Professorship.

Declaration of Interests

The authors are not aware of any conflicts of interest related to the content of this work.

References

1. Benveniste, A., Caillaud, B., Elmqvist, H., Ghorbal, K., Otter, M., Pouzet, M.: Structural Analysis of Multi-Mode DAE Systems. In: Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control. pp. 253–263. ACM, Pittsburgh Pennsylvania USA (Apr 2017). doi: 10.1145/3049797.3049806
2. Bhat, S.P., Bernstein, D.S.: Example of indeterminacy in classical dynamics **36**(2), 545–550. doi: 10.1007/BF02435747
3. Branicky, M.S.: General hybrid dynamical systems: Modeling, analysis, and control. In: Alur, R., Henzinger, T.A., Sontag, E.D. (eds.) Hybrid Systems III. pp. 186–200. Springer, Berlin, Heidelberg (1996). doi: 10.1007/BFb0020945
4. Campbell, S.L., Gear, C.W.: The index of general nonlinear DAEs. Numerische Mathematik **72**(2), 173–196 (Dec 1995). doi: 10.1007/s002110050165
5. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: Symbolic computation of differential equivalences. In: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 137–150. ACM, St. Petersburg FL USA (Jan 2016). doi: 10.1145/2837614.2837649
6. Cardelli, L., Tribastone, M., Tschaikowski, M., Vandin, A.: ERODE: A Tool for the Evaluation and Reduction of Ordinary Differential Equations. In: Legay, A., Margaria, T. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 310–328. Springer, Berlin, Heidelberg (2017). doi: 10.1007/978-3-662-54580-5_19
7. Cleaveland, R., Mitsch, S., Platzer, A.: Formally Verified Next-generation Airborne Collision Avoidance Games in ACAS X. ACM Trans. Embed. Comput. Syst. **22**(1), 10:1–10:30 (Oct 2022). doi: 10.1145/3544970
8. Davoren, J., Nerode, A.: Logics for hybrid systems. Proceedings of the IEEE **88**(7), 985–1010 (Jul 2000). doi: 10.1109/5.871305
9. Fritzson, P., Engelson, V.: Modelica — A unified object-oriented language for system modeling and simulation. In: Jul, E. (ed.) ECOOP'98 — Object-Oriented Programming. pp. 67–90. Springer, Berlin, Heidelberg (1998). doi: 10.1007/BFb0054087
10. Fulton, N., Mitsch, S., Quesel, J.D., Völp, M., Platzer, A.: KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In: Felty, A., Middeldorp, A. (eds.) CADE. LNCS, vol. 9195, pp. 527–538. Springer, Berlin (2015). doi: 10.1007/978-3-319-21401-6_36
11. Ghorbal, K., Sogokon, A.: Characterizing positively invariant sets: Inductive and topological methods. Journal of Symbolic Computation **113**, 1–28 (Nov 2022). doi: 10.1016/j.jsc.2022.01.004
12. Hellwig, J., Platzer, A.: A Real-Analytic Approach to Differential-Algebraic Dynamic Logic (May 2025). doi: 10.48550/arXiv.2505.19323
13. Henzinger, T.: The theory of hybrid automata. In: Proceedings 11th Annual IEEE Symposium on Logic in Computer Science. pp. 278–292 (Jul 1996). doi: 10.1109/LICS.1996.561342
14. Kabra, A., Mitsch, S., Platzer, A.: Verified Train Controllers for the Federal Railroad Administration Train Kinematics Model: Balancing Competing Brake and Track Forces. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **41**(11), 4409–4420 (Nov 2022). doi: 10.1109/TCAD.2022.3197690
15. Kabra, A., Mitsch, S., Platzer, A.: Verified train controllers for the Federal Railroad Administration train kinematics model: Balancing competing brake and track forces. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **41**(11), 4409–4420 (2022). doi: 10.1109/TCAD.2022.3197690

16. Krantz, S.G., Parks, H.R.: A Primer of Real Analytic Functions. Birkhäuser, Boston, MA (2002). doi: 10.1007/978-0-8176-8134-0
17. Lin, Q., Mitsch, S., Platzer, A., Dolan, J.M.: Safe and Resilient Practical Waypoint-Following for Autonomous Vehicles. IEEE Control Systems Letters **6**, 1574–1579 (2022). doi: 10.1109/LCSYS.2021.3125717
18. Mattsson, S.E., Söderlind, G.: Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives. SIAM Journal on Scientific Computing **14**(3), 677–692 (May 1993). doi: 10.1137/0914043
19. Olivetti, N. (ed.): Automated Reasoning with Analytic Tableaux and Related Methods, 16th International Conference, TABLEAUX 2007, Aix en Provence, France, July 3-6, 2007, Proceedings, LNCS, vol. 4548. Springer, Berlin (2007)
20. Pantelides, C.C.: The Consistent Initialization of Differential-Algebraic Systems. SIAM Journal on Scientific and Statistical Computing **9**(2), 213–231 (Mar 1988). doi: 10.1137/0909014
21. Platzer, A.: Differential dynamic logic for hybrid systems. J. Autom. Reas. **41**(2), 143–189 (2008). doi: 10.1007/s10817-008-9103-8
22. Platzer, A.: Differential-algebraic dynamic logic for differential-algebraic programs. J. Log. Comput. **20**(1), 309–352 (2010). doi: 10.1093/logcom/exn070
23. Platzer, A.: Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics. Springer, Heidelberg (2010). doi: 10.1007/978-3-642-14509-4
24. Platzer, A.: Logics of dynamical systems. In: LICS. pp. 13–24. IEEE, Los Alamitos (2012). doi: 10.1109/LICS.2012.13
25. Platzer, A.: A complete uniform substitution calculus for differential dynamic logic. J. Autom. Reas. **59**(2), 219–265 (2017). doi: 10.1007/s10817-016-9385-1
26. Platzer, A.: Logical Foundations of Cyber-Physical Systems. Springer (2018). doi: 10.1007/978-3-319-63588-0
27. Platzer, A., Tan, Y.K.: Differential equation invariance axiomatization. J. ACM **67**(1), 6:1–6:66 (2020). doi: 10.1145/3380825
28. Tan, Y.K., Mitsch, S., Platzer, A.: Verifying switched system stability with logic. In: Bartocci, E., Putot, S. (eds.) Hybrid Systems: Computation and Control (part of CPS Week 2022), HSCC'22. ACM (2022). doi: 10.1145/3501710.3519541