Relational Differential Dynamic Logic

Juraj Kolčák[*†], Ichiro Hasuo[*‡], Jérémy Dubut[*§], Shin-ya Katsumata[*], David Sprunger[*] and Akihisa Yamada[*]
[*] National Institute of Informatics, Tokyo, Japan
[†] LSV, CNRS & ENS Paris-Saclay, Université Paris-Saclay, France
[‡] The Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan
[§] Japanese-French Laboratory for Informatics, Tokyo, Japan

***Abstract*—In the field of quality assurance of *hybrid systems* (that combine continuous physical dynamics and discrete digital control), Platzer's *differential dynamic logic* (dL) is widely recognized as a deductive verification method with solid mathematical foundations and sophisticated tool support. Motivated by benchmarks provided by our industry partner, we study *relational extension* of dL, aiming to formally prove statements such as "earlier deployment of emergency brake decreases collision speed." A main technical challenge here is to relate two states of two dynamics *at different time points*. Our main contribution is a theory of a suitable *simulation* notion (a relational extension of *differential invariant* that is a central proof method in dL), and a derived technique of *time stretching*. The latter features particularly high applicability, since the user does not have to synthesize a simulation out of the air. We derive new inference rules for dL from these notions, and demonstrate their use using a couple of automotive case studies.**

***Index Terms*—hybrid system, cyber-physical system, formal verification, theorem proving, dynamic logic, differential equation, relational reasoning**

I. Introduction

A. Hybrid Systems

It has been more than a decade since the term *cyber-physical system* (CPS) was coined and became a target of ever-growing research efforts. In particular, *quality assurance* of CPS is attracting renewed attention in the last few years, with the rise of *automated driving*. In the foreseeable future, millions of cars will be driving on streets with the degree of automation that is greater than ever by magnitudes. Ensuring safety and reliability of those automated driving systems is therefore a pressing social and economic challenge.

Quality assurance of CPS poses a unique academic challenge, too, namely the combination of continuous physical dynamics and discrete digital control. This important aspect of CPS has been referred to as *hybrid system*. In the research efforts on hybrid systems, two communities have naturally joined forces: *control theory* whose traditional application domain is continuous dynamics, and *formal methods* that have mainly focused on the analysis of software systems. The collaboration has produced a number of novel scientific observations and practical techniques. They include the use of formal methods techniques in control (bisimilarity [7], temporal logic specification [6], and so on), and conversely, transfer of control theory notions (such as Lyapunov functions) to formal methods. See e.g. [20].

Thanks are due to Kenji Kamijo, Yoshiyuki Shinya, and Takamasa Suetomi from Mazda Motor Corporation for helpful discussions. The authors are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST. I.H. is supported by Grants-in-Aid No. 15KT0012, JSPS. The work is done during J.K.'s internship at National Institute of Informatics, Tokyo, Japan.

B. Deductive Verification of Hybrid Systems

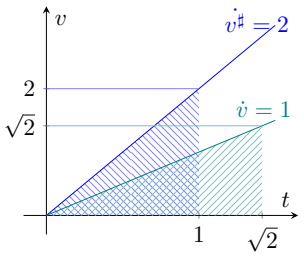
In the formal methods community, a traditional classification of techniques consists of *model checking* (that is usually automata-based and automatic), and *deductive verification* (that is based on logic and can be automated or interactive).[1] The applicability of automated model checking techniques is naturally limited for hybrid systems, since the latter has continuous state spaces and thus the search spaces become uncountable. This has led to the active study of *discrete abstraction* of hybrid dynamics, where a principal method is the use of *approximate bisimulations* derived from suitable Lyapunov functions. See e.g. [7].

The deductive approach, in contrast, is not necessarily prohibited for hybrid systems. Logical formulas can stand for infinitely many states thanks to their free variables; and formal deduction of those formulas is no less valid even if the semantic domain is the set of reals, for example.

That said, it is still a hard task to design a deductive system that is actually *useful* in hybrid system verification. Such a system should come with concise and intuitive syntax for expressing continuous-time dynamics (i.e. differential equations), as well as powerful and versatile reasoning principles for such dynamics.

Platzer's *differential dynamic logic* dL [15] adequately meets these criteria. Its syntax is systematic and intuitive, extending the classic formalism of *dynamic logic* [8] with differential equations as programs. Its proof rules encapsulates several essential proof principles about differential equations, notable among which are *differential invariant* (DI) for universal properties and *side deduction* for existential properties. The logic dL has served as a general platform that accommodates a variety of techniques, including those which come from real algebraic geometry. See e.g. [16], where Grönwall's inequality is used to derive the so-called Darboux inequality rule. Furthermore, dL comes with sophisticated tool support: the latest tool KeYmaera X [12] comes with graphical interface for interactive proving and a number of automation heuristics.

[1] The classification is not definite: there are some works that eagerly aim at integration of (logic- or type system-based) theorem proving and model checking, including [10].



The two hatched areas designate the traveled distances ($x = x^\sharp = 1$). We can compute the collision speeds ($v = \sqrt{2}$ and $v^\sharp = 2$) via the closed-form solutions of the differential equations (1), concluding $v \le v^\sharp$ when $x = x^\sharp = 1$.

Fig. 1. An ad-hoc proof for Example I.1

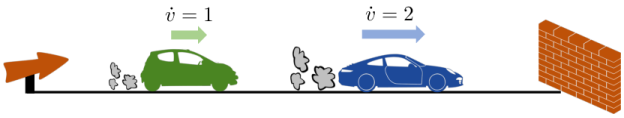
C. Relational Reasoning on Hybrid Systems

In this work, we aim at a new direction in the field of deductive verification for hybrid systems. We are motivated by the following example; it came up in our collaboration with automotive industry.

Example I.1 (leading example: collision speed)**.** Consider two cars C and C$^\sharp$, whose positions and velocities are denoted by $x, x^\sharp$ and $v, v^\sharp$, respectively. Their dynamics are governed by the following differential equations.

$$\dot{x} = v, \quad \dot{v} = 1; \qquad \dot{x^\sharp} = v^\sharp, \quad \dot{v^\sharp} = 2. \tag{1}$$

We assume that they both start at the same position ($x = x^\sharp = 0$), with the initial velocity $v = v^\sharp = 0$, and they both drive towards a wall at the position $x = x^\sharp = 1$. Our question is: *which car has a greater collision speed?*



A natural answer is the second car C$^\sharp$ that accelerates harder (see (1)). Therefore we aim to prove the following claim.

$$\text{The two dynamics in (1), with the initial states } x = x^\sharp = v = v^\sharp = 0, \text{ exhibit } v \le v^\sharp \text{ when } x = x^\sharp = 1. \tag{2}$$

We further aim to prove the claim in a *formal* and *modular* manner, so that the proof methods we use apply to other examples of the same nature. Therefore, differential dynamic logic dL is naturally a platform of our choice.

It turns out that this *relational reasoning* that compares two different dynamics is not easy in dL. Giving an ad-hoc proof is not hard: see Figure 1, where the proof relies on the closed-form solutions of the dynamics. However, we prefer not to use closed-form solutions (that are not always available).

Moreover, it is desirable that our proof is solely by *local reasoning*, without inspecting the global properties of the dynamics. The logic dL offers a powerful rule for that purpose, namely the *differential invariant* (DI) rule. The rule roughly says: if $g(x)$ is initially nonnegative, and the time derivative of g (along the dynamics—it is given by the Lie derivative) is nonnegative, then $g(x)$ stays nonnegative *no matter how long* the dynamics runs. Note that a global safety property is reduced to a local property about a Lie derivative; the DI rule is a continuous-time analogue of the celebrated proof method by *loop invariants*.

The discussion so far already suggests a viable strategy: instead of (differential) invariants for unary safety proofs, in relational reasoning we can use "binary invariants," that is, *(bi)simulations*. Our main contribution of this paper is to develop a theory about simulations and related notions between continuous-time dynamics, and to embody the theory in the form of concise and useful proof rules in dL.

Note that such simulations necessarily relate *different time points* of the two dynamics. For example, in Figure 1, comparing v and $v^\sharp$ at the same time point does not suffice: at $t = t^\sharp = 1$, the car C$^\sharp$ hits the wall but C does not yet. Although it is easy to see that $v < v^\sharp$ at time 1, C continues driving, in the course of which C's velocity v keeps growing.

D. Technical Contributions

In this paper we make the following technical contributions.

1) **(Formulation of relational reasoning in dL)** We formulate the problem of relational reasoning in differential dynamic logic dL. We focus on comparing different continuous-time dynamics (therefore comparing different discrete-time controls is currently out of our scope). We observe that this problem is expressible in the syntax of dL (e.g. in [14], [16]); nevertheless we introduce a syntactic shorthand for readability (§III-A).
2) **(Simulation and the (SIM) rule)** We define *simulation* between two dynamics, and formulate its soundness result for relational reasoning (§III-B). The latter result gives rise to a syntactic proof rule (SIM), in §III-C.
3) **(Time stretching and the (TS) rule)** While foundational, our theory up to this point does not answer the question of how to find a simulation. Restricting to so-called *equational exit conditions*, we devise a syntactic method for synthesizing a simulation-like notion, called *time stretch function*. In §IV we develop the theory, eventually leading to a succinct proof rule (TS).
   We find the (TS) rule powerful: it *synchronizes* two dynamics and reduces relational reasoning to common proof obligations dL. For the latter, we can fully exploit the advanced proof rules already developed in dL.
4) **(Additional rules including the (DII) rule)** Besides the two principal new rules (SIM) and (TS), we introduce some dL rules that we needed for our case studies (§V). Notable is what we call the different *inductive* invariant (DII) rule; it is much like the differential invariant (DI) rule in dL (see e.g. [16]) but it allows to additionally assume the target inequality itself in an evolution domain constraint.
   These rule seem to be new, although they might be derivable from the comprehensive calculus in [14].
5) **(Case studies)** In addition to Example I.1, we conduct three case studies that are more complex (§VI). They are inspired by our industry collaboration.

E. Relational Reasoning in Industry Practice

The current work stemmed out of our collaboration with an industry partner. From this experience, we argue that relational reasoning on hybrid systems has ample practical significance.

Firstly, *monotonicity* properties as exemplified in Example I.1 abound even in real-world examples. Monotonicity properties take the following form: we are given a parametrized model $M(p)$; and we aim to show that

$$p_1 < p_2 \text{ implies } M(p_2) \text{ is less safe than } M(p_1). \quad (3)$$

These properties occur often especially in the context of *product lines*, such as Euro car segments A–F.

Secondly, proof goals in relational reasoning (such as monotonicity) tend to be easier to establish, compared to (unary, non-relational) safety specifications. The proof of the latter is generally hard, requiring inspection of every small detail of the system. Therefore we expect that, in relational reasoning, the general difficulty of scaling deductive methods to real-world examples is less of a problem.

Thirdly, relational reasoning (especially on monotonicity) can serve as a powerful technique in *test-case reduction*. As we discussed previously, it can happen that a real-world example is too complex for its safety property to be deductively verified. In this situation, one would turn to empirical quality assurance methods (i.e. *testing*). If a monotonicity property (such as (3)) can be deductively established (it is often easier than safety), the monotonicity property allows us to focus our testing efforts to the extreme case $M(p_{\max})$. This spares all the test cases that we would have spent for $M(p)$ with smaller p's.

F. Related Work

We already mentioned some related works, including those on dL. Some other related works are discussed here.

Simulink (from the Mathworks, Inc.) is the industry standard in modeling hybrid systems; therefore, any quality assurance method is desired to be able to take Simulink models as its input. This is not easy for formal verification methods, notably because Simulink models do not have rigorous semantics. The recent work [11] tackles this problem, identifying a fragment of Simulink, and devising a translator from Simulink models to dL programs. Their translation is ingenious, and their tool is capable of proving rather complicated properties when used in combination with KeYmaera X [12].

This work is about relational extension of dL. Relational extension of the *Floyd–Hoare logic*—as the discrete-time prototype of dL—has been energetically pursued, too, especially in the context of *differential privacy*. See e.g. [1], [3], [4].

In deductive verification of hybrid systems, an approach alternative to Platzer's dL uses *nonstandard analysis* [17] and regards continuous dynamics as if it is discrete [18], [19]. The logic built on top is literally the same as the classic Floyd–Hoare logic, whose soundness in the extended hybrid setting is shown by a model-theoretic result called the *transfer principle*. Its tool support has been pursued, too [9].

G. Organization and Notations

In §II we recall the differential dynamic logic dL: its syntax, semantics and some proof rules. Our target problem of relational reasoning is formulated in §III, where we introduce a syntactic shorthand $[\langle\!\langle \delta \mid \delta^\sharp \rangle\!\rangle E] B$. The theory of simulation and the (SIM) proof rule are introduced there, too. The theory (and its limitation in applicability) motivates our theory of time stretching in §IV, where we restrict to equational exit conditions. The resulting (TS) rule, combined with some additional rules that we introduce in §V, are exploited in the three case studies in §VI. We conclude in §VII.

II. Preliminaries: the Logic dL

We base ourselves on *differential dynamic logic* (dL) [13], [14]. While some key concepts and proof rules are recalled here, the reader is referred to [13], [14] for full details.

A. Syntax

We assume a set of variables that are denoted by $x, y, \ldots$, and a set of function symbols $f, g, \ldots$ that come with the arity information. We write $\boldsymbol{x}$ for a vector of variables $x_1, \ldots, x_n$ (following [16]), and write $f(\boldsymbol{x})$ for the application of an n-ary function symbol f to these variables. Finally, we write boldface $\boldsymbol{f}(\boldsymbol{x})$ to designate a vector $\boldsymbol{f}(\boldsymbol{x}) = (\, f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}) \,)$ of function applications.

We first introduce hybrid programs.

Definition II.1 (hybrid programs, differential dynamics)**.** *Hybrid programs* are defined by the following BNF expression. Here $\boldsymbol{x}$ is a vector of variables, $\boldsymbol{f}(\boldsymbol{x})$ is a vector of function applications where all the function symbols in $\boldsymbol{f}$ has the arity $|\boldsymbol{x}|$, and P, Q are formulas of real arithmetic.

$$\alpha_1, \alpha_2 \ ::= \ ?P \mid \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q \mid \alpha_1; \alpha_2 \mid \alpha_1 \cup \alpha_2$$

Hybrid programs in the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q$ are called *differential dynamics*, where $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ is called its *differential equation system* and the formula Q is called its *evolution domain constraint*. We write simply $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ instead of $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, \text{TRUE}$.

The hybrid programs here include standard constructs such as tests ($?P$), sequential compositions and non-deterministic branchings. The *assignment* and *iteration* constructs (i.e. $x := f(x)$ and α^*) in dL are dropped for simplicity, since we do not use them in this paper. Such constructs can, however, be included if necessary.

In this paper, we are especially interested in analyzing differential dynamics $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q$. The intention is that the dynamics develops according to the differential equation $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, as long as the evolution domain constraint Q is satisfied.

Definition II.2 (dL formulas)**.** Formulas are defined by the following grammar, where f_1 (resp. f_2) is a function symbol with arity $|\boldsymbol{x}_1|$ (resp. $|\boldsymbol{x}_2|$), x is a variable, α is a hybrid program and $\sim \in \{<, \leq, >, \geq, =\}$ is a comparison operator.

$$\varphi, \varphi_1, \varphi_2 ::= f_1(\boldsymbol{x}_1) \sim f_2(\boldsymbol{x}_2) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \forall x.\, \varphi \mid [\alpha]\, \varphi \mid \langle\alpha\rangle\varphi$$

Although dL could allow arbitrary formulas in tests $?P$ and as evolution domain constraints Q, we limit ourselves to first-order formulas in those constructs. This is also done in [16].

B. Semantics

In this section we briefly recall the interpretation of a dL formula φ from Definition II.2, following [14], [16].

Let $\mathcal{V} = \{x_1, \ldots, x_n\}$ be the set of variables that occur in the given formula. A *state* $\omega \in \mathbb{R}^{\mathcal{V}}$ is then an assignment of a real value to each variable in $\mathcal{V}$. An *interpretation* I assigns, to each function symbol f of arity m, a continuous function $I(f) : \Omega \to \mathbb{R}$, where the domain $\Omega \subseteq \mathbb{R}^m$ is an open subset.

Remark II.3. For the semantics of dL, not much is required about the interpretation $I(f)$. It is however common to require that the differential equation $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ has a unique maximal solution. While the local Lipschitz property is enough (Picard–Lindelöf's theorem), we can also impose a stronger property, such as C^1, rationals, polynomials, etc. For the purpose of this paper, any of these suffices, although in the polynomial case we will need some trick to accommodate the canonical time stretch function (that is rational per se, see §IV).

From now on, for simplicity, we assume that $I(f)$ is C^1.

Given a formula φ, we write $I[\![\varphi]\!]$ for the set of all states in which φ holds in the interpretation I. The semantics of comparison operations and first order formulas are then defined as usual, e.g. $\omega \in I[\![f_1(y_1, \ldots, y_k)) \leq f_2(z_1, \ldots, z_l)]\!]$ if $I(f_1)(\omega(y_1), \ldots, \omega(y_k)) \leq I(f_2)(\omega(z_1), \ldots, \omega(z_l))$, and $\omega \in I[\![\neg\varphi]\!]$ if $\omega \notin I[\![\varphi]\!]$.

The semantics of hybrid programs are given as transition relations on states, i.e. $I[\![\alpha]\!] \subseteq \mathbb{R}^n \times \mathbb{R}^n$.

Definition II.4 (hybrid program semantics)**.** The semantics of hybrid programs are given inductively as follows.

(1) $I[\![?P]\!] = \{(\omega, \omega) \mid \omega \in I[\![P]\!]\}$.
(2) $I[\![\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q]\!] = \{(\omega_1, \omega_2) \mid \exists t \in \mathbb{R}_{\geq 0}.\, \omega_1 = \psi(0) \wedge \omega_2 = \psi(t) \wedge \forall t' \in [0, t].\, \psi(t') \in I[\![Q]\!]\}$ where ψ is a solution of the dynamics $\dot{\boldsymbol{x}} = I(\boldsymbol{f})(\boldsymbol{x})$ on $[0, t]$.
(3) $I[\![\alpha_1 \cup \alpha_2]\!] = I[\![\alpha_1]\!] \cup I[\![\alpha_2]\!]$,
(4) $I[\![\alpha_1; \alpha_2]\!] = I[\![\alpha_1]\!] \circ I[\![\alpha_2]\!]$.

The semantics of the modal formula $I[\![[\alpha]\varphi]\!]$ are then given as $\{\omega \mid (\omega, \omega') \in I[\![\alpha]\!]$ implies $\omega' \in I[\![\varphi]\!]\}$. For the dual operator: $I[\![\langle\alpha\rangle\varphi]\!] = I[\![\neg[\alpha]\neg\varphi]\!]$. These are all as in [16].

To ease notation, we write simply $[\![\varphi]\!]$ or $[\![\alpha]\!]$ instead of $I[\![\varphi]\!]$ or $I[\![\alpha]\!]$ respectively, the interpretation I being implicit.

C. Proof Rules

In this paper we use several dL proof rules taken from the literature; see e.g. [14], [16]. In particular, we heavily use those rules which deal with differential dynamics, such as the *differential invariant* (DI) and the *differential cut* (DC) rules.

Definition II.5 (some dL proof rules)**.** One can use the following proof rules in dL, where $(\sim, \simeq) \in \{(=, =), (>, \geq), (\geq, \geq)\}$ and $\mathcal{L}_f g(\boldsymbol{x})$ denotes the Lie derivative of the function symbol g with arity $\boldsymbol{x}$ with respect to the dynamics given by $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ [16].

$$\frac{\Gamma, Q \vdash g(\boldsymbol{x}) \sim 0 \quad \Gamma \vdash [\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q]\, \mathcal{L}_f g(\boldsymbol{x}) \simeq 0}{\Gamma \vdash [\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q]\, g(\boldsymbol{x}) \sim 0}\ \text{DI}$$

$$\frac{\Gamma \vdash [\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q]\, C \quad \Gamma \vdash [\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, (Q \wedge C)]\, \varphi}{\Gamma \vdash [\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q]\, \varphi}\ \text{DC}$$

$$\frac{Q \vdash \varphi}{\Gamma \vdash [\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q]\, \varphi}\ \text{DW}$$

The differential invariant rule (DI) is the central rule for proving safety properties [14], [16]: it reduces a global property of the dynamics to local reasoning by the means of Lie derivatives. The differential cut rule (DC) on the other hand allows the evolution domain constraint Q to be strengthened by a proposition C that is proven to be invariant throughout the dynamics. The differential weakening rule (DW) allows to prove the postcondition directly from the evolution domain constraint, disregarding the differential dynamics.

III. Relational Reasoning in Differential Dynamic Logic, and Simulations

We formulate the kind of relational reasoning in dL that we are interested in. It turns out that no extension of the syntax is needed, although we introduce a syntactic shorthand for readability. After doing so in §III-A, we present our theory of *simulation* in §III-B, eventually leading to a syntactic proof rule (SIM) in §III-C.

A. Relational Differential Dynamics (RDD) Formulas

Definition III.1 (relational differential dynamics (RDD) formula)**.** A dL formula A is called a *relational differential dynamics formula* (RDD formula) if it is of the following form:

$$A \equiv \Big(\big[\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q(\boldsymbol{x});\ \dot{\boldsymbol{x}^\sharp} = \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp) \,\&\, Q^\sharp(\boldsymbol{x}^\sharp);\ ?E\big]\, B \Big),$$

where we require that the variables in $\boldsymbol{x}$ and those in $\boldsymbol{x}^\sharp$ are disjoint. The expressions E and B are dL formulas containing variables from both $\boldsymbol{x}$ and $\boldsymbol{x}^\sharp$.

To improve readability we introduce the following syntax denoting the above dL formula:

$$A \equiv \quad \left[\left\langle\!\!\left\langle \frac{\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q(\boldsymbol{x})}{\dot{\boldsymbol{x}^\sharp} = \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp) \,\&\, Q^\sharp(\boldsymbol{x}^\sharp)} \right\rangle\!\!\right\rangle E\right] B. \tag{4}$$

It is also written horizontally as:

$$\begin{aligned} A \equiv \quad & \big[\langle\!\langle \delta \mid \delta^\sharp \rangle\!\rangle\, E\big]\, B, \\ \text{where} \quad & \delta \equiv (\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q(\boldsymbol{x})) \quad \text{and} \\ & \delta^\sharp \equiv \big(\dot{\boldsymbol{x}^\sharp} = \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp) \,\&\, Q^\sharp(\boldsymbol{x}^\sharp)\big). \end{aligned} \tag{5}$$

The formula E is called the *exit condition*; the formula B is called the *postcondition*.

Our interpretation of the RDD formula A in Definition III.1 is that δ and $\delta^\sharp$ run in parallel and get synchronized at the end by means of the exit condition E. It is important that the two dynamics can take different amounts of time; see §I-C.

Notation III.2 (solutions $\psi, \psi^\sharp$)**.** In the setting of Definition III.1, by Picard-Lindelöf's theorem, the differential equation $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ in δ has a unique maximal solution, for the initial state $\boldsymbol{x}_0$, and its interval of definition is of the form $[0, T_{\boldsymbol{x}_0})$, with $T_{\boldsymbol{x}_0} \in \mathbb{R}_{\geq 0} \cup \{\infty\}$. This unique solution of $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ from $\boldsymbol{x}_0$ is denoted by

$$\psi(\boldsymbol{x}_0, -) : \quad [0, T_{\boldsymbol{x}_0}) \longrightarrow \mathbb{R}^{|\boldsymbol{x}|}.$$

Similarly, the unique solution of $\dot{\boldsymbol{x}^\sharp} = \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp)$ in $\delta^\sharp$ from $\boldsymbol{x}^\sharp_0$ is denoted by $\psi^\sharp(\boldsymbol{x}^\sharp_0, -) : [0, T^\sharp_{\boldsymbol{x}^\sharp_0}) \to \mathbb{R}^{|\boldsymbol{x}^\sharp|}$. In case the initial state is obvious, we write simply $\psi(t)$ and $\psi^\sharp(t)$.

Example III.3 (collision speed)**.** The problem in Example I.1 is expressed as the following RDD formula: $A \equiv [\langle\!\langle \delta \mid \delta^\sharp \rangle\!\rangle E] B$, where

$$\delta \equiv \big(\dot{x} = v, \dot{v} = 1\big), \quad \delta^\sharp \equiv \big(\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = 2\big),$$
$$E \equiv \big(x = x^\sharp = 1\big), \quad \text{and } B \equiv \big(v \leq v^\sharp\big).$$

B. Simulations for RDD Formulas

Our simulation notion for RDD formulas is based on the following transition system T_δ induced by δ.

Definition III.4 (transition system T_δ)**.** Let $\delta \equiv (\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q(\boldsymbol{x}))$ be a differential dynamics. It gives rise to a transition system $T_\delta = (\mathbb{R}^{|\boldsymbol{x}|}, \xrightarrow{T_\delta})$, where the state space is the set $\mathbb{R}^{|\boldsymbol{x}|}$ of vectors, and the transition relation $\xrightarrow{T_\delta} \subseteq \mathbb{R}^{|\boldsymbol{x}|} \times \mathbb{R}^{|\boldsymbol{x}|}$ is defined as follows:

$$\boldsymbol{x}_1 \xrightarrow{T_\delta} \boldsymbol{x}_2 \quad \overset{\text{def.}}{\iff} \quad (\boldsymbol{x}_1, \boldsymbol{x}_2) \in [\![\delta]\!] .$$

The definition can be described using the solution ψ of $\delta \equiv (\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q(\boldsymbol{x}))$ as follows (cf. Notation III.2): $\boldsymbol{x}_1 \xrightarrow{T_\delta} \boldsymbol{x}_2$ if and only if there exists $t \in [0, T_{x_1})$ such that 1) $\boldsymbol{x}_2 = \psi(\boldsymbol{x}_1, t)$, and 2) $\psi(\boldsymbol{x}_1, t') \in [\![Q(\boldsymbol{x})]\!]$ for each $t' \in [0, t]$.

Definition III.5 (simulation)**.** Let $A \equiv [\langle\!\langle \delta \mid \delta^\sharp \rangle\!\rangle E] B$ be the RDD formula in Definition III.1. A relation $R \subseteq \mathbb{R}^{|\boldsymbol{x}|} \times \mathbb{R}^{|\boldsymbol{x}^\sharp|}$ is a *simulation* between δ and $\delta^\sharp$ if, for any pair $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp{}_0) \in R$ and for any $\boldsymbol{x}_1$ such that $\boldsymbol{x}_0 \xrightarrow{T_\delta} \boldsymbol{x}_1$, there exists $\boldsymbol{x}^\sharp{}_1$ such that $\boldsymbol{x}^\sharp{}_0 \xrightarrow{T_{\delta^\sharp}} \boldsymbol{x}^\sharp{}_1$ and $(\boldsymbol{x}_1, \boldsymbol{x}^\sharp{}_1) \in R$.

The following properties will be needed in our use of simulations. See Proposition III.7 later.

Definition III.6. We say a simulation R *supports* B under E if we have $R \cap [\![E]\!] \subseteq [\![B]\!]$.

We say an exit condition E is *essentially included* in a simulation R if we have

$$[\![E]\!] \cap \left\{ (\boldsymbol{x}_1, \boldsymbol{x}^\sharp{}_1) \;\middle|\; \exists (\boldsymbol{x}_0, \boldsymbol{x}^\sharp{}_0) \in R.\, \boldsymbol{x}_0 \xrightarrow{T_\delta} \boldsymbol{x}_1, \boldsymbol{x}^\sharp{}_0 \xrightarrow{T_{\delta^\sharp}} \boldsymbol{x}^\sharp{}_1 \right\} \subseteq R.$$

Here are some intuitions. The support condition guarantees that, as long as we stay in the simulation R, the postcondition B holds when we exit. The intention behind the essential inclusion condition is that $[\![E]\!]$ should be included in R, so that R speaks about every possible exit situation. However, the proper inclusion $[\![E]\!] \subseteq R$ is too much to ask, especially when we happen to choose a lax E. Therefore essential inclusion restricts the inclusion requirement to the reachable part.

Proposition III.7 (relational reasoning by simulations)**.** *Let $A \equiv [\langle\!\langle \delta \mid \delta^\sharp \rangle\!\rangle E] B$ be the RDD formula in Definition III.1, and let R be a simulation relation between δ and $\delta^\sharp$.*

Assume that R supports the postcondition B under the exit condition E, and that E is essentially included in R (Definition III.6). Then we have

$$R \subseteq [\![A]\!] \quad .$$

Proof. Let $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp{}_0) \in R$ be arbitrary and let $\boldsymbol{x}_1$ be such that $\boldsymbol{x}_0 \xrightarrow{T_\delta} \boldsymbol{x}_1$. Let us now consider an arbitrary $\boldsymbol{x}^\sharp{}_1$ such that $\boldsymbol{x}^\sharp{}_0 \xrightarrow{T_{\delta^\sharp}} \boldsymbol{x}^\sharp{}_1$ and $(\boldsymbol{x}_1, \boldsymbol{x}^\sharp{}_1) \in [\![E]\!]$. We can assume such $\boldsymbol{x}^\sharp{}_1$ to exist as otherwise, $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp{}_0)\, [\![\delta; \delta^\sharp; ?E]\!] = \varnothing$. Since E is essentially included in R, we have also $(\boldsymbol{x}_1, \boldsymbol{x}^\sharp{}_1) \in R$. Finally, since R supports B under E, $(\boldsymbol{x}_1, \boldsymbol{x}^\sharp{}_1) \in R \cap [\![E]\!] \subseteq [\![B]\!]$. □

Example III.8 (collision speed)**.** Let us illustrate how we can prove RDD formulas using simulations by solving our leading example on collision speed (Example I.1). Recall the formal representation $A \equiv [\langle\!\langle \delta \mid \delta^\sharp \rangle\!\rangle E] B$ of the problem from Example III.3; our goal is to prove that the RDD formula A is true under the initial condition $\Gamma \equiv (x = x^\sharp = 0 \land v = v^\sharp = 0)$.

By Proposition III.7, it is enough to construct a simulation $R \subseteq \mathbb{R}^2 \times \mathbb{R}^2$ that supports the postcondition $B \equiv (v \leq v^\sharp)$, such that the exit condition $E \equiv (x = x^\sharp = 1)$ is essentially included in R. We moreover need that R contains the initial condition Γ, i.e., $\big((0,0), (0,0) \big) \in R$.

We use the closed-form solutions of the dynamics at hand to construct R, namely

$$\psi : \mathbb{R}_{\geq 0} \to \mathbb{R}^2, \qquad \psi(t) = \big(t^2/2, t \big); \text{ and}$$
$$\psi^\sharp : \mathbb{R}_{\geq 0} \to \mathbb{R}^2, \qquad \psi^\sharp(t^\sharp) = \big((t^\sharp)^2, 2t^\sharp \big).$$

As explained earlier, we want to compare the speeds of those two dynamics at the same *position*, not at the same *time*. Thus we want to define R by

$$R = \big\{ \big(\psi(t), \psi^\sharp(t^\sharp) \big) \mid \pi_1\big(\psi(t)\big) = \pi_1\big(\psi^\sharp(t^\sharp)\big) \big\},$$

where π_1 is the first projection (extracting positions). Solving $\frac{t^2}{2} = \pi_1(\psi(t)) = \pi_1(\psi^\sharp(t^\sharp)) = (t^\sharp)^2$, we obtain

$$R = \left\{ \big((t^2/2, t), (t^2/2, \sqrt{2}t) \big) \;\middle|\; t \in \mathbb{R}_{\geq 0} \right\}$$

It is then easy to check that R is indeed a simulation that satisfies the assumptions of Proposition III.7.

C. Simulation Based Proof Rule

Using Proposition III.7, we formulate a syntactic dL proof rule which allows proving an RDD formula using a simulation.

Definition III.9 (simulation proof rule (SIM))**.** The *simulation rule* is given as follows, using four premises.

$$\frac{\begin{array}{c} R \vdash [\delta]\langle\delta^\sharp\rangle R \qquad R, E \vdash B \qquad R \vdash [\langle\!\langle \delta \mid \delta^\sharp \rangle\!\rangle E] R \\ \Gamma \vdash [?Q(\boldsymbol{x}) \land Q^\sharp(\boldsymbol{x}^\sharp)] R \end{array}}{\Gamma \vdash [\langle\!\langle \delta \mid \delta^\sharp \rangle\!\rangle E] B} \text{ SIM}$$

Note that, to be precise, R here is a dL formula whose interpretation is indented to be a simulation.

The first premise expresses the simulation condition on R; the second is for the support of B under E; the third is the essential inclusion of E in R; and the last demands that R holds in the initial states.

We observe that, in the (SIM) rule, the conditions required of R in Proposition III.7 are succinctly expressed in the dL syntax. In particular, the deft expression $R \vdash [\delta]\langle\delta^\sharp\rangle R$ of the simulation condition by the combination of two modalities demonstrates the versatility of dL.

Theorem III.10. *The simulation proof rule (SIM) in Definition III.9 is sound.* □

Remark III.11. How useful the rule (SIM) is depends on the expressivity of the logic dL, that is, how many simulations R we can express in the logic. Given the close tie between the theory of (bi)simulations and that of fixed points (such as the Knaster–Tarski theorem), we expect it be useful to be able to express fixed points in dL. A possible example is an extension of the dL syntax that has the least and greatest fixed operators: a user writes a function Φ and the expression $\mu x.\,\Phi(x)$ designates its fixed point. Such extension of dL is future work.

IV. Time Stretch Functions and Reasoning Thereby

Although theoretically simple and foundational, the simulation rule (SIM) introduced in Definition III.9 is largely generic making it impractical for many properties. In particular, the proof of the first premise, showing that R is in fact a simulation, is likely to require explicit solution of (some) of the differential equations.

To avoid the need for explicit solutions, and following the intuitions that we developed in §III, we introduce another construction which, although only usable under constrained circumstances, provides a nice way to compare two different dynamics. If simulations compare states, this new construction features a function that relates the elapsed times of the two dynamics. It is called a *time stretch function*. The main goal of this section is to describe this semantical tool in order to design a proof rule in dL allowing us to prove relational differential dynamics formulas. After some semantical preparations, describing some fundamental properties of time stretching – much as synchronizing two dynamics according some exit conditions – we prove the soundness of our rule.

We will illustrate the usefulness of this rule in §VI, by solving examples inspired by our collaboration with the industry.

A. Time Stretch Functions

A time stretch function is just a monotonic rescaling of time:

Definition IV.1 (time stretch function)**.** Let $T \in \mathbb{R}_{\geq 0}$. A function $k\colon [0,T] \to \mathbb{R}_{\geq 0}$ is called a *time stretch function* if k is C^1, $k(0) = 0$ and $\dot{k}(t) > 0$ for each $t \in [0,T]$.

Remark IV.2. In particular, k is strictly increasing. Then k is a bijection from $[0,T]$ to $[0,k(T)]$. We denote the inverse from $[0,k(T)]$ to $[0,T]$ by k^{-1}.

B. Stretched Dynamics

Our main interest of a time stretch function is to relate states of two different dynamics, much as those of a relational differential dynamics formula $A \equiv [\langle\!\langle \delta \mid \delta^\sharp \rangle\!\rangle\, E]\, B$. More precisely, a time stretch function will relate a time of one dynamics to the a time of the other. This will allow us to combine two dynamics (with two different time scales) into one (with a unique time scale). In this section, we look at how to rescale a dynamic by using a time stretch function and that the dynamics produced by this process visits the same states as the initial dynamics, but at a different time.

Definition IV.3 (stretched dynamics $\delta^\sharp{}_k$)**.** Let A be the RDD formula in Definition III.1, and let k be a time stretch function (Definition IV.1). From a dynamics $\delta^\sharp \equiv (\dot{\boldsymbol{x}}^\sharp = \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp)\ \&\ Q^\sharp(\boldsymbol{x}^\sharp))$, we obtain a new dynamics

$$\delta^\sharp{}_k \equiv \Big(\big(\dot{\boldsymbol{x}}^\sharp = \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp)\cdot \dot{k}(t)\big)\ \&\ Q^\sharp(\boldsymbol{x}^\sharp)\Big). \tag{6}$$

The dynamics $\delta^\sharp{}_k$ is called the dynamics *stretched by* k.

Definition IV.3 of the stretched dynamics relies on the time t. This is not a real problem, as we will not use this stretched dynamics inside dL, but only to help our proofs, the continuity of the derivative of k being enough to apply Picard-Lindelöf's theorem. We can, however, use it inside dL by assuming t to be a part of the $\boldsymbol{x}$ vector and requiring k to be smooth enough, e.g., C^2.

Provided the solution ψ_k of the stretched system exists at the relevant interval, it retains the results of the solutions of the original system:

Lemma IV.4. *Let A be the RDD formula in Definition III.1, k be a time stretch function and $\boldsymbol{x}^\sharp{}_0 \in \mathbb{R}^{|\boldsymbol{x}^\sharp|}$. Let $T \in \mathbb{R}_{\geq 0}$ be arbitrary such that $\delta^\sharp$ has a solution $\psi^\sharp$ on $[0,k(T)]$. Then the function that maps t to $\psi^\sharp(k(t))$ is a solution of the stretched dynamics δ_k.*

Reciprocally, if the stretched dynamics δ_k has a solution ψ_k on $[0,T]$, then the function that maps $t^\sharp$ to $\psi_k(k^{-1}(t^\sharp))$ is a solution of $\delta^\sharp$ on $[0,k(T)]$, with k^{-1} being the inverse of k (see Remark IV.2).

Proof. Both directions boils down in deriving the candidate solutions:

- For the first direction, $\frac{d\psi^\sharp(k(t))}{dt}(s) = \dot{k}(s)\cdot\dot{\psi}^\sharp(k(s)) = \dot{k}(s)\cdot\boldsymbol{f}^\sharp(\psi^\sharp(k(s)))$.
- For the other direction, $\frac{d\psi_k(k^{-1}(t))}{dt}(s) = \dot{k^{-1}}(s)\cdot\dot{\psi}_k(k^{-1}(s)) = \frac{1}{\dot{k}(k^{-1}(s))}\cdot\dot{k}(k^{-1}(s))\cdot\boldsymbol{f}^\sharp(\psi_k(k^{-1}(s))) = \boldsymbol{f}^\sharp(\psi_k(k^{-1}(s)))$. □

C. Canonical Time Stretching, Semantically

For the moment, the time stretch function arbitrarily relates states of two dynamics, keeping the same solutions. Now, we would like to construct particular time stretch functions so that

the two dynamics synchronize their exit time. Given an RDD formula A as in Definition III.1, we restrict the exit condition to a single equality $E \equiv (g(\boldsymbol{x}) = g^\sharp(\boldsymbol{x}^\sharp))$ with g and $g^\sharp$ strictly monotonic in some sense. We show it is possible to obtain a time stretch function k from $\delta, \delta^\sharp, g$ and $g^\sharp$.

Definition IV.5 (monotonic exit condition)**.** Let A be an RDD formula as in Definition III.1. Assume that the exit condition $E \equiv (g(\boldsymbol{x}) = g^\sharp(\boldsymbol{x}^\sharp))$ is an equality, with g and $g^\sharp$ smooth enough (e.g., C^2).

Given $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp_0)$, let us define the following two functions:

$$g_f\colon [0, T_{\boldsymbol{x}_0}) \longrightarrow \mathbb{R}, \qquad g_f(t) = g(\psi(\boldsymbol{x}_0, t));$$
$$g^\sharp_{f^\sharp}\colon [0, T_{\boldsymbol{x}^\sharp_0}) \longrightarrow \mathbb{R}, \qquad g^\sharp_{f^\sharp}(t^\sharp) = g^\sharp(\psi^\sharp(\boldsymbol{x}^\sharp_0, t^\sharp)).$$

We say that the exit condition E in A is *monotonic at* $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp_0)$ if the derivatives of g_f and $g^\sharp_{f^\sharp}$ are both strictly positive or both strictly negative.

Definition IV.6 (canonical time stretch function)**.** Let $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp{}_0)$ be some initial states, and A be an RDD formula as in Definition III.1. Assume that the exit condition $E \equiv (g(\boldsymbol{x}) = g^\sharp(\boldsymbol{x}^\sharp))$ is monotonic at $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp_0)$ (Definition IV.5) and that $g(\boldsymbol{x}_0) = g^\sharp(\boldsymbol{x}^\sharp_0)$. Assume given $t, t^\sharp \in \mathbb{R}_{\geq 0}$ with $g(\psi(\boldsymbol{x}_0, t)) = g^\sharp(\psi^\sharp(\boldsymbol{x}^\sharp_0, t^\sharp))$. Then we define the function:

$$k\colon [0, t] \longrightarrow [0, t^\sharp], \quad k(s) = (g^\sharp_{f^\sharp})^{-1} \circ g_f(s)$$

called the *canonical time stretch function*.

Proposition IV.7. *In the setting of Definition IV.6, k is a time stretch function with $k(t) = t^\sharp$.*

Proof. First, let us prove that k is well-defined. Since $g^\sharp_{f^\sharp}$ is strictly monotonic on $[0, t^\sharp]$, it has an inverse $h = (g^\sharp_{f^\sharp})^{-1}$, defined from $g^\sharp_{f^\sharp}([0, t^\sharp])$ to $[0, t^\sharp]$. By assumption, $g^\sharp_{f^\sharp}(0) = g_f(0)$ and $g^\sharp_{f^\sharp}(t^\sharp) = g_f(t)$, so that h is defined from $g_f([0, t])$ to $[0, t^\sharp]$. Then k is defined from $[0, t]$ to $[0, t^\sharp]$. Furthermore, since g and $g^\sharp$ are continuously differentiable and $\boldsymbol{f}$ and $\boldsymbol{f}^\sharp$ are continuous, k is C^1. Since $g(\boldsymbol{x}_0) = g^\sharp(\boldsymbol{x}^\sharp_0)$, then $k(0) = 0$. Finally,

$$\begin{aligned} \dot{k}(s) &= \dot{g_f}(s)\dot{h}(g_f(s)) && \text{derivation of } h \circ g_f \\ &= \frac{\dot{g_f}(s)}{g^\sharp_{f^\sharp}(k(s))} && \text{derivation of } h = (g_f)^{-1} \end{aligned}$$

which is strictly positive by monotonicity of the exit condition. □

D. Towards a Syntactic Representation

From now on, we aim at a syntactic representation of the reasoning with time stretch functions inside dL. We answer the following question: Is there any suitable sound rule that we can use to prove RDD formulas, using time stretch functions?

The main idea is to combine the two dynamics δ and $\delta^\sharp$ using the canonical time stretch function k, in order to get a unique dynamics, and do proofs in dL on this new dynamics instead. A natural way to combine δ and $\delta^\sharp$ would be to study the stretched dynamics by k (Definition IV.3):

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \dot{\boldsymbol{x}^\sharp} = f^\sharp(\boldsymbol{x}^\sharp) \cdot \dot{k}(t) \,\&\, Q(\boldsymbol{x}) \wedge Q^\sharp(\boldsymbol{x}^\sharp)$$

When we expand the definition of k we obtain the following:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \dot{\boldsymbol{x}^\sharp} = \frac{\mathcal{L}_f g(\psi(\boldsymbol{x}_0, t))}{\mathcal{L}_{f^\sharp} g^\sharp(\psi^\sharp(\boldsymbol{x}^\sharp_0, k(t)))} \cdot \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp) \,\&\, Q(\boldsymbol{x}) \wedge Q^\sharp(\boldsymbol{x}^\sharp)$$

The problem is that this dynamics is not practical as it depends on the initial conditions and the actual solutions of the dynamics. However, using Lemma IV.4, we can replace the solution $\psi^\sharp(\boldsymbol{x}^\sharp_0, k(t))$ by $\boldsymbol{x}^\sharp$ and, by construction, we can replace the solution $\psi(\boldsymbol{x}_0, t)$ by $\boldsymbol{x}$, which leads to:

Definition IV.8. In the setting of Definition IV.6, we define:

$$\delta_A \equiv \left(\left(\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \dot{\boldsymbol{x}^\sharp} = \frac{\mathcal{L}_f g(\boldsymbol{x})}{\mathcal{L}_{f^\sharp} g^\sharp(\boldsymbol{x}^\sharp)} \cdot \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp) \right) \& Q(\boldsymbol{x}) \wedge Q^\sharp(\boldsymbol{x}^\sharp) \right)$$

called the *synchronized dynamics*.

Remark IV.9. Observe that, since g and $g^\sharp$ are C^2, the function that maps $\boldsymbol{x}, \boldsymbol{x}^\sharp$ to $\frac{\mathcal{L}_f g(\boldsymbol{x})}{\mathcal{L}_{f^\sharp} g^\sharp(\boldsymbol{x}^\sharp)} \cdot \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp)$ is C^1, as required in Remark II.3.

The following will be the main argument of the proof of soundness of our dL rule, stating that, much as the stretched dynamics, the combined dynamics preserves the visited states.

Proposition IV.10. *Assume given $\boldsymbol{x}_0 \in \mathbb{R}^{|\boldsymbol{x}|}$ and $\boldsymbol{x}^\sharp_0 \in \mathbb{R}^{\boldsymbol{x}^\sharp}$ such that $g(\boldsymbol{x}_0) = g^\sharp(\boldsymbol{x}^\sharp_0)$, and the exit condition is monotonic.*

Assume $t, t^\sharp \in \mathbb{R}_{\geq 0}$ with $g(\psi(\boldsymbol{x}_0, t)) = g^\sharp(\psi^\sharp(\boldsymbol{x}^\sharp_0, t^\sharp))$. Then the function which maps $s \in [0, t]$ to

$$(\psi(\boldsymbol{x}_0, s), \psi^\sharp(\boldsymbol{x}^\sharp_0, k(s)))$$

is a solution of the synchronized dynamics on $[0, t]$, with the initial condition $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp_0)$.

Reciprocally, assume that $(\Phi, \Phi^\sharp)$ is a solution of δ_A on $[0, T]$, with the initial condition $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp_0)$, then Φ, $\Phi^\sharp$ allow the presentation:

$$\Phi = \psi(\boldsymbol{x}_0, \_) \qquad \Phi^\sharp = \psi^\sharp(\boldsymbol{x}^\sharp_0, \bar{k}(\_))$$

where $\bar{k}$ is some time stretch function. Furthermore, for every $s \in [0, T]$, $g(\Phi(s)) = g^\sharp(\Phi^\sharp(s))$.

Proof. For the first direction, the only thing to prove is that $\theta(s) = \psi^\sharp(\boldsymbol{x}^\sharp_0, k(s))$ is solution of $\dot{\boldsymbol{x}^\sharp} = \frac{\mathcal{L}_f g(\boldsymbol{x})}{\mathcal{L}_{f^\sharp} g^\sharp(\boldsymbol{x}^\sharp)} \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp)$:

$$\begin{aligned} \dot{\theta}(s) &= \dot{k}(s) f^\sharp(\psi^\sharp(\boldsymbol{x}^\sharp_0, k(s))) \\ &= \frac{\dot{g_f}(s)}{g^\sharp_{f^\sharp}(k(s))} f^\sharp(\psi^\sharp(\boldsymbol{x}^\sharp_0, k(s))) \\ &= \frac{\mathcal{L}_f g(\psi(\boldsymbol{x}_0, s))}{\mathcal{L}_{f^\sharp} g^\sharp(\psi^\sharp(\boldsymbol{x}^\sharp_0, k(s)))} f^\sharp(\psi^\sharp(\boldsymbol{x}^\sharp_0, k(s))) \end{aligned}$$

For the other direction, let us first observe that $\Phi = \psi(\boldsymbol{x}_0, \_)$ is obvious by unicity of the solution. Now, let us construct $\bar{k}$. It is defined as the unique primitive of the function $h(s) = \frac{\mathcal{L}_f g(\Phi(s))}{\mathcal{L}_{f^\sharp} g^\sharp(\Phi^\sharp(s))}$ such that $\bar{k}(0) = 0$. To prove that $\bar{k}$ is a time stretch function, it only remains to prove that it is strictly increasing, that is, is derivative h is strictly positive. $h(0) = \frac{\mathcal{L}_f g(\boldsymbol{x}_0)}{\mathcal{L}_{f^\sharp} g^\sharp(\boldsymbol{x}^\sharp_0)}$, which is strictly positive by monotonicity of the exit condition. Since $\Phi = \psi(\boldsymbol{x}_0, \_)$, and again by monotonicity of the exit solution, we know that h never takes the values 0. Finally, since h is continuous, it is always strictly

positive. So then, we can look at the stretched dynamics $\delta^\sharp_k$. By construction, $\Phi^\sharp$ is a solution of $\delta^\sharp_k$ on $[0,t]$, so we can conclude using Lemma IV.4. Finally, the last requirement is proved as follows. Define the following continuously derivable functions on $[0,T]$:

$$r(s) = g(\Phi(s)), \quad r^\sharp(s) = g^\sharp(\Phi^\sharp(s)).$$

Those functions are equal because: $r(0) = g(\boldsymbol{x}_0) = g^\sharp(\boldsymbol{x}^\sharp_0) = r^\sharp(0)$, and $\dot{r}(s) = \mathcal{L}_f g(\Phi(s)) = \dot{r}^\sharp(s)$. □

E. Time Stretch Function Based Proof Rule

Now, we have all the semantical ingredients to describe our dL rule and prove its soundness. We call it the *time stretch rule* (TS); it utilizes the synchronized dynamics to combine the two differential dynamics. Once the dynamics are synchronized, the time stretch rule may be easily followed by other dL rules previously developed for non-relational reasoning. The rule derives an equivalence, much as the (DI) rule as presented in [14].

Definition IV.11 (time stretch proof rule)**.** Let $A \equiv [\langle\!\langle \delta \mid \delta^\sharp \rangle\!\rangle E]\, B$ be the RDD formula in Definition III.1; assume that the exit condition $E \equiv (g(\boldsymbol{x}) = g^\sharp(\boldsymbol{x}^\sharp))$ is monotonic.

The time stretch proof rule for A is defined as follows.

$$\frac{\Gamma \vdash [?Q(\boldsymbol{x}) \wedge Q^\sharp(\boldsymbol{x}^\sharp)]\, E \quad \Gamma \vdash [\delta; \delta^\sharp]\, \frac{\mathcal{L}_f g(\boldsymbol{x})}{\mathcal{L}_{f^\sharp} g^\sharp(\boldsymbol{x}^\sharp)} > 0}{\Gamma \vdash [\langle\!\langle \delta \mid \delta^\sharp \rangle\!\rangle E]\, B \iff [\delta_A]\, B}\ \text{TS}$$

Note that the time stretch rule (TS) uses the standard dL syntax and is therefore readily integrable with the rest of the dL proof rules and into the theorem prover KeYmaera X.

Theorem IV.12. *The time stretch proof rule (TS) as defined in Definition IV.11 is sound.*

Proof. Assume that the premises hold. Assume given $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp_0) \in [\![\Gamma]\!]$. Both directions of the (TS) rule are consequences of Proposition IV.10, since the premises imply its assumptions. □

V. Differential Inductive Invariants and Other Auxiliary Proof Rules

Here we introduce some auxiliary proof rules that we find crucial in many examples of relational reasoning in dL.

A. Differential Inductive Invariants

It turns out that the differential invariant rule (DI) as introduced in Definition II.5, although powerful, falls short of being able to capture some invariant properties. As we have encountered such properties in our case studies, we propose an inductive version of the differential invariant, which, although slightly less general, provides more power by including the postcondition in the evolution domain constraint.

Definition V.1. Fix $n \geq 1$. The n-th order differential inductive invariant (DII_n) proof rule is given as follows, assuming that $\boldsymbol{f}$ is C^{n-1} and g are C^n and δ is a shorthand for the differential dynamics $\delta \equiv \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q \wedge g(\boldsymbol{x}) \geq 0$:

$$\frac{\Gamma, Q \vdash g(\boldsymbol{x}) \geq 0 \quad \Gamma \vdash [\delta]\, D_n g(\boldsymbol{x})}{\Gamma \vdash [\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q]\, g(\boldsymbol{x}) \geq 0}\ \text{DII}_n$$

where

$$D_n g(\boldsymbol{x}) \equiv \bigvee_{p=0}^{n-1} \bigwedge_{k=1}^{p} \mathcal{L}_f^{(k)} g(\boldsymbol{x}) \geq 0 \wedge \mathcal{L}_f^{(p+1)} g(\boldsymbol{x}) > 0$$

and $\mathcal{L}_f^{(k)} g$ is the k-th Lie derivative of g with respect to $\boldsymbol{f}$.

In particular, (DII_1) will be of particular use:

$$\frac{\Gamma, Q \vdash g(\boldsymbol{x}) \geq 0 \quad \Gamma \vdash [\delta]\, \mathcal{L}_f g(\boldsymbol{x})}{\Gamma \vdash [\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q]\, g(\boldsymbol{x}) \geq 0}\ \text{DII}_1$$

Theorem V.2. *The differential inductive invariant rules (DII_n) in Definition V.1 are sound.*

B. Auxiliary Relational Rules

The following rules, while many of them are rather straightforward, play essential roles in many examples of relational reasoning. We may further identify two main categories of the auxiliary rules, first, rules that allow us to manipulate or introduce (monotonic) equality exit conditions; second, rules that facilitate relational reasoning in the standard syntax of dL.

1) Monotonic Condition Swap: First auxiliary rule we introduce is the *monotonic condition swap* rule (MSC). MSC swaps the exit condition with the postcondition, given the postcondition is given as inequality and some monotonicity criteria are satisfied. Swapping the exit condition and the postcondition is useful for obtaining exit condition with simpler derivatives (as illustrated in §VI-A) or to align the exit condition with initial condition (§VI-B).

The MSC rule comes in several variations based on the direction of the monotonicity of the functions involved in the exit condition and postcondition. The variation for all functions increasing is given in Figure 2.

Theorem V.3. *The* monotonic condition swap (MSC) *rule as given in Figure 2 is sound.*

2) Generating Exit Conditions: The other proof rules designed to manipulate equality exit conditions are the *relational differential cut* (RDC) and *exit condition propagation* (ECP) rules. RDC introduces new equality exit condition based on other tests in style of differential cut rule (DC). ECP allows to copy the exit condition to an earlier place in the dynamics under some monotonicity conditions.

Definition V.4 (relational differential cut rule)**.** The *relational differential cut* rule is given as follows, where $\delta \equiv (\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q(\boldsymbol{x}))$ and $\delta^\sharp \equiv (\dot{\boldsymbol{x}}^\sharp = \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp) \,\&\, Q^\sharp(\boldsymbol{x}^\sharp))$ are differential dynamics, α and β are shorthands for the hybrid programs:

$$\alpha \equiv \delta; \delta^\sharp; ?g(\boldsymbol{x}) = g^\sharp(\boldsymbol{x}^\sharp); ?P \qquad \beta \equiv \delta; \delta^\sharp; ?P$$

$$\frac{\Gamma \vdash [\alpha]\, \varphi \qquad \Gamma \vdash [\beta]\, g(\boldsymbol{x}) = g^\sharp(\boldsymbol{x}^\sharp)}{\Gamma \vdash [\delta; \delta^\sharp; ?P]\, \varphi}\ \text{RDC}$$

$$\frac{\begin{array}{c}\Gamma \vdash \left[\left\langle\!\left\langle \dfrac{\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q(\boldsymbol{x})}{\dot{\boldsymbol{x}}^\sharp = \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp) \,\&\, Q^\sharp(\boldsymbol{x}^\sharp)} \right\rangle\!\right\rangle h(\boldsymbol{x}) = h^\sharp(\boldsymbol{x}^\sharp)\right] g(\boldsymbol{x}) \geq g^\sharp(\boldsymbol{x}^\sharp) \\ \Gamma \vdash \left[?Q(\boldsymbol{x}) \wedge Q^\sharp(\boldsymbol{x}^\sharp)\right] h(\boldsymbol{x}) \leq h^\sharp(\boldsymbol{x}^\sharp) \qquad \Gamma \vdash \left[\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q(\boldsymbol{x})\right] \mathcal{L}_f g(\boldsymbol{x}) > 0 \\ \Gamma \vdash \left[\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q(\boldsymbol{x})\right] \mathcal{L}_f h(\boldsymbol{x}) \geq 0 \qquad \Gamma \vdash \left[\dot{\boldsymbol{x}}^\sharp = \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp) \,\&\, Q^\sharp(\boldsymbol{x}^\sharp)\right] \mathcal{L}_{f^\sharp} h^\sharp(\boldsymbol{x}^\sharp) \geq 0\end{array}}{\Gamma \vdash \left[\left\langle\!\left\langle \dfrac{\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q(\boldsymbol{x})}{\dot{\boldsymbol{x}}^\sharp = \boldsymbol{f}^\sharp(\boldsymbol{x}^\sharp) \,\&\, Q^\sharp(\boldsymbol{x}^\sharp)} \right\rangle\!\right\rangle g(\boldsymbol{x}) = g^\sharp(\boldsymbol{x}^\sharp)\right] h(\boldsymbol{x}) \leq h^\sharp(\boldsymbol{x}^\sharp)} \text{ MCS}$$

Fig. 2. Proof rule for monotonic condition swap for relations using monotonically increasing functions.

Theorem V.5. *The relational differential cut (RDC) rule is sound.*

Definition V.6 (exit condition propagation rule)**.** The *exit condition propagation* rule is given as follows, where $\delta \equiv (\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q(\boldsymbol{x}))$, $\delta^\sharp{}_1 \equiv (\dot{\boldsymbol{x}}^\sharp = \boldsymbol{f}^\sharp{}_1(\boldsymbol{x}^\sharp) \,\&\, Q^\sharp{}_1(\boldsymbol{x}^\sharp))$ and $\delta^\sharp{}_2 \equiv (\dot{\boldsymbol{x}}^\sharp = \boldsymbol{f}^\sharp{}_2(\boldsymbol{x}^\sharp) \,\&\, Q^\sharp{}_2(\boldsymbol{x}^\sharp))$ are differential dynamics, and α is a shorthand for the hybrid program:

$$\alpha \equiv \delta; \delta^\sharp{}_1; ?g(\boldsymbol{x}) = g^\sharp(\boldsymbol{x}^\sharp); ?P(\boldsymbol{x}^\sharp); \delta; \delta^\sharp{}_2; ?g(\boldsymbol{x}) = g^\sharp(\boldsymbol{x}^\sharp)$$

$$\frac{\begin{array}{c}\Gamma \vdash [\alpha]\varphi \\ \Gamma \vdash [\delta^\sharp{}_1] \mathcal{L}_{\boldsymbol{f}^\sharp{}_1} g^\sharp(\boldsymbol{x}^\sharp) \geq 0 \quad \Gamma \vdash [\delta^\sharp{}_1; \delta^\sharp{}_2] \mathcal{L}_{\boldsymbol{f}^\sharp{}_2} g^\sharp(\boldsymbol{x}^\sharp) \geq 0\end{array}}{\Gamma \vdash [\delta; \delta^\sharp{}_1; ?P(\boldsymbol{x}^\sharp); \delta^\sharp{}_2; ?g(\boldsymbol{x}) = g^\sharp(\boldsymbol{x}^\sharp)]\varphi} \text{ ECP}$$

Theorem V.7. *The exit condition propagation rule (*ECP*) is sound.*

C. Commutation and Absorption

Finally, we introduce rather straightforward, but convenient rules which allow us to comfortably manipulate relational differential dynamics formulas within the classical syntax of dL. The *sequential composition commutativity* rule (SCC) allows us to swap hybrid program acting on different variables, as they are independent of each other. The *merge identical dynamics* rule (MID) simply allows to merge two identical differential dynamics which directly follow each other.

Definition V.8 (sequential composition commutativity rule)**.** The *sequential composition commutativity* rule is given as follows in two versions (one for each modality), where $\alpha(\boldsymbol{x})$ denotes a hybrid program which acts on variables in $\boldsymbol{x}$ only.

$$\frac{\Gamma \vdash [\alpha^\sharp(\boldsymbol{x}^\sharp); \alpha(\boldsymbol{x})]\varphi}{\Gamma \vdash [\alpha(\boldsymbol{x}); \alpha^\sharp(\boldsymbol{x}^\sharp)]\varphi} \text{ SCC}_{[-]} \qquad \frac{\Gamma \vdash \langle\alpha^\sharp(\boldsymbol{x}^\sharp); \alpha(\boldsymbol{x})\rangle\varphi}{\Gamma \vdash \langle\alpha(\boldsymbol{x}); \alpha^\sharp(\boldsymbol{x}^\sharp)\rangle\varphi} \text{ SCC}_{\langle-\rangle}$$

Theorem V.9. *The sequential composition commutativity rules (SCC$_{[-]}$ and SCC$_{\langle-\rangle}$) are sound.*

Definition V.10 (merge identical dynamics rule)**.** The *merge identical dynamics* rule is given as follows in two versions (one for each modality), where $\delta \equiv (\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q(\boldsymbol{x}))$ is a differential dynamics (Definition II.1).

$$\frac{\Gamma \vdash [\delta]\varphi}{\Gamma \vdash [\delta; \delta]\varphi} \text{ MID}_{[-]} \qquad \frac{\Gamma \vdash \langle\delta\rangle\varphi}{\Gamma \vdash \langle\delta; \delta\rangle\varphi} \text{ MID}_{\langle-\rangle}$$

Theorem V.11. *The merge identical dynamics rules (MID$_{[-]}$ and MID$_{\langle-\rangle}$) are sound.*

VI. Case Studies

A. Collision Speed (Constant Acceleration)

In this section we apply the time stretch rule to an example from industry. For this example we consider two "identical" dynamics $\dot{x} = v, \dot{v} = a$ and $\dot{x}^\sharp = v^\sharp, \dot{v}^\sharp = a^\sharp$. Both dynamics represent a car with constant acceleration. The relational formula of interest specifies that if acceleration is larger in the second system, then the second car must necessarily travel faster after covering the same distance as the first car. Thus namely, upon collision with an equally distant obstacle. The formula φ_C is given formally as follows.

$$\begin{aligned} &0 = x = x^\sharp \wedge 0 < v = v^\sharp \wedge 0 < a < a^\sharp \\ &\Longrightarrow \left[\left\langle\!\left\langle \frac{\dot{x} = v, \dot{v} = a}{\dot{x}^\sharp = v^\sharp, \dot{v}^\sharp = a^\sharp} \right\rangle\!\right\rangle x = x^\sharp\right] v \leq v^\sharp \end{aligned} \tag{7}$$

One may remark that the dynamics considered in the formula φ_C are relatively simple and the explicit solution is therefore readily available. As the main strength of differential dynamic logic lies in the ability to prove properties of differential dynamics without explicit solution, however, we present a proof which, thanks to the time stretch rule, does not rely on any differential equation solutions.

Let us first mention that we treat the left side of the implication in (7) as the initial condition:

$$(0 = x = x^\sharp \wedge 0 < v = v^\sharp \wedge 0 < a < a^\sharp)$$

The proof, given in Figure 3, begins with a technical differential cut immediately followed by the time stretch rule. This is the most crucial step as it allows us to exploit the exit condition $x = x^\sharp$ without computing explicit solution, an option that does not readily present itself otherwise.

The result of the time stretch rule is the formula with only one dynamics, synchronized by $x = x^\sharp$. Standard dL rules may be thus applied to this formula, such as differential invariant (DI). (DI) rule is, however, too general for our case. To this end we employ the differential inductive invariant (DII$_1$) rule, which yields the crucial $v \leq v^\sharp$ in the evolution domain constraint at the price of strict inequality in the derivation.

The rest of the proof beyond the application of (DII$_1$) rule is mostly technical. The differential cut rule allows us to get rid of the fraction in our postcondition, thus paving way for another application of differential invariant, this time regular differential invariant (DI) rule is preferable. As one may remark, the second derivation yields the initial claim

$$0 = x = x^\sharp, 0 < v = v^\sharp, 0 < a < a^\sharp \vdash \left[\left\langle \frac{\dot{x} = v, \dot{v} = a}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp} \right\rangle\right] x = x^\sharp \Big] v \le v^\sharp \tag{8}$$

The target formula. Derived from (9) and (10) by the (DC)$\{v > 0\}$

$$0 = x, 0 < v, 0 < a \vdash [\dot{x} = v, \dot{v} = a]\, v > 0 \tag{9}$$

differential invariant (DI) and arithmetic fact

$$0 = x = x^\sharp, 0 < v = v^\sharp, 0 < a < a^\sharp \vdash \left[\left\langle \frac{\dot{x} = v, \dot{v} = a \;\&\; v > 0}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp} \right\rangle\right] x = x^\sharp \Big] v \le v^\sharp \tag{10}$$

(11), (12) and (15) by TS (Definition IV.11)

$$0 = x = x^\sharp, 0 < v = v^\sharp, 0 < a < a^\sharp \vdash x = x^\sharp \tag{11}$$

arithmetic fact

$$0 = x = x^\sharp, 0 < v = v^\sharp, 0 < a < a^\sharp \vdash \left[\left\langle \frac{\dot{x} = v, \dot{v} = a \;\&\; v > 0}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp} \right\rangle\right] \frac{v}{v^\sharp} > 0 \tag{12}$$

(13) and (14), by DW and DC$\{w > 0\}$

$$0 < a < a^\sharp, 0 < v^\sharp, v > 0 \vdash \left[\dot{v^\sharp} = a^\sharp\right] v^\sharp > 0 \tag{13}$$

(DI) and arithmetic fact

$$0 < a < a^\sharp, 0 < v^\sharp, v > 0 \vdash \left[\dot{v^\sharp} = a^\sharp \;\&\; v^\sharp > 0\right] \frac{v}{v^\sharp} > 0 \tag{14}$$

(DW) and arithmetic fact

$$0 = x = x^\sharp, 0 < v = v^\sharp, 0 < a < a^\sharp \vdash \left[\dot{x} = v, \dot{v} = a, \dot{x^\sharp} = \frac{v^\sharp \cdot v}{v^\sharp}, \dot{v^\sharp} = \frac{a^\sharp \cdot v}{v^\sharp} \;\&\; v > 0\right] v \le v^\sharp \tag{15}$$

(16) and (17), by (DII$_1$) (Definition V.1)

$$0 = x = x^\sharp, 0 < v = v^\sharp, 0 < a < a^\sharp \vdash v \le v^\sharp \tag{16}$$

arithmetic fact

$$0 = x = x^\sharp, 0 < v = v^\sharp, 0 < a < a^\sharp \vdash \left[\dot{x} = v, \dot{v} = a, \dot{x^\sharp} = \frac{v^\sharp \cdot v}{v^\sharp}, \dot{v^\sharp} = \frac{a^\sharp \cdot v}{v^\sharp} \;\&\; (v > 0 \land v \le v^\sharp)\right] a < \frac{a^\sharp \cdot v}{v^\sharp} \tag{17}$$

(18) and (19), by (DC)$\{a \cdot v^\sharp < a^\sharp \cdot v\}$

$$0 = x = x^\sharp, 0 < v = v^\sharp, 0 < a < a^\sharp \vdash \left[\dot{x} = v, \dot{v} = a, \dot{x^\sharp} = \frac{v^\sharp \cdot v}{v^\sharp}, \dot{v^\sharp} = \frac{a^\sharp \cdot v}{v^\sharp} \;\&\; (v > 0 \land v \le v^\sharp \land a \cdot v^\sharp < a^\sharp \cdot v)\right] a < \frac{a^\sharp \cdot v}{v^\sharp} \tag{18}$$

(DW) and arithmetic fact

$$0 = x = x^\sharp, 0 < v = v^\sharp, 0 < a < a^\sharp \vdash \left[\dot{x} = v, \dot{v} = a, \dot{x^\sharp} = \frac{v^\sharp \cdot v}{v^\sharp}, \dot{v^\sharp} = \frac{a^\sharp \cdot v}{v^\sharp} \;\&\; (v > 0 \land v \le v^\sharp)\right] a \cdot v^\sharp < a^\sharp \cdot v \tag{19}$$

(DI), (DW) and arithmetic fact

Fig. 3. The derivation of formula φ_C (7) in dL.

$v \le v^\sharp$, which is, however, already guaranteed to hold thanks to the differential inductive invariant.

In Appendix C, an alternative proof of φ_C is given, utilizing the monotonic condition swap (MSC) rule before applying the time stretch rule. Although the proof has more branches, it no longer requires the (DII$_1$) rule, resulting in an overall shorter proof. Moreover, using MSC allows us to relax the initial condition, namely, we may consider $a \le a^\sharp$ instead of the sharp inequality $a < a^\sharp$.

B. Collision Speed (Decaying Acceleration)

In this section we consider an extended version of the collision speed comparison from Section VI-A. In particular, instead of considering the cars to have constant acceleration, we consider the acceleration to "decay" when speed exceeds certain value. This decay represents the inability of the engine to supply enough power to overshadow the drag once the car reaches sufficiently high speed.

The acceleration decay at high speeds effectively translates to change in dynamics once sufficient speed is reached. Moreover, we consider the obstacle may be any distance from the starting point, including the possibility of the obstacle being too close for one or both of the cars to reach the speed at which the acceleration starts decaying (differential dynamics change) before collision. These two factors combined require us to consider the possibility of collision given any combination of dynamics. The formula φ_D, given in Figure 4, therefore grows quickly, however, retains the basic structure: Either collision happens or one of the cars reaches high enough speed to start losing acceleration, after which either the collision happens or the second car reaches high enough speed for acceleration to decay.

Again, we consider the left hand side of the implication in φ_D, to be the precondition and we further have $\Gamma \equiv (0 < a < a^\sharp \land 1 \le V)$ denote the static part of the precondition.

Due to the scale of the formula, we do not give a fully formal proof. Instead, we provide the proof strategy and highlight the crucial proof steps to give the reader enough intuition for the retrieval of the formal proof.

We start the proof by repeatedly expanding the nondeterministic choices $\alpha_0 \cup \alpha_1$ using rule ($[\cup]$) [14]. The expansion of nondeterministic choice yields a conjunction of five formulas representing the cases of collision before either v or $v^\sharp$ reaches V, collision after only v reaches V, collision after both v and $v^\sharp$ reach V but v reaches it first, etc.

One may remark that the two cases when both v and $v^\sharp$ reach V before collision only differ in the order in which v and $v^\sharp$ reach V. By treating the formulas with the auxiliary rules (SCC$_{[-]}$) (Definition V.8) and (MID$_{[-]}$) (Definition V.10), we show that the two order-dependent formulas are in fact syntactically equivalent. We give the four formulas after treatment with (SCC$_{[-]}$) and (MID$_{[-]}$) rules in Figure 5 in the forms we use to prove each individually.

Observe that the first formula (20) is identical to the constant acceleration case in Section VI-A limited by evolution domain constraints. Indeed, the formula (20) can be proven in with the techniques used in Section VI-A.

One may remark that as a result of the proof of the first formula (20), the second formula (21) is meaningless as we know $v = V \implies v^\sharp \ge V$. The formula still covers the case $v = v^\sharp$, however, and requires a proof, which turns out to be surprisingly challenging.

To prove that the postcondition $v \le v^\sharp$ holds after the second pair of dynamics, we first have to show it holds after the first pair of dynamics. To achieve this, we need to synchronize the first pair of dynamics, and therefore supply them with an equality exit condition. This is conducted by the exit condition propagation (ECP) rule from Definition V.6.

We may then introduce the condition $v \le v^\sharp$ into the evolution domain constraint of the synchronized dynamics by the differential cut rule (DC). The proof of $v \le v^\sharp$ after the synchronized dynamics corresponds to (15) in the proof of the constant acceleration case (Figure 3).

$$0 = x = x^\sharp \wedge 0 < v = v^\sharp \wedge 0 < a < a^\sharp \wedge 1 \leq V \Longrightarrow \left[\left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = a \,\&\, v \leq V}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp \,\&\, v^\sharp \leq V} \right\rangle\!\!\right\rangle (?x = x^\sharp) \cup \right.$$
$$\left(?v = V; \left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = \frac{a \cdot V}{v}}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp \,\&\, v^\sharp \leq V} \right\rangle\!\!\right\rangle (?x = x^\sharp) \cup \left(?v^\sharp = V; \left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = \frac{a \cdot V}{v}}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = \frac{a^\sharp \cdot V}{v^\sharp}} \right\rangle\!\!\right\rangle x = x^\sharp \right) \right) \cup$$
$$\left. \left(?v^\sharp = V; \left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = a \,\&\, v \leq V}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = \frac{a^\sharp \cdot V}{v^\sharp}} \right\rangle\!\!\right\rangle (?x = x^\sharp) \cup \left(?v = V; \left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = \frac{a \cdot V}{v}}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = \frac{a^\sharp \cdot V}{v^\sharp}} \right\rangle\!\!\right\rangle x = x^\sharp \right) \right) \right] v \leq v^\sharp$$

Fig. 4. Formula φ_D stating that a car with lower acceleration will collide with an obstacle at a set distance at lower speed than a car with higher acceleration. The acceleration is considered to decrease with increasing speed once sufficiently high speed it reached.

$$\left[\left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = a \,\&\, v \leq V}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp \,\&\, v^\sharp \leq V} \right\rangle\!\!\right\rangle x = x^\sharp \right] v \leq v^\sharp \tag{20}$$

$$\left[\left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = a \,\&\, v \leq V}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp \,\&\, v^\sharp \leq V} \right\rangle\!\!\right\rangle v = V \right] \left[\left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = \frac{a \cdot V}{v}}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp \,\&\, v^\sharp \leq V} \right\rangle\!\!\right\rangle x = x^\sharp \right] v \leq v^\sharp \tag{21}$$

$$\left[\left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = a \,\&\, v \leq V}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp \,\&\, v^\sharp \leq V} \right\rangle\!\!\right\rangle v^\sharp = V \right] \left[\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = \frac{a^\sharp \cdot V}{v^\sharp}; ?x = x^\sharp \right] v \leq v^\sharp \tag{22}$$

$$\left[\left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = a \,\&\, v \leq V}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp \,\&\, v^\sharp \leq V} \right\rangle\!\!\right\rangle v = V \wedge v^\sharp = V \right] \left[\left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = \frac{a \cdot V}{v}}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = \frac{a^\sharp \cdot V}{v^\sharp}} \right\rangle\!\!\right\rangle x = x^\sharp \right] v \leq v^\sharp \tag{23}$$

Fig. 5. Decomposition of ϕ_D into formulas specifying that the speed inequality holds for all combination of dynamics.

The following formula, obtained by extracting the crucial condition $v \leq v^\sharp$ by differential weakening (DW) rule, remains to be proven.

$$\Gamma, v \leq V, v^\sharp \leq V, v \leq v^\sharp, x = x^\sharp, v = V$$
$$\vdash \left[\left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = \frac{a \cdot V}{v}}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp \,\&\, v^\sharp \leq V} \right\rangle\!\!\right\rangle x = x^\sharp \right] v \leq v^\sharp$$

The proof again consists of synchronization by (TS) rule followed by the differential inductive invariant (DII_1) rule and a classical differential invariant (DI) rule.

The third formula (22) may be proven without any relational reasoning. We abstract the first pair of dynamics by the use of (DW) rule. The crucial part being to obtain $v \leq V$ in the precondition. The second dynamics on $\boldsymbol{x}^\sharp$ are subsequently also abstracted by weakening, after using differential cut (DC) rule to introduce $v^\sharp \geq V$ into the evolution domain constraint, thus capturing the monotonicity of $v^\sharp$. The constraints $v \leq V$ and $v^\sharp \geq V$ are enough to secure the coveted result $v \leq v^\sharp$.

Finally, the fourth formula (23) showcases a nice application of relational reasoning. Similarly to the second formula (21), we need to synchronize the first pair of dynamics. This time we introduce an exit condition $v = v^\sharp$ by the means of relational differential cut (RDC) rule. The synchronization of the first pair of Dynamics is again conducted with the purposes of introducing a precondition for the second pair of dynamics, in this case $x \geq x^\sharp$. Thus, the application of (TS), (DC) and (DW) rules leaves us with

$$\Gamma, v \leq V, v^\sharp \leq V, x \geq x^\sharp, v = v^\sharp, v = V, v^\sharp = V$$
$$\vdash \left[\left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = \frac{a \cdot V}{v}}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = \frac{a^\sharp \cdot V}{v^\sharp}} \right\rangle\!\!\right\rangle x = x^\sharp \right] v \leq v^\sharp \tag{24}$$

and, additionally, the proof of the side condition of (DC) rule:

$$\Gamma, v \leq V, v^\sharp \leq V, x \geq x^\sharp, v = v^\sharp, v = V, v^\sharp = V$$
$$\vdash \left[\left\langle\!\!\left\langle \frac{\dot{x} = v, \dot{v} = \frac{a \cdot V}{v}}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = \frac{a^\sharp \cdot V}{v^\sharp}} \right\rangle\!\!\right\rangle v = v^\sharp \right] x \geq x^\sharp$$

which is proved by applying (DI) rule twice.

At this stage, (TS) rule is not applicable to the obtained formula (24) since the exit condition $x = x^\sharp$ does not necessarily hold initially (we only have $x \geq x^\sharp$). Observe, however, that $v = v^\sharp$ is guaranteed. We therefore utilize the monotonic condition swap (MCS) rule, to switch the exit condition $x = x^\sharp$ with the postcondition $v \leq v^\sharp$.

$$0 = x = x^\sharp, 0 < v = v^\sharp, \Gamma$$
$$\vdash \left[\dot{x} = v, \dot{v} = a, \dot{x^\sharp} = \frac{v^\sharp \cdot a}{a^\sharp}, \dot{v^\sharp} = a \,\&\, \left(v \leq V \wedge v^\sharp \leq V \right) \right] x \geq x^\sharp$$

The proof is then finished by application of (TS) rule followed by a combination of (DC) and (DI) rules.

C. Collision Speed under Drag

Consider the differential equation $\dot{x} = v, \dot{v} = -v^p$. When $p = 2$, the dynamics is the usual *drag equation* in fluid dynamics (modulo constant multiplication). The case with $p = 1$ corresponds to the so-called *linear drag* or *viscous*

resistance; this modeling is known to be suited for objects moving slowly through a fluid that has no turbulence.

Our goal here is to derive the following sequent.

$$x = x^\sharp = 0 \land v = v^\sharp > 1 \vdash \left[\left\langle\left\langle \frac{\dot{x} = v, \dot{v} = -v}{\dot{x}^\sharp = v^\sharp, \dot{v}^\sharp = -(v^\sharp)^2} \right\rangle\right\rangle x = x^\sharp\right] \begin{array}{l} v^\sharp \le v \\ \lor v^\sharp \le 1 \end{array} \tag{25}$$

It models the following scenario: we roll two balls from the same height, and observe how hard they hit a wall after traveling the same distance. The two balls move in different fluids (i.e. with $p = 1$ and $p^\sharp = 2$). It is natural to expect that the latter ball undergoes stronger drag, hence $v^\sharp \le v$. The extra disjunct $v^\sharp \le 1$ is needed in the postcondition because, when v and $v^\sharp$ are small enough, the effect of the parameters p and $p^\sharp$ in the dynamics gets inverted. The sequent (25) is derived in dL as shown in Fig. 6. In (40) we introduce the *differential conditional cut* rule (DCC). Its soundness is straightforward: due to its second promise, once a trajectory leaves C, it never satisfies C henceforth, therefore $C \Rightarrow \phi$ is trivially true. We expect that the (DCC) rule is derivable from the calculus in [14], but we have not yet managed to do so.

We note that, while the differential equations here have closed-form solutions $v = v(0) \cdot e^{-t}$ and $v^\sharp = \frac{1}{t^\sharp + v^\sharp(0)^{-1}}$, our proof in Fig. 6 does not use them at all. Our reasoning is purely local, firstly synchronizing the two dynamics with respect to the exit condition $x = x^\sharp$, and then reasoning with differential invariants (the (DI) and (DII$_1$) rules).

VII. Conclusions and Future Work

In this paper, we extended the well-developed framework of differential dynamic logic dL [14]–[16] to *relational reasoning*. Towards the goal of obtaining modular and local proof principles (much like the (DI) rule in dL), we introduced the theory of *simulation* and *time stretching* (Sections III–IV), which allows us to successfully relate different time points of two dynamics. The resulting proof rules ((SIM) and (TS)) are concise, and our case studies (inspired by industry collaboration) demonstrate their usefulness. We also introduced some additional proof rules (§V), among which is the *differential inductive invariant* (DII) rule.

One imminent direction of future work is the integration of the new rules in the theorem prover KeYmaera X [12]. We are already working on it and our preliminary observations have been promising.

We are eager to further pursue practical application of relational reasoning about hybrid systems—especially in *test-case reduction*—as we laid out in §I-E. This will require integration with testing methods, whose example is *falsification* (i.e. search-based by optimization) methods that recently attract attention. See e.g. [2], [5], [21]. Towards industry application, another requirement is accommodation of Simulink models as input. The work [11] suggests this is feasible.

On the theoretical side, our future work is to extend our framework so that we can relate different *discrete* control structures. We believe the existing work on relational Hoare logic such as [1], [3], [4] will be useful.

$$x = x^\sharp = 0 \land v = v^\sharp > 1 \vdash \left[\left\langle\left\langle \frac{\dot{x} = v, \dot{v} = -v}{\dot{x}^\sharp = v^\sharp, \dot{v}^\sharp = -(v^\sharp)^2} \right\rangle\right\rangle x = x^\sharp\right] \begin{array}{l} v^\sharp \le v \\ \lor v^\sharp \le 1 \end{array} \tag{26}$$

The ultimate goal. Derived from (27), by the (?) rule in [14]

$$x = x^\sharp = 0 \land v = v^\sharp > 1 \vdash \left[\left\langle\left\langle \frac{\dot{x} = v, \dot{v} = -v}{\dot{x}^\sharp = v^\sharp, \dot{v}^\sharp = -(v^\sharp)^2} \right\rangle\right\rangle x = x^\sharp\right][?v^\sharp > 1] \quad v^\sharp \le v \tag{27}$$

(28), (29), (35), the (TS) rule

$$x = x^\sharp = 0 \land v = v^\sharp > 1 \vdash x = x^\sharp \tag{28}$$

arithmetic fact

$$x = x^\sharp = 0 \land v = v^\sharp > 1 \vdash \left[\begin{array}{l} \dot{x} = v, \dot{v} = -v; \\ \dot{x}^\sharp = v^\sharp, \dot{v}^\sharp = -(v^\sharp)^2 \end{array}\right] \frac{v}{v^\sharp} > 0 \tag{29}$$

(30), (33)

$$v > 0 \vdash [\dot{v} = -v] \quad v > 0 \tag{30}$$

(31), the Darboux inequality ($\mathrm{dbx}_>$) rule shown in (32)

$$\vdash -v \ge -1 \cdot v \tag{31}$$

arithmetic fact

$$\frac{Q \vdash \mathcal{L}_{\boldsymbol{f}} h \ge gh}{h > 0 \vdash [\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q]\, h > 0}\ \mathrm{dbx}_> \text{ (g is a polynomial)} \tag{32}$$

the Darboux inequality rule from [16]

$$v^\sharp > 0 \vdash \left[\dot{v^\sharp} = -(v^\sharp)^2\right] \quad v^\sharp > 0 \tag{33}$$

(34), the ($\mathrm{dbx}_>$) rule shown in (32)

$$\vdash -(v^\sharp)^2 \ge -v^\sharp \cdot v^\sharp \tag{34}$$

arithmetic fact

$$x = x^\sharp = 0 \land v = v^\sharp > 1 \vdash \left[\dot{x} = v, \dot{v} = -v, \dot{x^\sharp} = v^\sharp \cdot \tfrac{v}{v^\sharp}, \dot{v^\sharp} = -(v^\sharp)^2 \cdot \tfrac{v}{v^\sharp}\right][?v^\sharp > 1] \quad v^\sharp \le v \tag{35}$$

(36), the (?) rule in [14]

$$\begin{array}{l} x = x^\sharp = 0 \land \\ v = v^\sharp > 1 \end{array} \vdash \left[\begin{array}{l} \dot{x} = v, \dot{v} = -v, \\ \dot{x}^\sharp = v^\sharp \cdot \frac{v}{v^\sharp}, \dot{v}^\sharp = -(v^\sharp)^2 \cdot \frac{v}{v^\sharp} \end{array}\right] v^\sharp > 1 \Rightarrow v^\sharp \le v \tag{36}$$

(37), the (DC) rule. The other premise of the rule is discharged much like (30)

$$\begin{array}{l} x = x^\sharp = 0 \land \\ v = v^\sharp > 1 \end{array} \vdash \left[\begin{array}{l} \dot{x} = v, \dot{v} = -v, \\ \dot{x}^\sharp = v^\sharp \cdot \frac{v}{v^\sharp}, \dot{v}^\sharp = -(v^\sharp)^2 \cdot \frac{v}{v^\sharp} \\ \&(v > 0 \land v^\sharp > 0) \end{array}\right] v^\sharp > 1 \Rightarrow v^\sharp \le v \tag{37}$$

(38), (39), and the differential conditional cut rule (DCC) presented in (40)

$$\begin{array}{l} x = x^\sharp = 0 \land \\ v = v^\sharp > 1 \end{array} \vdash \left[\begin{array}{l} \dot{x} = v, \dot{v} = -v, \\ \dot{x}^\sharp = v^\sharp \cdot \frac{v}{v^\sharp}, \dot{v}^\sharp = -(v^\sharp)^2 \cdot \frac{v}{v^\sharp} \\ \&\,(v > 0 \land v^\sharp > 1) \end{array}\right] v^\sharp \le v \tag{38}$$

(41), the (;) rule from [14]

$$v > 0 \land v^\sharp > 0 \land v^\sharp \le 1 \vdash \left[\begin{array}{l} \dot{x} = v, \dot{v} = -v, \\ \dot{x}^\sharp = v^\sharp \cdot \frac{v}{v^\sharp}, \dot{v}^\sharp = -(v^\sharp)^2 \cdot \frac{v}{v^\sharp} \\ \&\,(v > 0 \land v^\sharp > 0) \end{array}\right] v^\sharp \le 1 \tag{39}$$

the (DI) rule and an arithmetic fact

$$\frac{\Gamma \vdash [\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, (Q \land C)]\, \varphi \quad Q, \neg C \vdash [\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q]\, \neg C}{\Gamma \vdash [\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \,\&\, Q]\, (C \Rightarrow \varphi)}\ \mathrm{DCC} \tag{40}$$

The *differential conditional cut* rule

$$\begin{array}{l} x = x^\sharp = 0 \land \\ v = v^\sharp > 1 \end{array} \vdash \left[\begin{array}{l} \dot{x} = v, \dot{v} = -v, \\ \dot{x}^\sharp = v^\sharp \cdot \frac{v}{v^\sharp}, \dot{v}^\sharp = -(v^\sharp)^2 \cdot \frac{v}{v^\sharp} \\ \&\; v > 0 \land v^\sharp > 1 \end{array}\right] v^\sharp \le v \tag{41}$$

(42), the (DII$_1$) rule. The first premise of the rule is immediately discharged

$$\begin{array}{l} x = x^\sharp = 0 \land \\ v = v^\sharp > 1 \end{array} \vdash \left[\begin{array}{l} \dot{x} = v, \dot{v} = -v, \\ \dot{x}^\sharp = v^\sharp \cdot \frac{v}{v^\sharp}, \dot{v}^\sharp = -(v^\sharp)^2 \cdot \frac{v}{v^\sharp} \\ \&\; v > 0 \land v^\sharp > 1 \land v^\sharp \le v \end{array}\right] \begin{array}{r} -(v^\sharp)^2 \cdot \frac{v}{v^\sharp} \\ < -v \end{array} \tag{42}$$

(43), the (DW) rule

$$v > 0 \land v^\sharp > 1 \land v^\sharp \le v \vdash -(v^\sharp)^2 \cdot \frac{v}{v^\sharp} < -v \tag{43}$$

arithmetic fact

Fig. 6. The drag example: a derivation of the sequent (25) in dL

References

[1] Alejandro Aguirre, Gilles Barthe, Marco Gaboardi, Deepak Garg, and Pierre-Yves Strub. A relational logic for higher-order programs. *PACMPL*, 1(ICFP):21:1–21:29, 2017.

[2] Yashwanth Annpureddy, Che Liu, Georgios E. Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In Parosh Aziz Abdulla and K. Rustan M. Leino, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 17th International Conference, TACAS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, volume 6605 of *Lecture Notes in Computer Science*, pages 254–257. Springer, 2011.

[3] Nick Benton. Simple relational correctness proofs for static analyses and program transformations. In *POPL 2004*, pages 14–25, 2004.

[4] Arthur Azevedo de Amorim, Marco Gaboardi, Justin Hsu, and Shinya Katsumata. Metric semantics for probabilistic relational reasoning. *CoRR*, abs/1807.05091, 2018.

[5] Alexandre Donzé. Breach, A toolbox for verification and parameter synthesis of hybrid systems. In Tayssir Touili, Byron Cook, and Paul B. Jackson, editors, *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*, volume 6174 of *Lecture Notes in Computer Science*, pages 167–170. Springer, 2010.

[6] Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications. In Klaus Havelund, Manuel Núñez, Grigore Rosu, and Burkhart Wolff, editors, *Formal Approaches to Software Testing and Runtime Verification, First Combined International Workshops, FATES 2006 and RV 2006, Seattle, WA, USA, August 15-16, 2006, Revised Selected Papers*, volume 4262 of *Lecture Notes in Computer Science*, pages 178–192. Springer, 2006.

[7] Antoine Girard and George J. Pappas. Approximate bisimulation: A bridge between computer science and control theory. *Eur. J. Control*, 17(5-6):568–578, 2011.

[8] David Harel, Jerzy Tiuryn, and Dexter Kozen. *Dynamic Logic*. MIT Press, Cambridge, MA, USA, 2000.

[9] Ichiro Hasuo and Kohei Suenaga. Exercises in *Nonstandard Static Analysis* of hybrid systems. In P. Madhusudan and Sanjit A. Seshia, editors, *CAV*, volume 7358 of *Lect. Notes Comp. Sci.*, pages 462–478. Springer, 2012.

[10] Naoki Kobayashi. Model checking higher-order programs. *J. ACM*, 60(3):20:1–20:62, 2013.

[11] Timm Liebrenz, Paula Herber, and Sabine Glesner. Deductive verification of hybrid control systems modeled in simulink with keymaera X. In Jing Sun and Meng Sun, editors, *Formal Methods and Software Engineering - 20th International Conference on Formal Engineering Methods, ICFEM 2018, Gold Coast, QLD, Australia, November 12-16, 2018, Proceedings*, volume 11232 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 2018.

[12] Stefan Mitsch and André Platzer. The keymaera X proof IDE - concepts on usability in hybrid systems theorem proving. In Catherine Dubois, Paolo Masci, and Dominique Méry, editors, *Proceedings of the Third Workshop on Formal Integrated Development Environment, F-IDE@FM 2016, Limassol, Cyprus, November 8, 2016.*, volume 240 of *EPTCS*, pages 67–81, 2016.

[13] A. Platzer. The complete proof theory of hybrid systems. In *2012 27th Annual IEEE Symposium on Logic in Computer Science*, pages 541–550, June 2012.

[14] André Platzer. A complete uniform substitution calculus for differential dynamic logic. *J. Autom. Reas.*, 59(2):219–265, 2017.

[15] André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, 2018.

[16] André Platzer and Yong Kiam Tan. Differential equation axiomatization: The impressive power of differential ghosts. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 819–828. ACM, 2018.

[17] Abraham Robinson. *Non-standard analysis*. Princeton Univ. Press, 1966.

[18] Kohei Suenaga and Ichiro Hasuo. Programming with infinitesimals: A while-language for hybrid system modeling. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP (2)*, volume 6756 of *Lect. Notes Comp. Sci.*, pages 392–403. Springer, 2011.

[19] Kohei Suenaga, Hiroyoshi Sekine, and Ichiro Hasuo. Hyperstream processing systems: nonstandard modeling of continuous-time signals. In Roberto Giacobazzi and Radhia Cousot, editors, *POPL*, pages 417–430. ACM, 2013.

[20] Toru Takisaka, Yuichiro Oyabu, Natsuki Urabe, and Ichiro Hasuo. Ranking and repulsing supermartingales for reachability in probabilistic programs. In Shuvendu K. Lahiri and Chao Wang, editors, *Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings*, volume 11138 of *Lecture Notes in Computer Science*, pages 476–493. Springer, 2018.

[21] Zhenya Zhang, Gidon Ernst, Sean Sedwards, Paolo Arcaini, and Ichiro Hasuo. Two-layered falsification of hybrid systems guided by monte carlo tree search. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 37(11):2894–2905, 2018.

APPENDIX A
PROOF OF SOUNDNESS OF THE DIFFERENTIAL INDUCTIVE INVARIANT (DII_n)

To prove the soundness of this rule, we have to use some properties of higher derivatives of real functions:

Lemma A.1. *Fix $n \geq 1$. Let $h : [0,t] \longrightarrow \mathbb{R}$, where $t \in \mathbb{R}_{>0}$ be a function of class C^n, and let $\tilde{t} \in [0,t)$ be such that:*

- $\forall 0 \leq k < n.\ h^{(k)}(\tilde{t}) \geq 0$,
- $h^{(n)}(\tilde{t}) > 0$.

Then, there exists a $\tilde{t} < t' < t$ such that for all $\tilde{t} < t'' \leq t'$, $h(t'') > 0$.

Proof. We conduct the proof by induction on n.

- **case** $n = 1$**:** By definition, the derivative $\dot{h}(\tilde{t})$ of h at time $\tilde{t}$ is the following right limit approaching $\tilde{t}$:

  $$\dot{h}(\tilde{t}) = \lim_{t' \to \tilde{t}_+} \frac{h(t') - h(\tilde{t})}{t' - \tilde{t}}$$

  Thus in particular, for every $\epsilon > 0$, there exists $\eta > 0$ such that $\tilde{t} < t' < \min(\tilde{t}+\eta, t)$ implies $\dot{h}(\tilde{t}) - \epsilon < \frac{h(t')-h(\tilde{t})}{t'-\tilde{t}}$. Let us now fix arbitrary $\epsilon < \dot{h}(\tilde{t})$ and $\tilde{t} < t' < \min(\tilde{t}+\eta, t)$. Such ϵ and t' exist since $\dot{h}(\tilde{t}) > 0$ and $t > \tilde{t}$. Then, for every $\tilde{t} < t'' \leq t'$, $h(t'') > (\dot{h}(\tilde{t}) - \epsilon)(t'' - \tilde{t}) + h(\tilde{t}) > 0$ and $\tilde{t} < t' < t$.
- **case** $n > 1$**:** Applying the induction hypothesis on $\dot{h}$ we obtain time $\tilde{t} < t' < t$ such that for every $\tilde{t} < t'' \leq t'$, $\dot{h}(t'') > 0$. By Lagrange remainder's theorem, for any such $\tilde{t} < t'' \leq t'$, there exists $\tilde{t} < t''' < t''$ such that $h(t'') = h(\tilde{t}) + \dot{h}(t''') \cdot (t'' - \tilde{t}) > 0$.

□

Now, we are all set for proving the soundness of differential inductive invariant rules:

Proof (Of Theorem V.2). Fix $n \geq 1$. We conduct the proof by contraposition. Let $\boldsymbol{x}_0$ be an initial state such that $\boldsymbol{x}_0 \vDash \Gamma$ and let there exist a time $t \in \mathbb{R}_{\geq 0}$ such that $g(\psi(t)) < 0$ and $(\boldsymbol{x}_0, \psi(t)) \in [\![\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})\ \&\ Q]\!]$. Finally, let $h : [0,t] \to \mathbb{R}$ be the following C^n function:

$$t' \to g(\psi(t'))$$

Thus by definition $h^{(k)}(t') = \mathcal{L}_{\boldsymbol{f}}^{(k)} g(\psi(t'))$, for all $k \leq n$.

Since $[\![\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})\ \&\ Q]\!]$ is non-empty, we know $Q(\boldsymbol{x}_0)$ and therefore, by $\boldsymbol{x}_0 \vDash \Gamma$, $\Gamma, Q \vdash 0 \leq g(\boldsymbol{x}) = h(0)$.

Let us now prove that the set:

$$U = \{t' \in [0,t] \mid \forall t'' \leq t'.\ h(t'') \geq 0\}$$

is closed in $[0,t]$. Since $[0,t]$ is a complete metric space, it is enough to prove that U is closed under limits. Let us thus consider a sequence $(t_n)_{n\in\mathbb{N}} \subseteq U^{\mathbb{N}}$ that converges to $t_\infty \in [0,t]$. We want to prove that for every $t'' \leq t_\infty$, $h(t'') \geq 0$. First, since h is continuous, $h^{-1}([0,\infty))$ is closed. Furthermore, since $t_n \in U \subseteq h^{-1}([0,\infty))$, we have $t_\infty \in h^{-1}([0,\infty))$ and thus $h(t_\infty) \geq 0$. Now let us take $0 \leq t'' < t_\infty$ arbitrary. Since $(t_n)_n$ converges to t_∞, there exists an integer n such that $|t_\infty - t_n| < \epsilon$ for every $\epsilon > 0$. Thus by choosing $0 < \epsilon < t_\infty - t''$, there is an integer n such that $t'' < t_n$. Then, since $t_n \in U$, $h(t'') \geq 0$. Consequently, U is non-empty and compact in $\mathbb{R}$, and thus has a maximal element $\tilde{t}$. Observe that $\tilde{t} < t$ due to $g(\psi(t)) < 0$.

We now show that $\psi(\tilde{t})$ falsifies the second premise. Since $0 < \tilde{t} < t$ and $\tilde{t} \in U$ we know $(\boldsymbol{x}_0, \psi(\tilde{t})) \in [\![\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})\ \&\ Q \wedge g(\boldsymbol{x}) \geq 0]\!]$. We now prove that $D_n g(\psi(\tilde{t}))$ does not hold by contradiction. This means that we assume that there is $p < n$ such that for all $k \leq p$, $h^{(k)}(\tilde{t}) \geq 0$ and $h^{(p+1)}(\tilde{t}) > 0$. Then by Lemma A.1, there is $\tilde{t} < t' < t$ such that for all $\tilde{t} < t'' \leq t'$, $h(t'') > 0$. Thus, $t' \in U$, which contradicts the maximality of $\tilde{t}$.

APPENDIX B
PROOF OF SOUNDNESS OF AUXILIARY RELATIONAL RULES §V-B

A. Soundness of Monotonic Condition Swap (MSC)

Let $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp_0) \vDash \Gamma$ be an arbitrary initial state satisfying the premises.

We conduct the proof by contradiction. Let thus $\boldsymbol{x}_1$ and $\boldsymbol{x}^\sharp_1$ be such that $\boldsymbol{x}_0 \xrightarrow{T_\delta} \boldsymbol{x}_1$, $\boldsymbol{x}^\sharp_0 \xrightarrow{T_{\delta^\sharp}} \boldsymbol{x}^\sharp_1$, $g(\boldsymbol{x}_1) = g^\sharp(\boldsymbol{x}^\sharp_1)$ and $h(\boldsymbol{x}_1) > h^\sharp(\boldsymbol{x}^\sharp_1)$.

Let us now consider $\boldsymbol{x}_2$ such that $h(\boldsymbol{x}_2) = h^\sharp(\boldsymbol{x}^\sharp_1)$. We know such $\boldsymbol{x}_2$ exists as $h(\boldsymbol{x}_0) \leq h^\sharp(\boldsymbol{x}^\sharp_0)$ and both h and $h^\sharp$ are increasing. Moreover, by $h(\boldsymbol{x}_0) \leq h(\boldsymbol{x}_2)$ and $h(\boldsymbol{x}_2) < h(\boldsymbol{x}_1)$ we have $\boldsymbol{x}_0 \xrightarrow{T_\delta} \boldsymbol{x}_2$ and $\boldsymbol{x}_2 \xrightarrow{T_\delta} \boldsymbol{x}_1$.

Since $h(\boldsymbol{x}_2) = h^\sharp(\boldsymbol{x}^\sharp_1)$, we can use the first premise to get $g(\boldsymbol{x}_2) \geq g^\sharp(\boldsymbol{x}^\sharp_1)$. Finally, since g is strictly increasing, we get $g(\boldsymbol{x}_1) > g(\boldsymbol{x}_2) \geq g^\sharp(\boldsymbol{x}^\sharp_1)$ which is a contradiction with $g(\boldsymbol{x}_1) = g^\sharp(\boldsymbol{x}^\sharp_1)$.

B. Soundness of Relational Differential Cut (RDC)

Let $(\boldsymbol{x}_0 \cdot \boldsymbol{x}^\sharp_0, \boldsymbol{x}_1 \cdot \boldsymbol{x}^\sharp_1) \in [\![\delta; \delta^\sharp; ?P]\!]$ such that $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp_0)$ satisfies the premises. From the second premise we obtain $g(\boldsymbol{x}_1) = g^\sharp(\boldsymbol{x}^\sharp_1)$. Thus, in turn, the first premise yields $B(\boldsymbol{x}_0, \boldsymbol{x}^\sharp_0)$.

C. Soundness of Exit Condition Propagation (ECP)

We conduct the proof for version where $\sim$ stands for $\geq$, the version with $\leq$ is symmetric.

Let $(\boldsymbol{x}_0, \boldsymbol{x}^\sharp_0)$ be an arbitrary initial state satisfying the premise and let $\boldsymbol{x}^\sharp_1$ be such that $(\boldsymbol{x}^\sharp_0, \boldsymbol{x}^\sharp_1) \in [\![\delta^\sharp_1]\!]$ such that $P(\boldsymbol{x}^\sharp_1)$. Since $\mathcal{L}_{\boldsymbol{f}^\sharp_1} g^\sharp \geq 0$ at any $\delta^\sharp_1$-reachable state, we know $g^\sharp(\boldsymbol{x}^\sharp_0) \leq g^\sharp(\boldsymbol{x}^\sharp_1)$.

Let further $\boldsymbol{x}_2$ and $\boldsymbol{x}^\sharp_2$ be such that $(\boldsymbol{x}^\sharp_0, \boldsymbol{x}^\sharp_2) \in [\![\delta^\sharp_2]\!]$, $(\boldsymbol{x}_0, \boldsymbol{x}_2) \in [\![\delta]\!]$ and $E(\boldsymbol{x}_2, \boldsymbol{x}^\sharp_2)$. Since $\mathcal{L}_{\boldsymbol{f}^\sharp_2} g^\sharp \geq 0$ at any $\delta^\sharp_2$-reachable state, we know $g^\sharp(\boldsymbol{x}^\sharp_1) \leq g^\sharp(\boldsymbol{x}^\sharp_2)$.

Let now $\boldsymbol{x}_1$ be such that $g(\boldsymbol{x}_1) = g^\sharp(\boldsymbol{x}^\sharp_1)$. Then, by $g(\boldsymbol{x}_0) = g^\sharp(\boldsymbol{x}^\sharp_0)$ and $g(\boldsymbol{x}_2) = g^\sharp(\boldsymbol{x}^\sharp_2)$ we also have $g(\boldsymbol{x}_0) \leq g(\boldsymbol{x}_1) \leq g(\boldsymbol{x}_2)$. From continuity of g, such $\boldsymbol{x}_1$ must thus be reachable by δ from $\boldsymbol{x}_0$ and $\boldsymbol{x}_2$ has to be reachable by δ from $\boldsymbol{x}_1$. Then, by the first premise, $(\boldsymbol{x}_2, \boldsymbol{x}^\sharp_2) \vDash \varphi$

$$0 < a \leq a^\sharp, 0 = x = x^\sharp, 0 < v = v^\sharp \vdash \left[\left\langle \frac{\dot{x} = v, \dot{v} = a}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp} \right\rangle\right] x = x^\sharp \Big] v \leq v^\sharp \tag{44}$$

(45), (46), (47), (48) and (49), by (MSC) (Figure 2)

$$0 < a \leq a^\sharp, 0 = x = x^\sharp, 0 < v = v^\sharp \vdash v \leq v^\sharp \tag{45}$$

arithmetic fact

$$0 < a \leq a^\sharp, 0 = x = x^\sharp, 0 < v = v^\sharp \vdash [\dot{x} = v, \dot{v} = a] v > 0 \tag{46}$$

(DI) and arithmetic fact

$$0 < a \leq a^\sharp, 0 = x = x^\sharp, 0 < v = v^\sharp \vdash [\dot{x} = v, \dot{v} = a] a \geq 0 \tag{47}$$

(DW) and arithmetic fact

$$0 < a \leq a^\sharp, 0 = x = x^\sharp, 0 < v = v^\sharp \vdash \left[\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp\right] a^\sharp \geq 0 \tag{48}$$

(DW) and arithmetic fact

$$0 < a \leq a^\sharp, 0 = x = x^\sharp, 0 < v = v^\sharp \vdash \left[\left\langle \frac{\dot{x} = v, \dot{v} = a}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp} \right\rangle\right] v = v^\sharp \Big] x \geq x^\sharp \tag{49}$$

(50), (51), (52), by (TS) (Definition IV.11)

$$0 < a \leq a^\sharp, 0 = x = x^\sharp, 0 < v = v^\sharp \vdash v = v^\sharp \tag{50}$$

arithmetic fact

$$0 < a \leq a^\sharp, 0 = x = x^\sharp, 0 < v = v^\sharp \vdash \left\langle \frac{\dot{x} = v, \dot{v} = a}{\dot{x^\sharp} = v^\sharp, \dot{v^\sharp} = a^\sharp} \right\rangle \frac{a}{a^\sharp} > 0 \tag{51}$$

(DW) and arithmetic fact

$$0 < a \leq a^\sharp, 0 = x = x^\sharp, 0 < v = v^\sharp \vdash \left[\dot{x} = v, \dot{v} = a, \dot{x^\sharp} = \frac{v^\sharp \cdot a}{a^\sharp}, \dot{v^\sharp} = \frac{a^\sharp \cdot a}{a^\sharp}\right] x \geq x^\sharp \tag{52}$$

(DI) twice and arithmetic fact

Fig. 7. The derivation of formula φ_C (7) in dL using the (MSC) rule.

D. Soundness of Sequential Composition Commutativity (SCC)

We can interpret α and $\alpha^\sharp$ solely in their respective variables $\boldsymbol{x}$ and $\boldsymbol{x}^\sharp$ with any additional variables having no effect on the outcome. Thus, if $(\boldsymbol{x}_0, \boldsymbol{x}_1) \in [\![\alpha(\boldsymbol{x})]\!]$, we also have $(\boldsymbol{x}_0 \cdot \boldsymbol{x}^\sharp{}_0, \boldsymbol{x}_1 \cdot \boldsymbol{x}^\sharp{}_0) \in [\![\alpha(\boldsymbol{x})]\!]$ for arbitrary $\boldsymbol{x}^\sharp{}_0$. Similarly, $(\boldsymbol{x}^\sharp{}_0, \boldsymbol{x}^\sharp{}_1) \in [\![\alpha^\sharp(\boldsymbol{x}^\sharp)]\!]$ gives us $(\boldsymbol{x}_0 \cdot \boldsymbol{x}^\sharp{}_0, \boldsymbol{x}_0 \cdot \boldsymbol{x}^\sharp{}_1) \in [\![\alpha^\sharp(\boldsymbol{x}^\sharp)]\!]$ for arbitrary $\boldsymbol{x}_0$.

E. Soundness of Merge Identical Dynamics (MID)

Let $(\boldsymbol{x}_0, \boldsymbol{x}_1) \in [\![\delta]\!]$ be arbitrary. Surely, $Q(\boldsymbol{x}_1)$ holds, thus we also have $(\boldsymbol{x}_1, \boldsymbol{x}_1) \in [\![\delta]\!]$. Therefore, by definition of interpretation of sequential composition (Definition II.4), $(\boldsymbol{x}_0, \boldsymbol{x}_1) \in [\![\delta; \delta]\!]$.

APPENDIX C
PROOF OF COLLISION SPEED WITH CONSTANT ACCELERATION USING MSC

The alternative proof of the formula φ_C (7) in Section VI-A which relies on (MSC) (Figure 2) instead of (DII_1) is given in Figure 7.

The simplicity of the proof has been achieved chiefly by v and $v^\sharp$, which are swapped into the exit condition, having constant valued Lie derivatives.