A COMPLETE AXIOMATIZATION OF QUANTIFIED DIFFERENTIAL DYNAMIC LOGIC FOR DISTRIBUTED HYBRID SYSTEMS

ANDRÉ PLATZER

Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, USA
e-mail address: aplatzer@cs.cmu.edu

Abstract. We address a fundamental mismatch between the combinations of dynamics that occur in cyber-physical systems and the limited kinds of dynamics supported in analysis. Modern applications combine communication, computation, and control. They may even form dynamic distributed networks, where neither structure nor dimension stay the same while the system follows hybrid dynamics, i.e., mixed discrete and continuous dynamics.

We provide the logical foundations for closing this analytic gap. We develop a formal model for distributed hybrid systems. It combines quantified differential equations with quantified assignments and dynamic dimensionality-changes. We introduce a dynamic logic for verifying distributed hybrid systems and present a proof calculus for this logic. This is the first formal verification approach for distributed hybrid systems. We prove that our calculus is a sound and complete axiomatization of the behavior of distributed hybrid systems relative to quantified differential equations. In our calculus we have proven collision freedom in distributed car control even when an unbounded number of new cars may appear dynamically on the road.

1. Introduction

Many safety-critical computers are embedded in cyber-physical systems like cars [HESV91, SRS+06] and aircraft [DMC05]. How do we know that their designs will work as intended? Most initial designs do not. And some deployed systems still do not. Ensuring the correct functioning of cyber-physical systems is a central challenge in computer science, mathematics, and engineering, because it is the key to designing smart and reliable control. Scientists

1998 ACM Subject Classification: F.3.1 Logics and Meanings of Programs: Specifying and Verifying and Reasoning about Programs, F.4.1 Mathematical Logic and Formal Languages: Mathematical Logic, D.2.4 Software Engineering: Software/Program Verification, C.1.m: Hybrid Systems, C.2.4 Computer-Communication Networks: Distributed Systems, D.4.7 Organization and Design: Distributed Systems.

Key words and phrases: Differential dynamic logic, Distributed hybrid systems, Axiomatization, Theorem proving, Quantified differential equations, Proof theory.

An extended abstract has appeared at CSL'10 [Pla10c].

This material is based upon work supported by the National Science Foundation under NSF CAREER Award CNS-1054246, NSF EXPEDITION CNS-0926181, and under Grant Nos. CNS-1035800 and CNS-0931985, by the NASA grant NNG-05GF84H, and by the ONR award N00014-10-1-0188.

DOI:10.2168/LMCS-???
© André Platzer
Creative Commons

and engineers need analytic tools to understand and predict the behavior of their systems. As systems become ever more complex, it becomes prohibitively expensive or impossible to test all possible interactions and rule out unsafe behavior by simulation. Formal verification techniques are used routinely to overcome this for finite systems. But for cyber-physical systems, there is not even a foundation for verification that would cover all required behavior.

There is a fundamental mismatch between the actual dynamics of cyber-physical system applications and the limits imposed by current modeling and analysis. Cyber-physical systems in automotive, aviation, railway, and power grids combine *communication, computation, and control*. Combining computation and control leads to *hybrid systems* [ACHH92, Bra95, Hen96, BBM98, Pla10b], whose behavior involves both discrete and continuous dynamics originating, e.g., from discrete control decisions and differential equations of motion. Combining communication and computation leads to *distributed systems* [Lyn96, AL01, AdBO10], whose dynamics are discrete transitions of system parts that communicate with each other. They may form *dynamic distributed systems*, where the structure of the system is not fixed but evolves over time and agents may appear or disappear during the system evolution.

Combinations of all three aspects (communication, computation, and control) are used in sophisticated applications, e.g., cooperative distributed car control [HESV91] and decentralized aircraft control [PSFB07]. Neither the structure nor dimension of the system stay the same, because new cars can appear on the street or leave it; see Fig. 1. These systems are *(dynamic) distributed hybrid systems*, i.e., systems that combine the dynamics of distributed systems with the discrete and continuous dynamics of hybrid systems. More generally, distributed hybrid systems are multi-agent hybrid systems that interact through remote communication or physical interaction. They cannot be considered just as a distributed system (because, e.g., the continuous evolution of positions and velocities matters crucially for collision freedom in car control) nor just as a hybrid system (because the evolving system structure and appearance of new agents can make an otherwise collision-free system unsafe). It is generally impossible to split the analysis of distributed hybrid systems soundly into an analysis of a distributed system (without continuous movement) and an analysis of a hybrid system (without structural changes or appearance), because all kinds of dynamics interact. Just like hybrid systems are diffcult to analyze from a purely discrete or a purely continuous perspective [Hen96, Pla12].

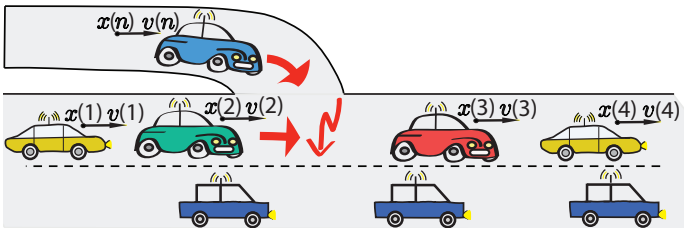


Figure 1: Distributed car control.

Distributed hybrid systems have been considered to varying degrees in modeling languages [DGV96, Rou04, KSPL06, MS06]. In order to build these systems, however, scientists and engineers also need analytic tools to understand and predict their behavior. But formal verification and proof techniques do not yet support the required combination of dynamical effects—which is not surprising given the numerous sources of undecidability for distributed hybrid systems verification.

In this article, we provide the logical foundations to close this fundamental analytic gap. We develop *quantified hybrid programs* (QHPs) as a formal model for distributed hybrid systems, which combine dynamical effects from multiple sources: *discrete transitions, continuous evolution, dimension changes, and structural dynamics.* In order to account for changes in the dimension and for co-evolution of an unbounded and evolving number of participants, we generalize the notion of states from assignments for primitive system variables like x to full first-order structures. In a QHP, function term $x(i)$ may denote the position of car i of type C, the term $f(i)$ could be the car registered by communication as the car following car i, and the term $d(i, f(i))$ could denote the minimum safety distance negotiated between car i and its follower $f(i)$. The values of all these terms may evolve *for all* i as time progresses according to interacting laws of discrete and continuous dynamics, because all cars evolve simultaneously. They are also affected by changing the system dimension as new cars appear, disappear, or by reconfiguring the system structure dynamically, e.g., by remote communication or physical interaction. The defining characteristic of QHPs is that they allow *quantified hybrid dynamics* in which variables like i that occur in function arguments of the system dynamics are quantified over, such that the system co-evolves, e.g., *for all* cars i of type C. This quantification is necessary to characterize the distributed hybrid systems dynamics with an unbounded and possibly evolving number of participants. Quantification is also necessary to represent structural dynamics when the number of participants is not fixed.

There is a crucial difference between a primitive system variable x and a first-order function term $x(i)$, where i is quantified over. Hybrid dynamics of primitive system variables can model a concrete number of, say, four cars (putting scalability issues aside), but neither a parametric number of n cars nor systems with a variable number of cars (a number n that may change over time). With first-order function symbols $x(i)$ and hybrid dynamics quantifying over all cars i, a single QHP can represent *any* number of cars at once. QHPs can even represent (dis)appearance of cars by changing the domain that quantifiers range over dynamically at runtime. QHPs are thus a formal model for general (dynamic) distributed hybrid systems.

Verification of distributed hybrid systems is challenging. We show that they have three independent sources of undecidability: discrete dynamics, continuous dynamics, and structural/dimensional dynamics. As an analysis tool for distributed hybrid systems, we introduce a specification and verification logic for QHPs that we call *quantified differential dynamic logic* (QdℒC). QdℒC provides dynamic logic [Pra76, HKT00] modal operators $[\alpha]$ and $\langle\alpha\rangle$ that refer to the states reachable by QHP α and can be placed in front of any formula. Formula $[\alpha]\phi$ expresses that all states reachable by system α satisfy formula ϕ, while $\langle\alpha\rangle\phi$ expresses that there is at least one reachable state satisfying ϕ. These modalities can express necessary or possible properties of the transition behavior of QHP α. With its ability to specify and verify properties of (dynamic) distributed hybrid systems and quantified dynamics, QdℒC is a major extension of prior work for static hybrid systems [Pla08a, Pla10a] and conventional discrete programs [BP06, Rüm06].

Our primary contributions are:

- We introduce a *formal system model and semantics* that succinctly captures the logical quintessence of (dynamic) distributed hybrid systems with joint discrete, continuous, structural, and dimension-changing dynamics.
- We introduce a *specification and verification logic* for (dynamic) distributed hybrid systems.

- We present a *proof calculus* for this logic, which, to the best of our knowledge, is the *first verification approach* that can handle distributed hybrid systems with their hybrid dynamics and unbounded (and evolving) dimensions and structure.
- We prove that this compositional calculus is a *sound and complete axiomatization* of (dynamic) distributed hybrid systems relative to quantified differential equations.
- We have used our proof calculus to verify *collision freedom in a distributed car control system*, where an unbounded number of new cars may appear dynamically on the road.

In particular, we extend our previous extended abstract [Pla10c] by 28 pages worth of

- soundness and relative completeness proofs
- new results on ineffective fragments
- more detailed explanations and more examples
- new derived proof rules
- new formal proofs illustrating the interaction of quantifiers, first-order function symbols, and quantified system dynamics in detail
- a proof of collision avoidance in a simple distributed car control system, and a new result about a more advanced distributed car control system.

This work constitutes the logical foundation for analysis of distributed hybrid systems. Since distributed hybrid control is the key to control numerous advanced systems, analytic approaches have significant potential for applications. With a theorem prover based on our approach, we have verified collision avoidance in a distributed car control system, which is out of scope for other approaches. The approach presented here has been used subsequently for verifying distributed adaptive cruise control systems for highways [LPN11] and distributed air traffic control [Pla11].

Our verification approach for distributed hybrid systems is a fundamental extension compared to previous approaches. In much the same way as first-order logic increases the expressive power over propositional logic (quantifiers and function symbols are required to express properties of unbounded structures), $\mathsf{Qd}\mathcal{L}$ increases the expressive power over its predecessors (because first-order functions and quantifiers in the dynamics of QHPs are required to characterize systems with unbounded and changing dimensions).

2. Related Work

Multi-party distributed control has been suggested for car control [HESV91] and air traffic control [DMC05]. Due to limits in verification technology, no formal analysis of the distributed hybrid dynamics has been possible for these systems yet. Analysis results include discrete message handling [HESV91] or collision avoidance for two participants [DMC05]. In distributed car control and air traffic control systems, appearance of new participants is a major unsolved challenge for formal verification.

Ad-hoc informal arguments have been used to discuss distribution effects away, e.g., assuming that at most 4 cars are close to one another. These arguments are treacherous, though. They are very case-specific and do not lend themselves to formal verification within one proof system because they need arguments outside the proof system to work. In distributed car control, for instance, it might, at first sight, be convincing to suspect that it would be enough to consider every possible constellation of, say, four cars. This breaks down at second thought, though, because, without a formal proof, there is no reason to believe that a locally consistent and safe system would be globally safe and consistent. Consider

an example for the situation in Fig. 1, for instance. Even if hybrid systems verification techniques could show that local patterns consisting of the four cars $\{1, 2, n, 3\}$ are safe and that local patterns consisting of the four cars $\{2, n, 3, 4\}$ are safe, the full system consisting of all cars $\{1, 2, n, 3, 4\}$ still does not have to be safe. For example, the local pattern $\{1, 2, n, 3\}$ could be safe, because it will ask car n to change lanes and ask car 2 to keep speed and car 3 to speed up. But the pattern $\{2, n, 3, 4\}$ could be safe, because it will ask car n to change lanes but, instead, ask car 2 to slow down and car 3 to keep speed. Those two locally safe patterns still lead to a globally incompatible maneuver choice resulting in a crash, because both cars 2 and 3 would be forced to keep the speed (for they would otherwise collide with car 1 or 4, respectively) and, henceforth, collide with car n during its lane change. More generally, independent actions in different parts of a system may still end up interacting by rippling effects. It is, thus, crucial to understand and verify the emergent behavior resulting from local control principles. The full distributed hybrid systems dynamics needs to be considered and we cannot generally hope to prove meaningful properties by simply ignoring part of the dynamics.

The importance of understanding dynamic / reconfigurable distributed hybrid systems was recognized in modeling languages SHIFT [DGV96] and R-Charon [KSPL06] before. They focused on simulation and compilation [DGV96] or the development of a semantics [KSPL06], so that no verification is possible yet. For stochastic simulation, see [MS06], where soundness has not been proven, because ensuring coverage is difficult by a random simulation. See [ZPC10] for a discussion of statistical evidence that can be obtained for randomized discrete-time hybrid systems by fair (i.i.d. sampled) simulation. This technique neither covers distributed hybrid systems nor continuous-time hybrid systems nor nondeterministic dynamics, all of which we cover in this article.

For distributed hybrid systems, even giving a formal semantics is very challenging [CJR95, Rou04, KSPL06, vBMR$^+$06]! Zhou et al. [CJR95] gave a semantics for a hybrid version of CSP in the Extended Duration Calculus [ZRH92]. Rounds [Rou04] gave a semantics in a rich set theory for a spatial logic for a hybrid version of the π-calculus. In the hybrid π-calculus, processes interact with a continuously changing environment, but cannot themselves evolve continuously, which would be crucial to capture the physical movement of traffic agents. From the semantics alone, no verification is possible in these approaches, except perhaps by manual reasoning in the semantics.

Other process-algebraic approaches, like χ [vBMR$^+$06], have been developed for modeling and simulation purposes. Verification is still limited to small fragments that can be translated directly to other verification tools like PHAVer or UPPAAL, which have fixed dimensions and restricted dynamics (thus no distributed hybrid systems).

Our approach is completely different. It is based on first-order structures and dynamic logic. We focus on developing a logic that supports distributed hybrid dynamics directly and that is amenable to automated theorem proving in the logic itself.

For a detailed discussion of verification approaches for static real-time and hybrid systems, we refer to previous work [Pla08a, Pla10a, Pla08b, Pla10b]. Our previous work and other verification approaches for static hybrid systems cannot verify distributed hybrid systems. Distributed hybrid systems may have an unbounded and changing number of components/participants, which cannot be represented with any fixed finite number of dimensions of the state space. In distributed car control, for instance, there is no prior limit on the number of cars on the street. Even when there is a limit, explicit replication of the

system, say, 100 times, does not yield a scalable verification approach, because most hybrid systems verification approaches scale exponentially in the number of participants or worse.

Approaches for distributed systems [AL01] do not cover hybrid systems, because the addition of differential equations to distributed systems is even more challenging than the addition of differential equations to discrete dynamics has been when forming hybrid systems. There is not even a bound on the number of differential equations that would need to be added to faithfully hybridize a distributed system.

In summary, previous approaches to distributed hybrid systems are limited to modeling, simulation, or the definition of a semantics. No formal verification technique was known for distributed hybrid systems before.

3. Syntax

As a formal logic for specifying and verifying correctness properties of distributed hybrid systems, we introduce *quantified differential dynamic logic* (Q$\mathsf{d}\mathcal{L}$). Q$\mathsf{d}\mathcal{L}$ combines dynamic logic for reasoning about all ($[\alpha]\phi$) or some ($\langle\alpha\rangle\phi$) system runs of a system α [Pra76, HKT00] with many-sorted first-order logic for reasoning about all ($\forall i:C\ \phi$) or some ($\exists i:C\ \phi$) objects of a sort C, e.g., the sort of all cars. The most important defining characteristic of Q$\mathsf{d}\mathcal{L}$ is that α can be a distributed hybrid system, because the Q$\mathsf{d}\mathcal{L}$ system model of *quantified hybrid programs* (QHP) supports quantified operations that affect *all* objects of a sort C at once. If C is the sort of cars, the quantified assignment $\forall i:C\ a(i):=a(i)+1$ increases the respective accelerations $a(i)$ of *all cars* i at once by a single instantaneous discrete jump. It can be used to model simultaneous discrete changes in multiple agents at once. Discrete changes where only some of the cars change their acceleration, others do not, are easy to model with quantified assignments by masking. The quantified differential equation $\forall i:C\ v(i)'=a(i)$ represents a continuous evolution of the respective velocities $v(i)$ of *all cars* i at the same time according to their acceleration by their respective differential equations $v(i)'=a(i)$. Again, continuous evolutions where only some of the cars evolve, others remain stopped, are easy to model with quantified differential equations by masking. These quantified assignments and quantified differential equation systems of QHPs are crucial for representing distributed hybrid systems where an unbounded number of objects co-evolve simultaneously, because no finite set of classical assignments and classical differential equations could represent that. Note that, because of the close semantical relationship, we use the same quantifier notation $\forall i:C$ for quantified operations in programs and for quantifiers in logical formulas, instead of a separate notation $\Pi_{i:C}$ for parallel products in programs.

Interaction by communication can be modeled by (possibly quantified) discrete assignments to share data between agents i and j in QHPs. Physical interaction, instead, may be modeled either by (possibly quantified) discrete assignments when an agent i activates a response in agent j by an instantaneous discrete action (e.g., pushing a physical button) or by a (possibly quantified) differential equation involving multiple agents i and j when they come into physical contact and act jointly over a (nonzero) period of time (e.g., both agents jointly lifting and pulling on a rigid object). Observe that the cyber structure of the system reconfigures dynamically when discrete communication topologies change, whereas the physical structure reconfigures dynamically when agents engage in physical contact. QHPs for the latter case may involve structural changes in the quantified differential equation.

We model the appearance of new participants in the distributed hybrid system, e.g., new cars entering the road, by a QHP $n:=\mathsf{new}\,C$. It creates a new object of type C, thereby

extending the range of all subsequent quantified assignments or quantified differential equations ranging over created objects of type C. With quantifiers and function terms, new can be handled in an entirely modular way. In order to reduce the conceptual complexity, we first focus on the syntax and semantics of QdℒC and postpone the discussion of actual existence and creation until Section 5. We will see that actual existence and creation are completely modular extensions.

The model of QHPs is of independent interest as a formal model for distributed hybrid systems. Inside a QHP, logical formulas can occur in state tests for conditional execution. We thus explain logical formulas, terms, and sorts first. Conversely, however, a QHP α occurs inside the modalities ($[\alpha]$ and $\langle\alpha\rangle$) of QdℒC formulas, which state properties of the behavior of α. Hence, QHPs may occur inside QdℒC formulas yet formulas may occur inside QHPs. The subsequent definitions of QdℒC and QHP are thus to be understood by simultaneous induction. It is easier to start with sorts, terms, and logical formulas first and then explain the QHP model subsequently.

3.1. **Quantified Differential Dynamic Logic.** We introduce quantified differential dynamic logic (QdℒC), which is the first formal logic for specifying and verifying correctness properties of distributed hybrid systems. QdℒC is a combination of many-sorted first-order logic with dynamic logic, generalized to a system model (QHPs) for distributed hybrid systems.

Sorts. QdℒC supports a (finite) number of object sorts, e.g., the sort of all cars and that of all aircraft. For continuous quantities of distributed hybrid systems like positions or velocities, we add the sort $\mathbb{R}$ of real numbers. It would be easy to add subtyping of sorts; see previous work [BP06] for details. We refrain from doing so, because that just obscures the logical essence of our approach.

The primary purpose of the sorts is to distinguish different kinds of objects in multi-agent hybrid systems in which different kinds of agents occur, e.g., cars of sort C, traffic lights of sort T, lanes of sort L, and aircraft of sort A.

Terms. QdℒC terms are built from a set of (sorted) function and variable symbols as in many-sorted first-order logic. In particular, each function symbol f has a fixed type $C_1 \times \cdots \times C_n \to D$ for some $n \in \mathbb{N}$ and some sorts $D, C_1, \ldots, C_n$ such that f only accepts argument terms $\theta_1, \ldots, \theta_n$ of the respective sorts $C_1, \ldots, C_n$ and then $f(\theta_1, \ldots, \theta_n)$ is a term of sort D. We use these function symbols to represent the state of the system or other parameters. In a car control scenario like that in Fig. 1, for instance, we could use function symbol x to represent the positions of cars, i.e., the term $x(i)$ could represent the position of car i and $x(j)$ the position of car j. Similarly, the term $v(i)$ could represent the velocity of car i and $a(i)$ its acceleration. These terms have sort $\mathbb{R}$, whereas a term $l(i)$ that represents the car in front of car i has sort C.

Unlike in first-order logic, the interpretation of function symbols can change when transitioning from one state to the other while following the dynamics of a distributed hybrid system. The value of position $x(i)$ will change over time as car i drives down the street. The value of $x(i)$ would also change if the argument term i changes its value and now refers to a different car than before. Even objects may appear or disappear as the distributed hybrid system evolves. We use function symbol $\mathsf{E}(\cdot)$ to distinguish between objects i that actually exist ($\mathsf{E}(i) = 1$) and those that have not been created yet or exist no

longer ($\mathsf{E}(i) = 0$), depending on the value of $\mathsf{E}(i)$, which may also change its interpretation from state to state. We use $0, 1, +, -, \cdot$ with the usual notation and fixed semantics for nonlinear real arithmetic. Divisions can be added when guarding against divisions by zero [Pla08a]. For $n \geq 0$ we abbreviate $f(s_1, \ldots, s_n)$ by $f(\vec{s})$ using vectorial notation and we use $\vec{s} = \vec{t}$ for component-wise equality.

Formulas. The formulas of Qd$\mathcal{L}$ are defined as in first-order dynamic logic plus many-sorted first-order logic .

Definition 3.1. (Qd$\mathcal{L}$ FORMULAS).The formulas of Qd$\mathcal{L}$ are defined by the following grammar (ϕ, ψ are formulas, θ_1, θ_2 are terms of the same sort, i is a variable of sort C, and α is a QHP as defined in Section 3.2):

$$\phi, \psi \ ::= \ \theta_1 = \theta_2 \mid \theta_1 \geq \theta_2 \mid \neg\phi \mid \phi \wedge \psi \mid \forall i : C \ \phi \mid \exists i : C \ \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi$$

We use standard abbreviations to define $\leq, >, <, \vee, \rightarrow$. Sorts $C \neq \mathbb{R}$ have no ordering and only $\theta_1 = \theta_2$ is allowed, not $\theta_1 \geq \theta_2$. For sort $\mathbb{R}$, we abbreviate $\forall x : \mathbb{R} \ \phi$ by $\forall x \ \phi$ and $\exists x : \mathbb{R} \ \phi$ by $\exists x \ \phi$. In the following, all formulas and terms have to be well-typed. For instance, $x(i) = l(i)$ is no formula if x has type $C \rightarrow \mathbb{R}$ and l has type $C \rightarrow C$ for a sort $C \neq \mathbb{R}$ or if i has a sort $D \neq C$. Qd$\mathcal{L}$ formula $[\alpha]\phi$ expresses that *all states* reachable by QHP α satisfy formula ϕ. Likewise, $\langle\alpha\rangle\phi$ expresses that *there is at least one state* reachable by α for which ϕ holds.

For short notation, we allow *conditional terms* of the form $\mathsf{if}\,\phi\,\mathsf{then}\,\theta_1\,\mathsf{else}\,\theta_2\,\mathsf{fi}$ (where θ_1 and θ_2 have the same sort). This term evaluates to θ_1 if the formula ϕ is true and to θ_2 otherwise. We generally consider formulas with conditional terms as abbreviations, e.g., $\psi(\mathsf{if}\,\phi\,\mathsf{then}\,\theta_1\,\mathsf{else}\,\theta_2\,\mathsf{fi})$ abbreviates $(\phi \rightarrow \psi(\theta_1)) \wedge (\neg\phi \rightarrow \psi(\theta_2))$. Conditional terms can be understood as an additional operator for terms and formulas as well.

Example. A major challenge in distributed car control systems [HESV91] is that they do not follow fixed, static setups. Instead, new situations can arise dynamically that change structure and dimension of the system whenever new cars appear on the road from on-ramps or leave it; see Fig. 1. As a running example, we model a *distributed car control system DCCS*. First, we consider desirable Qd$\mathcal{L}$ properties of the system *DCCS* for which we will later develop a series of increasingly more realistic models as QHPs.

If i is a term of type C (for cars), let $x(i)$ denote the position of car i, $v(i)$ its current velocity, and $a(i)$ its current acceleration; see Fig. 1. A car control system is collision-free at a state if all cars are at different positions, i.e., $\forall i{\neq}j : C \ x(i){\neq}x(j)$. Without a quantifier we could not describe that all cars on a highway are in a collision-free state, because there is a large number of cars on the highway and we may not know how many. The car control system is globally collision-free if it will always stay collision-free. The following Qd$\mathcal{L}$ formula expresses that the system *DCCS* controls cars in a way that is always collision-free:

$$(\forall i, j : C \ \mathcal{M}(i, j)) \ \rightarrow \ [DCCS] \ \forall i{\neq}j : C \ x(i){\neq}x(j) \tag{3.1}$$

It says that cars following the distributed hybrid systems dynamics of *DCCS* are always collision-free (postcondition), provided that *DCCS* starts in an initial state satisfying a formula $\mathcal{M}(i, j)$ for all cars i, j (precondition). In particular, the modality $[DCCS]$ expresses that all states reachable by following the distributed hybrid system *DCCS* satisfy the postcondition $\forall i{\neq}j : C \ x(i){\neq}x(j)$. The simple-most choice for the formula $\mathcal{M}(i, j)$ in

the precondition is a formula that characterizes a simple compatibility condition: for different cars $i \neq j$, the car that is further down the road (i.e., with greater position) neither moves slower nor accelerates slower than the other car, i.e.:

$$\begin{aligned}\mathcal{M}(i,j) \;\equiv\; i \neq j \rightarrow \big((x(i) < x(j) \wedge v(i) \leq v(j) \wedge a(i) \leq a(j)) \\ \vee\, (x(i) > x(j) \wedge v(i) \geq v(j) \wedge a(i) \geq a(j))\big)\end{aligned} \tag{3.2}$$

Even though this monotonicity condition is not the only safe choice for $\mathcal{M}(i,j)$, some precondition like $\forall i,j:C\ \mathcal{M}(i,j)$ is necessary, because car control is unsafe if the cars start with incompatible velocities or acceleration choices initially. In fact, we may suspect that a corresponding condition like this may have to hold all the time for the system to remain safe. The car controllers will thus have to make sure they maintain $\forall i,j:C\ \mathcal{M}(i,j)$ always. And formal verification will have to make sure that formula (3.1) is actually valid for the appropriate choices of *DCCS*.

How do we design the distributed hybrid system *DCCS* that satisfies the QdL formula (3.1)? What is an appropriate model for distributed hybrid systems? How can we then prove that (3.1) is true? Next, we introduce QHPs as a general model for distributed hybrid systems and then discuss possible choices of QHPs for *DCCS*. The reader should note that more sophisticated combinations of nested quantifiers and modalities are possible with QdL as well.

3.2. **Quantified Hybrid Programs.** As a formal model for distributed hybrid systems, we introduce *quantified hybrid programs* (QHPs). These are regular programs from dynamic logic [HKT00] to which we add quantified assignments and quantified differential equation systems for *distributed* hybrid dynamics. From these quantified assignments and quantified differential equations, QHPs are built like a Kleene algebra with tests [Koz97].

Definition 3.2. (Quantified hybrid programs).QHPs are defined by the following grammar (α,β are QHPs, i a variable of sort C, f is a function symbol, $\vec{s}$ is a vector of terms with sorts compatible to the arguments of f, θ is a term with sort compatible to the result of f, and χ is a formula of many-sorted first-order logic):

$$\alpha,\beta \;::=\; \forall i:C\ f(\vec{s}) := \theta \mid \forall i:C\ f(\vec{s})' = \theta \,\&\, \chi \mid ?\chi \mid \alpha \cup \beta \mid \alpha;\beta \mid \alpha^*$$

In order to simplify technical difficulties, we impose regularity assumptions on $f(\vec{s})$ in quantified assignments and quantified differential equations. We assume $\vec{s}$ to be either a vector of length 0 or that the mapping from the quantified variable i to $\vec{s}$ is *injective*. That is, each value of $\vec{s}$ can be exhibited by at most one choice of i. A system is injective, e.g., when at least one component of $\vec{s}$ is the quantified variable i. These assumptions can be relaxed, but are sufficient for our purposes; see Section 4 for a discussion of injectivity. For quantified differential equations, we further assume that f is an $\mathbb{R}$-valued function symbol so that derivatives can be defined.

Quantified State Change. The effect of *quantified assignment* $\forall i:C\ f(\vec{s}) := \theta$ is an instantaneous discrete jump assigning θ to $f(\vec{s})$ simultaneously for all objects i of sort C. Hence all $f(\vec{s})$ that are affected by $\forall i:C\ f(\vec{s}) := \theta$ will change their value to the respective θ simultaneously for all choices of i in a single discrete instant of time. Usually, i occurs in term θ, but does not have to. The effect of *quantified differential equation* $\forall i:C\ f(\vec{s})' = \theta \,\&\, \chi$ is a continuous evolution where, for all objects i of sort C, all differential equations $f(\vec{s})' = \theta$ hold at the same time and formula χ holds throughout the

evolution (the state always remains in the region described by χ, i.e., the evolution stops at any arbitrary time before it leaves χ). Again, i usually occurs in term θ. For the trivial evolution domain restriction $\chi \equiv \mathit{true}$, which is always satisfied, we also write $\forall i:C\ f(\vec{s})' = \theta$ instead of $\forall i:C\ f(\vec{s})' = \theta \,\&\, \mathit{true}$.

The dynamics of QHPs changes the interpretation of terms over time: $f(\vec{s})'$ is intended to denote the derivative of the interpretation of the term $f(\vec{s})$ over time during continuous evolution, not the derivative of $f(\vec{s})$ by its argument $\vec{s}$. For $f(\vec{s})'$ to be defined, we assume f is an $\mathbb{R}$-valued function symbol. Although our approach can be extended, we assume that f does not occur in $\vec{s}$. The most common choice of $\vec{s}$ in quantified assignments and quantified differential equations is just i. Other choices are possible for $\vec{s}$, e.g., $\vec{s} = (i, f(i))$ in $\forall i:C\ d(i, f(i)) := \frac{1}{2}a(i) + \frac{1}{2}a(f(i))$. The latter QHP could be used to model that, for each car i, the average acceleration of a car i and its follower $f(i)$ is assigned to a data field $d(i, f(i))$ that car i and its follower use to determine their safe distance.

Time itself is not special but implicit. If a clock variable t is needed in a QHP, it can be axiomatized by $t' = 1$, which is equivalent to $\forall i:C\ t' = 1$ where i does not occur in t. For such *vacuous quantification* (i does not occur anywhere), we may omit $\forall i:C$ from assignments and differential equations, which are then classical assignments and ordinary differential equations. Similarly, we may omit vectors $\vec{s}$ of length 0.

Regular Programs. The *test* action $?\chi$ is used to define conditions. Its effect is that of a *no-op* if the formula χ is true in the current state; otherwise, like *abort*, it allows no transitions. That is, if the test succeeds because formula χ holds in the current state, then the state does not change, and the system execution continues normally. If the test fails because formula χ does not hold in the current state, then the system execution cannot continue, is cut off and not considered any further.

The nondeterministic choice $\alpha \cup \beta$, sequential composition $\alpha;\beta$, and nondeterministic repetition α^* of programs are as in regular expressions but generalized to a semantics in distributed hybrid systems. *Nondeterministic choice* $\alpha \cup \beta$ is used to express behavioral alternatives between the transitions of α and β. That is, the QHP $\alpha \cup \beta$ can choose nondeterministically to follow the transitions of QHP α, or, instead, to follow the transitions of QHP β. The *sequential composition* $\alpha;\beta$ says that the QHP β starts executing after QHP α has finished (β never starts if α does not terminate). In $\alpha;\beta$, the transitions of α take effect first, until α terminates (if it does), and then β continues. Observe that, like repetitions, continuous evolutions within α can take more or less time, which causes uncountable nondeterminism. This nondeterminism is inherent in distributed hybrid systems, because they can operate in so many different ways, which is as such reflected in QHPs. *Nondeterministic repetition* α^* is used to express that the QHP α repeats any number of times, including zero times. When following α^*, the transitions of QHP α can be repeated over and over again, any nondeterministic number of times ($\geq$0).

QHPs (with their semantics and our proof rules) can be extended to systems of quantified differential equations, systems of simultaneous assignments to multiple functions f, g, and statements with multiple quantifiers ($\forall i:C\ \forall j:D\ \ldots$). This includes the quantified differential equation system $\forall i:C\ (x(i)' = v(i), v(i)' = a(i))$, which we can understand as a second-order quantified differential equation $\forall i:C\ (x(i)'' = a(i))$ or as a vectorial first-order quantified differential equation $\forall i:C\ \vec{z}(i)' = \vec{\theta}$ with $\vec{z}(i) = (x(i), v(i))$ and $\vec{\theta} = (v(i), a(i))$; see [Pla08a] for details on how to handle vectorial differential equations. It is similarly simple to extend our approach to quantified assignments with multiple function symbols

like $\forall i:C\ (a(i) := a(i)+1, t(i) := 0)$, which is a vectorial extension that can be handled like parallel updates in programs [BP06]. Our approach can also be extended to multiple quantifiers like in the quantified differential equation $\forall i:C\ \forall j:D\ f(i,j)' = a(i) - d(i,j)$ or the quantified assignment $\forall i:C\ \forall j:D\ d(i,j) := d(i,j) + a(i) + 1$. These quantifier blocks correspond to $\forall \vec{i}:\vec{C}$ with a vectorial variable $\vec{i}$ and a vectorial sort $\vec{C}$. Since these simple vectorial extensions [Pla08a, BP06] are a diversion from the logical essence of our approach, we simplify notation and do not consider these cases formally.

Example. Continuous movement of position $x(i)$ of car i with acceleration $a(i)$ is expressed by differential equation $x(i)'' = a(i)$, which corresponds to the first-order differential equation system $x(i)' = v(i), v(i)' = a(i)$ where $v(i)$ is the velocity of car i. Simultaneous movement of all cars with their respective accelerations $a(i)$ is expressed by the quantified differential equation $\forall i:C\ (x(i)'' = a(i))$ where quantifier $\forall i:C$ ranges over all cars, such that all cars co-evolve along their respective differential equations at the same time.

In addition to continuous dynamics, cars have discrete control. In the following QHP, discrete and continuous dynamics interact (repeatedly because of the * repetition operator):

$$\big(\forall i:C\ (a(i) := \mathsf{if}\ \forall j:C\ far(i,j)\ \mathsf{then}\ a\ \mathsf{else}\ -b\ \mathsf{fi});\ \ \forall i:C\ (x(i)'' = a(i))\big)^* \tag{3.3}$$

First, all cars i control their acceleration $a(i)$. Each car i chooses maximum acceleration $a \geq 0$ for $a(i)$ if its distance to all other cars j is far enough (some condition $far(i,j)$). Otherwise, i chooses full braking $-b < 0$. After all accelerations have been set, all cars move continuously along $\forall i:C\ (x(i)'' = a(i))$. Accelerations may change repeatedly, because the repetition operator * can repeat the QHP when the continuous evolution stops at any time.

Note that the presence of the function argument i in $x(i), v(i), a(i)$ is a decisive difference when comparing the QHP in (3.3) to hybrid systems and when comparing the QdℒC formula in (3.1) to hybrid systems properties. In hybrid systems, we are limited to using variables x, v, a of a single car. If we want to add a second car to a hybrid system model, new state variables y, w, c, new dynamics $y' = w, w' = c$, and new control need to be added for the second car. We can keep on adding any fixed finite number of state variables that way, but we need to know exactly how many cars there are on the street. This does not work when we want to model and verify situations with arbitrarily many cars or in distributed car control scenarios like Fig. 1, where new cars appear or disappear during the evolution of the system. A quantified differential equation like $\forall i:C\ (x(i)' = v(i), v(i)' = a(i))$, for example, cannot be expressed in hybrid systems, because we do not know how many cars i ranges over. If i did range over exactly 3 cars, called 1, 2, and 3, we could replace it by

$$x(1)' = v(1), v(1)' = a(1), x(2)' = v(2), v(2)' = a(2), x(3)' = v(3), v(3)' = a(3)$$

and change notation to obtain primitive state variables $x_1, v_1, a_1, x_2, v_2, a_2, x_3, v_3, a_3$ in an ordinary differential equation system

$$x_1' = v_1, v_1' = a_1, x_2' = v_2, v_2' = a_2, x_3' = v_3, v_3' = a_3$$

But this replacement does not work unless we know exactly how many cars are in the system. Even for systems with a fixed known but large number of participants, such flat representations as (non-distributed) hybrid systems are inefficient, because the system dimension is exponential in the number of participants and all reasoning needs to be repeated for each participant, or even for each pair of participants (collision freedom requires each pair of cars to remain safely separated). This is why we benefit from studying distributed hybrid systems.

One remaining issue with QHP (3.3) is that cars could still move backwards by braking long enough. But this does not capture braking. In order to say that cars can accelerate or brake but may never move backwards, we refine QHP (3.3) to the following QHP in which the evolution domain of the quantified differential equation is restricted (by &) to stay in the region $v(i) \geq 0$ where each car i has a nonnegative velocity:

$$\big(\forall i:C\ a(i) := (\mathsf{if}\ \forall j:C\ far(i,j)\ \mathsf{then}\ a\ \mathsf{else}\ \mathsf{if}\ v(i)>0\ \mathsf{then}\ -b\ \mathsf{else}\ 0\ \mathsf{fi}\ \mathsf{fi});\ \forall i:C\ (x(i)'=v(i), v(i)'=a(i)\,\&\,v(i)\geq 0)\big)^*$$

Observe that this controller is also smarter about the acceleration choices of cars than that in (3.3). It will choose 0 for $a(i)$ if car i does not move ($v(i) = 0$) but car i cannot accelerate safely either, because not all cars j are far enough away.

System Structure. The *communication model* that Qd$\mathcal{L}$ supports is that of shared variable communication. Suppose a car i has direct control over the acceleration of car j. Then, when i decides to brake, it could directly change the acceleration of car j as well using the QHP $a(j) := a(j) - 2$. In most system designs, control variables of other agents are not directly accessible but communication has to be used instead. In Qd$\mathcal{L}$, communication can be implemented by assigning to shared variables (delays in communication are easy to model by combining assignments with differential equations). Suppose $s(i)$ is the data field that car i queries periodically to track how much distance it is supposed to maintain relative to its leader car. Then the QHP $\forall i:C\ s(f(i)) := s(f(i)) + 10$ would cause each car i to tell its respective follower car $f(i)$ to increase the safety distance $s(i)$ by 10, e.g., when the road conditions are slippery.

Shared (first-order) variables are sufficient to model *discrete structural dynamics*, e.g., of changing communication links. If, for example, the car $f(i)$ following car i has left the street, car i may update its communication link to reflect this change in the structure of the system by running the QHP $f(i) := f(f(i))$ that updates the follower of i to the follower of $f(i)$, i.e., the follower of the follower of i. Other discrete structural changes in the system and communication patterns as well as all data structures can be modeled easily, since a complete object-oriented programming language [BP06] can be defined in Qd$\mathcal{L}$. Shared (first-order) variables are sufficient to model *continuous structural dynamics*, since structural changes in the continuous dynamics can be modeled by quantified differential equations that change their connectivity, i.e., which parts of the quantified differential equation depend on which other parts. For example, in QHP $\forall i:C\ (x(i)'' = a(i) + c(i,f(i))a(f(i)))$ the connectivity term $c(i, f(i))$ models whether or not the follower $f(i)$ of car i has physical bumper-to-bumper contact with car i, such that the acceleration $a(f(i))$ of car $f(i)$ also pushes car i forwards, not just car $f(i)$. The change of $c(i, f(i))$ from zero to non-zero represents a structural change in the physical dynamics structurally, because it structurally changes the effect of the continuous dynamics.

These examples illustrate how the discrete dynamics, continuous dynamics, and discrete and continuous structural dynamics of distributed hybrid systems with an arbitrary parametric number of participants can be modeled as a QHP. We defer the explanation of dimensional dynamics, i.e., dynamic appearance and disappearance of agents, to Section 5.

4. Semantics

The QdL semantics is a *constant domain Kripke semantics* [FM99] *with first-order structures as states* that associate total functions of appropriate type with function symbols. In constant domain, all states share the same domain for quantifiers. In particular, we choose to represent object creation not by changing the domain of states, but by changing the interpretation of the createdness flag $\mathbb{E}(i)$ of the object denoted by i. With $\mathbb{E}(i)$, object creation is definable in a modular way (as we elaborate in Section 5).

States. A *state* σ associates an (infinite) set $\sigma(C)$ of objects with each sort C, and it associates a function $\sigma(f)$ of appropriate type with each function symbol f, including $\mathbb{E}(\cdot)$. We assume $\mathbb{E}(\cdot)$ to have (unbounded but) finite support, i.e., each state only has a finite number of positions i at which $\mathbb{E}(i) = 1$. This makes sense in practice, because there is a varying and possibly large but still finite numbers of participants (e.g., cars). For simplicity, σ also associates a value $\sigma(i)$ of appropriate type with each variable i. The domain of $\mathbb{R}$ and the interpretation of $0, 1, +, -, \cdot$ is that of real arithmetic. We assume *constant domain* for each sort C: all states σ, τ share the same domains $\sigma(C) = \tau(C)$ for C. Sorts $C \neq D$ are disjoint: $\sigma(C) \cap \sigma(D) = \emptyset$. The set of all states is denoted by $\mathcal{S}$. The state σ_i^e agrees with σ except for the interpretation of variable i, which is changed to e.

Formulas. We use $\sigma[\![\theta]\!]$ to denote the value of term θ at state σ, which is defined as in first-order logic. Especially, $\sigma_i^e[\![\theta]\!]$ denotes the value of θ in state σ_i^e, i.e., in state σ with i interpreted as e. Further, $\rho(\alpha) \subseteq \mathcal{S} \times \mathcal{S}$ denotes the state transition relation of QHP α, which we define below.

Definition 4.1. (Semantics of QdL).The *interpretation* $\sigma \models \phi$ of QdL formula ϕ with respect to state σ is defined inductively as:

(1) $\sigma \models (\theta_1 = \theta_2)$ iff $\sigma[\![\theta_1]\!] = \sigma[\![\theta_2]\!]$; accordingly for $\geq$ (greater or equal).
(2) $\sigma \models \phi \wedge \psi$ iff $\sigma \models \phi$ and $\sigma \models \psi$; accordingly for $\neg$ (not).
(3) $\sigma \models \forall i : C\ \phi$ iff $\sigma_i^e \models \phi$ for all objects $e \in \sigma(C)$.
(4) $\sigma \models \exists i : C\ \phi$ iff $\sigma_i^e \models \phi$ for some object $e \in \sigma(C)$.
(5) $\sigma \models [\alpha]\phi$ iff $\tau \models \phi$ for all states τ with $(\sigma, \tau) \in \rho(\alpha)$.
(6) $\sigma \models \langle\alpha\rangle\phi$ iff $\tau \models \phi$ for some τ with $(\sigma, \tau) \in \rho(\alpha)$.

We say that ϕ is true at σ if $\sigma \models \phi$. QdL formula ϕ is *valid*, written $\vDash \phi$, iff $\sigma \models \phi$ for all σ.

Programs. QHPs have a compositional semantics. The semantics of a QHP is its reachability relation.

Definition 4.2. (Transition semantics of QHP).The *transition relation,* $\rho(\alpha) \subseteq \mathcal{S} \times \mathcal{S}$, of QHP α specifies which state $\tau \in \mathcal{S}$ is reachable from $\sigma \in \mathcal{S}$ by running QHP α. It is defined inductively:

(1) $(\sigma, \tau) \in \rho(\forall i : C\ f(\vec{s}) := \theta)$ iff state τ is identical to σ except that at each position $\vec{o}$ of f: if $\sigma_i^e[\![\vec{s}]\!] = \vec{o}$ for some object $e \in \sigma(C)$, then $\tau(f)\big(\sigma_i^e[\![\vec{s}]\!]\big) = \sigma_i^e[\![\theta]\!]$. If there are multiple objects e giving the same position $\sigma_i^e[\![\vec{s}]\!] = \vec{o}$, then all of the resulting states τ are reachable.
(2) $(\sigma, \tau) \in \rho(\forall i : C\ f(\vec{s})' = \theta\,\&\,\chi)$ iff there is a function $\varphi{:}[0, r] \to \mathcal{S}$ for some $r \geq 0$ with $\varphi(0) = \sigma$ and $\varphi(r) = \tau$ satisfying the following conditions. At each time $t \in [0, r]$, state $\varphi(t)$ is identical to σ, except that at each position $\vec{o}$ of f: if $\sigma_i^e[\![\vec{s}]\!] = \vec{o}$ for some object $e \in \sigma(C)$, then, at each time $\zeta \in [0, r]$:

- All differential equations hold and corresponding derivatives exist (trivial for $r=0$):
$$\frac{\mathsf{d}\,(\varphi(t)_i^e[\![f(\vec{s})]\!])}{\mathsf{d}t}(\zeta) = (\varphi(\zeta)_i^e[\![\theta]\!])$$
- The evolution domain is respected: $\varphi(\zeta)_i^e \models \chi$.

If there are multiple objects e giving the same position $\sigma_i^e[\![\vec{s}]\!] = \vec{o}$, then all of the resulting states τ are reachable.

(3) $\rho(?\chi) = \{(\sigma,\sigma) : \sigma \models \chi\}$
(4) $\rho(\alpha \cup \beta) = \rho(\alpha) \cup \rho(\beta)$
(5) $\rho(\alpha;\beta) = \rho(\beta) \circ \rho(\alpha) = \{(\sigma,\tau) : (\sigma,z) \in \rho(\alpha) \text{ and } (z,\tau) \in \rho(\beta) \text{ for a state } z\}$
(6) $(\sigma,\tau) \in \rho(\alpha^*)$ iff there is an $n \in \mathbb{N}$ with $n \geq 0$ and there are states $\sigma = \sigma_0, \ldots, \sigma_n = \tau$ such that $(\sigma_i, \sigma_{i+1}) \in \rho(\alpha)$ for all $0 \leq i < n$.

The semantics is *explicit change*: nothing changes unless an assignment or differential equation specifies how. In cases 1–2, only f changes and only at positions of the form $\sigma_i^e[\![\vec{s}]\!]$ for some interpretation $e \in \sigma(C)$ of i. If there are multiple such e that affect the same position $\vec{o}$, any of those changes can take effect by a nondeterministic choice. QHP $\forall i:C\ x := a(i)$ may change x to *any* $a(i)$. Hence, $[\forall i:C\ x := a(i)]\phi(x) \equiv \forall i:C\ \phi(a(i))$, because that modality considers *all* possibilities of changing x to *any* $a(i)$. In contrast, $\langle \forall i:C\ x := a(i)\rangle\phi(x) \equiv \exists i:C\ \phi(a(i))$, because that modality considers *some* possibility of changing x to *any* $a(i)$. Similarly, x can evolve along $\forall i:C\ x' = a(i)$ with any of the slopes $a(i)$. But evolutions cannot start with slope $a(c)$ and then switch to a different slope $a(d)$ later. Any choice for the quantified variable i is possible but i remains unchanged during each evolution.

We call a quantified assignment $\forall i:C\ f(\vec{s}) := \theta$ or a quantified differential equation $\forall i:C\ f(\vec{s})' = \theta \,\&\, \chi$ *injective* iff there is at most one e satisfying cases 1–2. For injective quantified assignments and injective quantified differential equations, conditions 1–2 can be simplified as follows:

(1′) $(\sigma,\tau) \in \rho(\forall i:C\ f(\vec{s}) := \theta)$ iff state τ is identical to σ except that for each $e \in \sigma(C)$: $\tau(f)\big(\sigma_i^e[\![\vec{s}]\!]\big) = \sigma_i^e[\![\theta]\!]$.
(2′) $(\sigma,\tau) \in \rho(\forall i:C\ f(\vec{s})' = \theta \,\&\, \chi)$ iff there is a function $\varphi{:}[0,r] \to \mathcal{S}$ for some $r \geq 0$ with $\varphi(0) = \sigma$ and $\varphi(r) = \tau$ such that for each $e \in \sigma(C)$ and each time $\zeta \in [0,r]$:
- All differential equations hold and corresponding derivatives exist (trivial for $r=0$):
$$\frac{\mathsf{d}\,(\varphi(t)_i^e[\![f(\vec{s})]\!])}{\mathsf{d}t}(\zeta) = (\varphi(\zeta)_i^e[\![\theta]\!])$$
- The evolution domain is respected: $\varphi(\zeta)_i^e \models \chi$.

We call quantified assignments and quantified differential equations *schematic* iff $\vec{s}$ is i (thus injective) and the only arguments to function symbols in θ are i. Schematic quantified differential equations like $\forall i:C\ f(i)' = a(i) \,\&\, \chi$ are very common, because distributed hybrid systems often have a family of similar differential equations replicated for multiple participants i. Their synchronization often comes from discrete communication on top of their continuous dynamics. Physically coupled differential equations are possible as well. They correspond to continuous physical interactions, e.g., if a car bumps into another car from the side, it radically changes the structure of the differential equations that determine its movement. Either case can be represented in QHPs, even if the schematic case is more common.

Cases 1–2 can be defined accordingly for vectorial extensions. These vectorial extensions are simple, just notationally cumbersome. For quantified assignments to multiple function symbols like in $\forall i:C\ (f(\vec{s}):=\theta, g(\vec{t}):=\vartheta)$ all changes to f and g according to case 1 are performed simultaneously when transitioning from state σ to τ [Pla08a, Pla10a, Rüm06]. The only difference to the sequential composition $(\forall i:C\ f(\vec{s}):=\theta);\ (\forall i:C\ g(\vec{t}):=\vartheta)$ is that in the quantified assignment to multiple functions, the change is simultaneous, hence $\vec{t}$ and ϑ are evaluated in the original state σ, not in the intermediate state that is reached after f has already been modified by $\forall i:C\ f(\vec{s}):=\theta$. For quantified differential equation systems with multiple function symbols like in $\forall i:C\ (Df(\vec{s})=\theta, g(\vec{t})'=\vartheta\,\&\,\chi)$ the changes to f and g according to case 2 are again simultaneous and all differential equations of the differential equation system need to hold at the same time. Multiple quantifiers like $\forall i:C\ \forall j:D$ in the quantified differential equation and quantified assignment are vectorial, i.e., "for some object $e\in\sigma(C)$" in cases 1–2 is replaced by "for some object $e\in\sigma(C)$ and some object $c\in\sigma(D)$", which are for i and j, respectively. That is, we replace σ_i^e with σ_{ij}^{ec} and $\varphi(t)_i^e$ with $\varphi(t)_{ij}^{ec}$ as well as $\varphi(\zeta)_i^e$ with $\varphi(\zeta)_{ij}^{ec}$ in cases 1–2.

Note that existence/uniqueness theorems for solutions of differential equations [Wal98] carry over to quantified differential equations. In particular, existence/uniqueness of solutions by Picard-Lindelöf / Cauchy-Lipschitz theorem [Wal98, Theorem 10.VI] and by Peano theorem [Wal98, Theorem 10.IX] carry over to case 2 of the semantics $\rho(\alpha)$ if it only affects a finite subdomain of $\sigma(C)$, because the quantifier then corresponds to a finite set of classical differential equations. (The number of differential equations may still change dynamically over time, though, so that the quantified differential equation system *cannot* be replaced with an unquantified differential equation system in the QHP). For infinite $\sigma(C)$, the theorems carry over to schematic $\forall i:C\ f(i)'=\theta\,\&\,\chi$, which give an (infinite) set of disconnected classical differential equations. In all these cases, Picard-Lindelöf's theorem implies that the solution is unique, when terms are continuously differentiable (on the open domain where divisors are non-zero). For an overview of results about general infinite-dimensional differential equations, see [Bog95].

5. Actual Existence and Object Creation

Up to now, we have been neglecting the effects of object creation and just pretended that the domain of objects would never change. In this section, we consider object creation and distinguish objects that actually exist physically from those that have not been created yet (or are not physically present in the part of the world reflected in the model). We will see that this distinction does not require any change of QdL. It is just a conceptual change of our understanding.

Actual Existence. For the QdL semantics, we chose constant domain semantics, i.e., all states share the same domains. Thus quantifiers range over all possible objects (*possibilist quantification* in constant domain semantics) not just over active existing objects (*actualist quantification* in varying domain semantics) [FM99]. In order to distinguish between *actual objects* that exist in a state, because they have already been created and can now actively take part in its evolution, versus *possible objects* that still passively await creation, we use function symbol $\mathsf{E}(\cdot)$. Function symbol $\mathsf{E}(\cdot)$ is similar to existence predicates in first-order modal logic [FM99], except that its value can be assigned to in QHPs.

Object Creation. For a term i of type $C \neq \mathbb{R}$, we use $\mathsf{E}(i) = 1$ to represent that the object denoted by i has been created and actually exists. We use $\mathsf{E}(i) = 0$ to represent that i has not been created or does not exist any longer. Object creation amounts to changing the interpretation of $\mathsf{E}(i)$. For an object denoted by i that has not been created ($\mathsf{E}(i) = 0$), object creation corresponds to the state change caused by assignment $\mathsf{E}(i) := 1$. With quantified assignments and function symbols, *object creation* is definable by a QHP:

$$n := \mathsf{new}\, C \;\equiv\; (\forall j : C\ n := j);\ ?(\mathsf{E}(n) = 0);\ \mathsf{E}(n) := 1 \tag{5.1}$$

This QHP assigns an arbitrary j of type C to n ($\forall j : C\ n := j$) that did not exist before (subsequent test $?\mathsf{E}(n) = 0$) and adjusts existence ($\mathsf{E}(n) := 1$). *Disappearance* of object i corresponds to $\mathsf{E}(i) := 0$. Our choice of constant domain semantics avoids semantic subtleties of varying domains about the meaning of free variables denoting non-existent objects as in free logics [FM99]. Denotation is standard in QdL. Terms may just denote objects that have not been activated yet. This is even useful to initialize new objects (e.g., $x(n) := 8$) before activation ($\mathsf{E}(n) := 1$).

Actualist Quantifiers. We define abbreviations for *actualist quantifiers* in formulas, quantified assignments, and quantified differential equations that range only over previously *created objects*, similar to relativization in modal logic [FM99] by masking:

$$\begin{aligned}
\forall i : C!\ \phi &\equiv \forall i : C\ (\mathsf{E}(i) = 1 \to \phi)\\
\exists i : C!\ \phi &\equiv \exists i : C\ (\mathsf{E}(i) = 1 \wedge \phi)\\
\forall i : C!\ f(\vec{s}) := \theta &\equiv \forall i : C\ f(\vec{s}) := (\mathsf{if}\, \mathsf{E}(i) = 1\, \mathsf{then}\, \theta\, \mathsf{else}\, f(\vec{s})\, \mathsf{fi})\\
\forall i : C!\ f(\vec{s})' = \theta &\equiv \forall i : C\ f(\vec{s})' = (\mathsf{if}\, \mathsf{E}(i) = 1\, \mathsf{then}\, \theta\, \mathsf{else}\, 0\, \mathsf{fi}) \;\equiv\; \forall i : C\ f(\vec{s})' = \mathsf{E}(i)\theta
\end{aligned}$$

The first two cases define quantifiers for actually existing objects. The last two cases define quantified state change for actually existing objects using conditional terms that choose effect θ if $\mathsf{E}(i) = 1$ and choose no effect, retaining the old value $f(\vec{s})$ or evolving with slope 0, if $\mathsf{E}(i) = 0$. The conditional terms can be avoided as indicated in the last column of the last row (similarly for quantified assignments). In all cases, the notation $C!$ signifies that the quantifier domain is restricted to actually existing objects of type C. Hence, $\forall i : C$ ranges over all objects of sort C, existent or not, whereas $\forall i : C!$ ranges only over those objects of sort C that actually exist in the current state.

We generally assume that QHPs involve only quantified assignments and differential equations that are restricted to created objects, because real systems only affect objects that are physically present, not those that will be created later. We still treat actualist quantification over $C!$ as a defined notion, in order to simplify the semantics and proof calculus by separating object creation from quantified state change rules in a modular way.

If only finitely many objects have been created in the initial state (say 0), then it is easy to see that only finitely many new objects will be created with finitely many such QHP transitions, because each quantified state change for $C!$ only ranges over a finite domain then. Recall that we assume $\mathsf{E}(\cdot)$ to have *(unbounded but) finite support*, i.e., each state only has a finite number of positions i at which $\mathsf{E}(i) = 1$. This makes sense in practice, because there is a varying and possibly large but still finite numbers of participants (e.g., cars).

Example. The car control examples in Section 3 were unaware of the distinction between actual existing and possible objects. Car control, of course, only affects created cars that are physically present, not the possible cars that have not been built yet or that are not present yet. To reflect this, the dynamics and properties, we only need to replace each occurrence of $\forall i : C$ with $\forall i : C!$ in the car control examples of Section 3. For instance the QdℒC formula (3.1) will be restricted to actual cars by adding $C!$ as follows:

$$(\forall i, j : C!\ \mathcal{M}(i,j)) \rightarrow [DCCS]\ \forall i{\neq}j : C!\ x(i){\neq}x(j) \tag{5.2}$$

In the precondition, we only demand that all cars that actually exist $(\forall i, j : C!\ \ldots)$ start from compatible positions with compatible velocities and accelerations, because we do not care about non-existent cars. In the postcondition, we only guarantee that all existing cars are at different positions, because we cannot really say what happens with cars that do not yet exist and that are beyond our control. The controller and dynamics in the QHP *DCCS* can be restricted to actual cars in the same way, e.g., in the following variant of (3.3):

$$\big(\forall i : C!\ (a(i) := \mathsf{if}\ \forall j : C!\ far(i,j)\ \mathsf{then}\ a\ \mathsf{else}\ -b\ \mathsf{fi});\ \ \forall i : C!\ (x(i)'' = a(i))\big)^* \tag{5.3}$$

Except conceptually, this restriction to created cars does not really affect the specification (nor its verification). This gets much more involved as soon as we create new objects at runtime or let them disappear again. When we create a new car that joins the system, or when a new car appears from an on-ramp (Fig. 1), then one more set of positions $x(n)$, velocities $v(n)$, and accelerations $a(n)$ comes out of nowhere and starts evolving along with the distributed car control dynamics. That new car n has not even been considered in the dynamics before it has been created. A real system cannot control what is not part of the system yet and thus must deal with new agents dynamically whenever they arrive.

A fairly challenging feature of distributed car control, thus, is that new cars may appear dynamically from on-ramps (Fig. 1) changing the set of active objects dynamically at runtime. To model this, we consider the following QHP:

$$DCCS\ \equiv\ (n := \mathsf{new}\, C; ?\forall i : C!\ \mathcal{M}(i,n); \forall i : C!\ (x(i)'' = a(i)))^* \tag{5.4}$$

Before following the continuous dynamics, this QHP creates a new car n at an arbitrary position $x(n)$ satisfying compatibility condition $\mathcal{M}(i,n)$ with respect to all other created cars i. Hence *DCCS* allows new cars to appear, but not drop right out of the sky in front of a fast car or run at the speed of light only 2 meters away. When cars appear into the horizon from on-ramps, this condition captures that a car is only allowed to join the lane ("appear" into the model world) if it cannot cause a crash with other existing cars (Fig. 1). Unboundedly many cars may appear during the operation of *DCCS* and change the system dimension arbitrarily, because of the repetition operator *.

DCCS is simple but shows how properties of distributed hybrid systems can be expressed in QdℒC. Joint dynamics of multiple components corresponds to compositions of quantified differential equation systems, quantified assignments, and object (dis)appearance. Structural dynamics corresponds to assignments to function terms. Say, $f(i)$ is the car registered by communication as the car following car i. Then a term $d(i, f(i))$, which denotes the minimum safety distance negotiated between car i and its follower, is a crucial part of the system dynamics. Restructuring the system in response to lane change corresponds to assigning a new value to $f(i)$, which impacts the value of $d(i, f(i))$ in the system dynamics.

$$([\cup])\ \frac{[\alpha]\phi \wedge [\beta]\phi}{[\alpha \cup \beta]\phi} \qquad ([;])\ \frac{[\alpha][\beta]\phi}{[\alpha;\beta]\phi} \qquad ([?])\ \frac{\chi \to \psi}{[?\chi]\psi}$$

$$(\langle\cup\rangle)\ \frac{\langle\alpha\rangle\phi \vee \langle\beta\rangle\phi}{\langle\alpha \cup \beta\rangle\phi} \quad (\langle;\rangle)\ \frac{\langle\alpha\rangle\langle\beta\rangle\phi}{\langle\alpha;\beta\rangle\phi} \qquad (\langle?\rangle)\ \frac{\chi \wedge \psi}{\langle?\chi\rangle\psi}$$

$$([']) \frac{\forall t{\geq}0\, \big((\forall 0{\leq}\tilde{t}{\leq}t\, [\forall i : C\ f(\vec{s}) := y_{\vec{s}}(\tilde{t})]\chi) \to [\forall i : C\ f(\vec{s}) := y_{\vec{s}}(t)]\phi\big)}{[\forall i : C\ f(\vec{s})' = \theta\,\&\,\chi]\phi}\,{}_1$$

$$(\langle'\rangle) \frac{\exists t{\geq}0\, \big((\forall 0{\leq}\tilde{t}{\leq}t\, \langle\forall i : C\ f(\vec{s}) := y_{\vec{s}}(\tilde{t})\rangle\chi) \wedge \langle\forall i : C\ f(\vec{s}) := y_{\vec{s}}(t)\rangle\phi\big)}{\langle\forall i : C\ f(\vec{s})' = \theta\,\&\,\chi\rangle\phi}\,{}_1$$

$$([:=]) \frac{\mathsf{if}\, \exists i : C\ \vec{s} = [\mathcal{A}]\vec{u}\, \mathsf{then}\, \forall i : C\ (\vec{s} = [\mathcal{A}]\vec{u} \to \phi(\theta))\, \mathsf{else}\, \phi(f([\mathcal{A}]\vec{u}))\, \mathsf{fi}}{\phi([\forall i : C\ f(\vec{s}) := \theta]f(\vec{u}))}\,{}_2$$

$$(\langle:=\rangle) \frac{\mathsf{if}\, \exists i : C\ \vec{s} = \langle\mathcal{A}\rangle\vec{u}\, \mathsf{then}\, \exists i : C\ (\vec{s} = \langle\mathcal{A}\rangle\vec{u} \wedge \phi(\theta))\, \mathsf{else}\, \phi(f(\langle\mathcal{A}\rangle\vec{u}))\, \mathsf{fi}}{\phi(\langle\forall i : C\ f(\vec{s}) := \theta\rangle f(\vec{u}))}\,{}_2$$

$$([:]) \frac{\Upsilon([\forall i : C\ f(\vec{s}) := \theta]\vec{u})}{[\forall i : C\ f(\vec{s}) := \theta]\, \Upsilon(\vec{u})}\,{}_3 \qquad ([:*])\ \frac{\forall j : C\ \phi(\theta)}{[\forall j : C\ n := \theta]\phi(n)} \qquad (\langle:*\rangle)\ \frac{\exists j : C\ \phi(\theta)}{\langle\forall j : C\ n := \theta\rangle\phi(n)}$$

$$(\mathbb{E})\ \frac{\mathit{true}}{\exists n : C\ \mathbb{E}(n) = 0}$$

$$(\exists\mathrm{r})\ \frac{\Gamma \to \phi(\theta), \exists x : C\ \phi(x), \Delta}{\Gamma \to \exists x : C\ \phi(x), \Delta}\,{}_4 \qquad (\forall\mathrm{r})\ \frac{\Gamma \to \phi(f(X_1, .., X_n)), \Delta}{\Gamma \to \forall x : C\ \phi(x), \Delta}\,{}_5$$

$$(\forall\mathrm{l})\ \frac{\Gamma, \phi(\theta), \forall x : C\ \phi(x) \to \Delta}{\Gamma, \forall x : C\ \phi(x) \to \Delta}\,{}_4 \qquad (\exists\mathrm{l})\ \frac{\Gamma, \phi(f(X_1, .., X_n)) \to \Delta}{\Gamma, \exists x : C\ \phi(x) \to \Delta}\,{}_5$$

$$(\mathrm{i}\forall)\ \frac{\mathrm{QE}(\forall X, Y\ (\mathsf{if}\, \vec{s} = \vec{t}\, \mathsf{then}\, \Phi(X) \to \Psi(X)\, \mathsf{else}\, \Phi(X) \to \Psi(Y)\, \mathsf{fi}))}{\Phi(f(\vec{s})) \to \Psi(f(\vec{t}))}\,{}_6 \quad (\mathrm{i}\exists)\ \frac{\mathrm{QE}(\exists X\ \bigwedge_i(\Phi_i \to \Psi_i))}{\Phi_1 \to \Psi_1\ \ \ldots\ \ \Phi_n \to \Psi_n}\,{}_7$$

$$([]gen)\ \frac{\phi \to \psi}{\Gamma, [\alpha]\phi \to [\alpha]\psi, \Delta} \qquad (\langle\rangle gen)\ \frac{\phi \to \psi}{\Gamma, \langle\alpha\rangle\phi \to \langle\alpha\rangle\psi, \Delta} \qquad (ind)\ \frac{\phi \to [\alpha]\phi}{\Gamma, \phi \to [\alpha^*]\phi, \Delta}$$

$$(con)\ \frac{v > 0 \wedge \varphi(v) \to \langle\alpha\rangle\varphi(v-1)}{\Gamma, \exists v\, \varphi(v) \to \langle\alpha^*\rangle\exists v{\leq}0\, \varphi(v), \Delta}\,{}_8$$

[1] $t, \tilde{t}$ are new logical variables and $y_{\vec{s}}(t)$ the simultaneous solutions of the (injective) differential equations $\forall i : C\ f(\vec{s})' = \theta$ with $f(\vec{s})$ as symbolic initial values.

[2] The occurrence of $f(\vec{u})$ in $\phi(f(\vec{u}))$ is not in scope of a modality (admissible substitution) and we abbreviate assignment $\forall i : C\ f(\vec{s}) := \theta$ by $\mathcal{A}$, which is assumed to be injective.

[3] $f \neq \Upsilon$ and the quantified assignment $\forall i : C\ f(\vec{s}) := \theta$ is injective. The same rule applies for $\langle\forall i : C\ f(\vec{s}) := \theta\rangle$ instead of $[\forall i : C\ f(\vec{s}) := \theta]$.

[4] θ is an arbitrary term of sort C, often a new logical variable X.

[5] f is a new (Skolem) function of appropriate type and $X_1, .., X_n$ are all free logical variables of $\forall x\, \phi(x)$.

[6] X, Y are new logical variables of sort $\mathbb{R}$. QE needs to be applicable to the formula in the premise.

[7] among all branches, the free (existential) logical variable X of sort $\mathbb{R}$ only occurs in the branches $\Phi_i \to \Psi_i$. QE needs to be defined for the formula in the premise, especially, no Skolem dependencies on X occur.

[8] logical variable v does not occur in α.

Figure 2: Rule schemata of the proof calculus for quantified differential dynamic logic.

$$(\neg\mathrm{r})\frac{\Gamma,\phi\to\Delta}{\Gamma\to\neg\phi,\Delta} \quad (\vee\mathrm{r})\frac{\Gamma\to\phi,\psi,\Delta}{\Gamma\to\phi\vee\psi,\Delta} \quad (\wedge\mathrm{r})\frac{\Gamma\to\phi,\Delta \quad \Gamma\to\psi,\Delta}{\Gamma\to\phi\wedge\psi,\Delta} \quad (\to\mathrm{r})\frac{\Gamma,\phi\to\psi,\Delta}{\Gamma\to(\phi\to\psi),\Delta}$$

$$(\neg\mathrm{l})\frac{\Gamma\to\phi,\Delta}{\Gamma,\neg\phi\to\Delta} \quad (\vee\mathrm{l})\frac{\Gamma,\phi\to\Delta \quad \Gamma,\psi\to\Delta}{\Gamma,\phi\vee\psi\to\Delta} \quad (\wedge\mathrm{l})\frac{\Gamma,\phi,\psi\to\Delta}{\Gamma,\phi\wedge\psi\to\Delta} \quad (\to\mathrm{l})\frac{\Gamma\to\phi,\Delta \quad \Gamma,\psi\to\Delta}{\Gamma,(\phi\to\psi)\to\Delta}$$

$$(ax)\frac{}{\Gamma,\phi\to\phi,\Delta} \quad (cut)\frac{\Gamma\to\phi,\Delta \quad \Gamma,\phi\to\Delta}{\Gamma\to\Delta}$$

Figure 3: Propositional rule schemata

6. Proof Calculus

In Fig. 2, we present a proof calculus for QdℒC formulas. The basic principle behind the proof rules is that they transform a QHP into structurally simpler logical formulas by symbolic decomposition. For our purposes, it is sufficient to understand the sequent notation informally, just for a systematic proof structure. With finite sets of formulas for the *antecedent* Γ and *succedent* Δ, *sequent* $\Gamma\to\Delta$ is an abbreviation for the formula $\bigwedge_{\phi\in\Gamma}\phi \to \bigvee_{\psi\in\Delta}\psi$. Our calculus uses standard proof rules for propositional logic with the cut rule; see Fig. 3. The proof rules are used backwards from the *conclusion* (goal below horizontal bar) to the *premises* (subgoals above bar).

In the QdℒC calculus, we use substitutions that take effect within formulas and programs (defined as usual). Only admissible substitutions are applicable, however, which is crucial for soundness. An application of a substitution σ is *admissible* if no replaced term θ occurs in the scope of a quantifier or modality binding a symbol in θ or in its replacement $\sigma\theta$. A modality *binds* a symbol f iff it contains an assignment to f (like $\forall i:C\ f(\vec{s}):=\theta$) or a differential equation containing an $f(\vec{s})'$ (like $\forall i:C\ f(\vec{s})'=\theta$). The substitutions in Fig. 2 that insert a term θ into $\phi(\theta)$ also have to be admissible for the proof rules to be applicable. We explain the QdℒC proof rules in the sequel.

Regular Rules. The first proof rules in Fig. 2 axiomatize sequential compositions ([;],⟨;⟩), nondeterministic choices ([∪],⟨∪⟩), and tests ([?],⟨?⟩) of regular programs as in dynamic logic [HKT00]. Like most other rules in Fig. 2, these rules do not contain sequent symbol →, i.e., they can be applied to any subformula. These rules represent (directed) equivalences: conclusion and premise are equivalent. The equivalences are directed in the sense that we only use them to replace occurrences of the conclusion with the premise (which is structurally simpler), not the other way around.

Nondeterministic choices split into their alternatives ([∪],⟨∪⟩). For rule [∪]: If all α transitions lead to states satisfying ϕ (i.e., $[\alpha]\phi$ holds) and all β transitions lead to states satisfying ϕ (i.e., $[\beta]\phi$ holds), then, all transitions of QHP $\alpha\cup\beta$, which choose between following α and following β, also lead to states satisfying ϕ (i.e., $[\alpha\cup\beta]\phi$ holds). Dually for rule ⟨∪⟩, if there is an α transition to a ϕ state ($\langle\alpha\rangle\phi$) or a β-transition to a ϕ state ($\langle\beta\rangle\phi$), then, in either case, there is a transition of $\alpha\cup\beta$ to ϕ ($\langle\alpha\cup\beta\rangle\phi$ holds), because $\alpha\cup\beta$ can choose which of those transitions to follow. A general principle behind the QdℒC proof rules is most noticeable in [∪],⟨∪⟩: these proof rules symbolically decompose the reasoning into two separate parts and analyze the fragments α and β separately, which makes the problem tractable and is good for scalability. For these symbolic structural decompositions, it is very helpful that QdℒC is a full logic that is closed under all logical operators, including

disjunction and conjunction, for then the premises in $[\cup],\langle\cup\rangle$ are QdL formulas again (unlike in Hoare logic [Hoa69]).

Sequential compositions are proven using nested modalities ($[;],\langle;\rangle$). For rule $[;]$: If after all α-transitions, all β-transitions lead to states satisfying ϕ (i.e., $[\alpha][\beta]\phi$ holds), then also all transitions of the sequential composition $\alpha;\beta$ lead to states satisfying ϕ (i.e., $[\alpha;\beta]\phi$ holds). The dual rule $\langle;\rangle$ uses the fact that if there is an α-transition, after which there is a β-transition leading to ϕ (i.e., $\langle\alpha\rangle\langle\beta\rangle\phi$), then there is a transition of $\alpha;\beta$ leading to ϕ (that is, $\langle\alpha;\beta\rangle\phi$), because the transitions of $\alpha;\beta$ are just those that first do any α-transition, followed by any β-transition (Section 4). Again, it is crucial that QdL is a full logic that considers reachability statements as modal operators, which can be nested, for then the premises in $[;],\langle;\rangle$ are QdL formulas again (unlike in Hoare logic [Hoa69]).

Tests are proven by assuming (with an implication in rule [?]) or showing (with a conjunction in rule $\langle?\rangle$) that the test succeeds, because test $?\chi$ can only make a transition when condition χ actually holds true (Section 4). Thus, for QdL formula $\langle?\chi\rangle\phi$, rule $\langle?\rangle$ is used to prove that formula χ holds true (otherwise there is no transition and thus the reachability property is false) and that formula ϕ holds after the resulting no-op. Dually, rule [?] for QdL formula $[?\chi]\phi$ assumes that formula χ holds true (otherwise there is no transition and thus nothing to show) and shows that ϕ holds after the resulting no-op.

Quantified Differential Equations. Rules $[']$,$\langle'\rangle$ handle continuous evolutions for quantified differential equations with first-order definable solutions. Given a solution for the quantified differential equation system with symbolic initial values $f(\vec{s})$, continuous evolution along differential equations can be replaced with a quantified assignment $\forall i:C\ f(\vec{s}) := y_{\vec{s}}(t)$ corresponding to the simultaneous solution (of the differential equations $\forall i:C\ f(\vec{s})' = \theta$ with $f(\vec{s})$ as symbolic initial values) and an additional quantifier for the evolution time t. In rule $[']$, postcondition ϕ needs to hold *for all* evolution durations $t \geq 0$. In rule $\langle'\rangle$, it needs to hold after *some* duration $t \geq 0$. The constraint on χ restricts the continuous evolution such that its solution $f(\vec{s}) := y_{\vec{s}}(\tilde{t})$ remains in the evolution domain region χ at all intermediate times $\tilde{t} \leq t$. This constraint simplifies to *true* if χ is *true*.

For schematic cases like $\forall i:C\ f(i)' = a(i)$, first-order definable solutions can be obtained by adding argument i to first-order definable solutions of the deparametrized version $f' = a$. For example, the following proof step uses rule $[']$ to turn a quantified differential equation system into a quantified assignment with an extra quantifier for the duration t of the evolution.

$$\frac{\forall i{\neq}j\ x(i){\neq}x(j) \rightarrow \forall t{\geq}0\,[\forall i\, x(i) := -\tfrac{b}{2}t^2 + v(i)t + x(i)]\,\forall j{\neq}k\ x(j){\neq}x(k)}{[']\forall i{\neq}j\ x(i){\neq}x(j) \rightarrow [\forall i\, x(i)' = v(i), v(i)' = -b]\,\forall j{\neq}k\ x(j){\neq}x(k)}$$

The quantified assignment $\forall i\, x(i) := -\frac{b}{2}t^2 + v(i)t + x(i)$ solving the above quantified differential equation system can be obtained easily from the solution $x := -\frac{b}{2}t^2 + vt + x$ of the deparametrized differential equation system $x' = v, v' = -b$, just by adding the parameter i back in and checking whether this gives a solution.

We only present proof rules for first-order definable solutions of quantified differential equations here. We refer to previous work [Pla10a] for induction techniques that handle differential equations without solving them and that work for nondeterministic differential equations with disturbances. We have shown recently that these differential induction techniques extend to quantified differential equations using *quantified differential invariants* [Pla11].

Quantified Assignments. Rules [:=],⟨:=⟩,[:] handle quantified assignments. Rule [:] characterizes the fact that quantified assignments to f have no effect on all other operators $\Upsilon \neq f$ (including other function symbols, $\wedge$, if then else fi), so that Υ will not be affected by the quantified assignment and can be skipped over. The argument $\vec{u}$ may still be affected by the quantified assignment, hence [:] prefixes $\vec{u}$ (component-wise) by $\forall i : C\ f(\vec{s}) := \theta$. Hence, the [:] rule maps a quantified assignment over all arguments homomorphically. For example, if Υ is an operator taking two arguments and is not the function symbol f, then rule [:] derives the proof step

$$[:]\frac{\Upsilon([\forall i : C\ f(\vec{s}) := \theta]u_1, [\forall i : C\ f(\vec{s}) := \theta]u_2)}{[\forall i : C\ f(\vec{s}) := \theta]\,\Upsilon(u_1, u_2)}$$

Rules [:=],⟨:=⟩ characterize how a quantified assignment to f affects the value of a term $f(\vec{u})$ (these rules are equivalent for the injective case, i.e., a match for at most one i). Their effect depends on whether the quantified assignment $\forall i : C\ f(\vec{s}) := \theta$ *matches* $f(\vec{u})$, i.e., there is a choice for i such that $f(\vec{u})$ is affected by the assignment, because $\vec{u}$ is of the form $\vec{s}$ for some i. Whether it matches or not cannot always be decided statically, because it may depend on the particular interpretations. Hence, the premises of rules [:=],⟨:=⟩ make a case distinction on matching by yielding an if-then-else formula. The formula if ϕ then ϕ_1 else ϕ_2 fi is short notation for $(\phi \to \phi_1) \wedge (\neg\phi \to \phi_2)$. If the quantified assignment does not match (else part), the occurrence of f in $\phi(f(\vec{u}))$ will be left unchanged, because f is not changed at position $\vec{u}$. If it matches (then part), the premise uses the term θ assigned to $f(\vec{s})$ instead of $f(\vec{u})$, either for all possible $i : C$ that match $f(\vec{u})$ in case of [:=], or for some of those $i : C$ in case of ⟨:=⟩. The universal and existential quantifiers pick the same unique i, because the quantified assignment needs to be injective for [:=],⟨:=⟩. In all cases, the original quantified assignment $\forall i : C\ f(\vec{s}) := \theta$, which we abbreviate by $\mathcal{A}$, will be applied to $\vec{u}$ in the premise, because the value of argument $\vec{u}$ may also be affected by $\mathcal{A}$, recursively.

A special case of [:=] applies to the schematic case where $\vec{s}$ is of the form i, which matches trivially:

$$[:=]\frac{\forall i : C\ (i = [\forall i : C\ f(i) := \theta]u \to \phi(\theta))}{\phi([\forall i : C\ f(i) := \theta]f(u))}$$

If f does not occur in u, then [:] simplifies this proof step further:

$$[:=],[:]\frac{\forall i : C\ (i = u \to \phi(\theta))}{\phi([\forall i : C\ f(i) := \theta]f(u))}$$

Recall that θ_i^u is the term θ with i replaced by u. Standard first-order reasoning simplifies the above to a derived rule that we again denote by [:=] (where f does not occur in u)

$$[:=]\frac{\phi(\theta_i^u)}{\phi([\forall i : C\ f(i) := \theta]f(u))}$$

Together with [:] to propagate the change to both arguments of $\neq$, this derived rule proves, for example, the following proof step:

$$[:=],[:]\frac{\forall i \neq j\, x(i) \neq x(j) \to \forall j \neq k\, (-\frac{b}{2}s^2 + v(j)s + x(j) \neq -\frac{b}{2}s^2 + v(k)s + x(k))}{\forall i \neq j\, x(i) \neq x(j) \to \forall j \neq k\, [\forall i\, x(i) := -\frac{b}{2}s^2 + v(i)s + x(i)]\, x(j) \neq x(k)}$$

Rules [:=],⟨:=⟩,[:] also apply for assignments without quantifiers, which correspond to vacuous quantification $\forall i : C$ where i does not occur anywhere. The following rule, for

example, is a special case of [:=]

$$[:=]\frac{\mathsf{if}\ s=[f(s):=\theta]k\ \mathsf{then}\ \phi(\theta)\ \mathsf{else}\ \phi(f([f(s):=\theta]k))\ \mathsf{fi}}{\phi([f(s):=\theta]f(k))}$$

If f does not occur in term k, then this special case of [:=] simplifies further to

$$[:=]\frac{\mathsf{if}\ s=k\ \mathsf{then}\ \phi(\theta)\ \mathsf{else}\ \phi(f(k))\ \mathsf{fi}}{\phi([f(s):=\theta]f(k))}$$

Note that the if-then-else case distinction is necessary in general, because the effect of the (vacuously quantified) assignment depends on whether $s=k$ holds, which may depend on what value k has at the moment. Rules [:∗],⟨:∗⟩ reduce nondeterministic assignments to universal or existential quantification. For the handling of other general nondeterministic assignments and nondeterministic differential equations, also see previous work [Pla10a].

It is easy, just notationally cumbersome, to extend rules [:=],⟨:=⟩,[:] to vectorial extensions including systems of quantified assignments to multiple function symbols like $\forall i:C\ (a(i):=a(i)+1,t(i):=0)$ following the ideas of parallel updates [BP06, Rüm06]. With those, it is also easy to extend rules [′],⟨′⟩ to quantified differential equation systems like $\forall i:C\ (x(i)'=v(i),v(i)'=a(i))$ where the solution is a system of quantified assignments.

Object Creation. Given our definition of $\mathsf{new}\,C$ as a QHP from Section 5, object creation can be proven by the other proof rules in Fig. 2. With this definition of $\mathsf{new}\,C$, we obtain, for example, the following derived rule using [;],[:∗],[?]

$$\frac{\forall n:C\ (\mathsf{E}(n)=0\rightarrow[\mathsf{E}(n):=1]\phi)}{[n:=\mathsf{new}\,C]\phi}$$

In addition, axiom E expresses that, for sort $C\neq\mathbb{R}$, there always is a new object n that has not been created yet ($\mathsf{E}(n)=0$), because domains are infinite. This is the only place where we are using the assumption about infinite domains. The primary purpose is to simplify technicalities that would arise if object creation could run out of objects and may thus fail if, e.g., no more cars can be created. If this resource limitation is intended in a particular system, it can be modeled easily using patterns like $n:=\mathsf{new}\,C\cup\mathit{fail}$.

Quantifiers. For quantifiers, we cannot just use standard rules [Fit96], because these are for uninterpreted first-order logic and work by instantiating quantifiers, eagerly as in ground tableaux or lazily by unification as in free variable tableaux [Fit96]. Q$\mathsf{d}\mathcal{L}$ is based on first-order logic interpreted over the reals [Tar51, CH91]. A formula like $\exists a:\mathbb{R}\ \forall x:\mathbb{R}\ (x^2+a>0)$ cannot be proven by the instantiation rules for the quantifiers but it is still valid for reals. Thus, for handling quantifiers over the reals, we would like to use the standard decision procedure for first-order real arithmetic (i.e., real-closed fields) instead, which is quantifier elimination [Tar51, CH91].

Definition 6.1. (Quantifier elimination). A first-order theory admits *quantifier elimination* if, with each formula ϕ, a quantifier-free formula $\mathrm{QE}(\phi)$ can be associated effectively that is equivalent (i.e., $\phi\leftrightarrow\mathrm{QE}(\phi)$ is valid) and has no additional free variables or function symbols. The operation QE is further assumed to evaluate formulas without variables, yielding a decision procedure for closed formulas of this theory (i.e., formulas without free variables).

Unfortunately, we cannot use quantifier elimination of the theory of real-closed fields [Tar51, CH91] either, because it cannot be applied to QdℒC formulas with modalities, since these are quantified reachability statements. Even in discrete dynamic logic, quantifiers plus modalities make validity Π^1_1-complete [HKT00, Theorem 13.1]. QE cannot handle sorts $C \neq \mathbb{R}$.

Instead, our QdℒC proof rules combine quantifier handling of many-sorted logic based on instantiation with theory reasoning by QE for the theory of reals. Figure 2 shows proof rules for quantifiers that combine with decision procedures for real-closed fields. Classical instantiation is sound for sort $\mathbb{R}$, just incomplete. For example, rule $\exists$r can solve the following arithmetic by instantiation:

$$\exists\text{r}\frac{a > 0 \rightarrow (a+1)^2 > a, \exists x\, x^2 > a}{a > 0 \rightarrow \exists x\, x^2 > a}$$

Rules $\exists$r and $\forall$l instantiate x with arbitrary terms θ, including a new free variable X, in which case $\exists$r and $\forall$l become the usual γ-rules of free-variable proof calculi [Fit96, FM99]:

$$(\exists\text{r})\quad \frac{\Gamma \rightarrow \phi(X), \exists x : C\ \phi(x), \Delta}{\Gamma \rightarrow \exists x : C\ \phi(x), \Delta} \qquad (\forall\text{l})\quad \frac{\Gamma, \phi(X), \forall x : C\ \phi(x) \rightarrow \Delta}{\Gamma, \forall x : C\ \phi(x) \rightarrow \Delta}$$

Rules $\forall$r and $\exists$l correspond to the liberalized δ^+-rule [HS94] that is a refinement of the classical δ-rule of free-variable tableaux [Fit96]. As in our previous work [Pla08a], rules i$\forall$ and i$\exists$ reintroduce and eliminate quantifiers over $\mathbb{R}$ once QE is applicable, because the remaining constraints are first-order real arithmetical in the respective variables. In particular, the quantifier rules can be used to postpone quantifier elimination until the remaining constraints are first-order, where the quantifier can be reintroduced by i$\forall$ and i$\exists$ [Pla08a].

Unlike in previous work, however, functions and different argument vectors can occur in QdℒC. If the argument vectors $\vec{s}$ and $\vec{t}$ in i$\forall$ have the same value, the same variable X can be reintroduced for $f(\vec{s})$ and $f(\vec{t})$, otherwise different variables $X \neq Y$ have to be used. Whether $\vec{s}$ and $\vec{t}$ have the same value cannot always be decided statically, so rule i$\forall$ makes a case distinction by an if-then-else. Rule i$\forall$ works accordingly for multiple occurrences of $f(\vec{s}), f(\vec{t}), f(\vec{u})$ and so on in arbitrary positions in the formula, where more variables X, Y, Z are introduced to quantify over. It is easy to turn rule i$\forall$ into a rule that successively substitutes one term $f(\vec{s})$ by a fresh variable X everywhere at a time instead of handling all $f(\vec{s}), f(\vec{t}), f(\vec{u})$ at once.

Rule i$\exists$ can reintroduce an existential quantifier for a free (existential) logical variable X and merges all proof branches containing X, because X has to satisfy all branches simultaneously. It thus has multiple conclusions. Rule i$\exists$ reintroduces an existential quantifier and performs quantifier elimination for a free (existential) logical variable X that has been introduced by $\exists$r,$\forall$l before by choosing a fresh variable X for θ. We use the same rule i$\exists$ as in previous work and refer to that work [Pla08a] for further explanations of merging.

Rules i$\forall$ and i$\exists$ require that quantifier elimination (QE) is applicable to the resulting formula. If the resulting formulas still have occurrences of the quantified variables in the scope of modalities, then QE is not applicable and rules i$\forall$ and i$\exists$ have to be postponed until the modalities have been dealt with by other proof rules from Fig. 2. Even for first-order formulas, we cannot just apply classical quantifier elimination in real-closed fields [Tar51], because the first-order theory of real-closed fields does not include function symbols. For example, $\forall i\,(a(i) \geq 0)$ is a formula of first-order real arithmetic *augmented with function symbols*, hence quantifier elimination in real-closed fields due to Tarski [Tar51] is

not applicable. It cannot even be expressed in quantifier-free form, because its truth-value depends on the value of function a at unboundedly many positions. This makes sense. QE is a decision procedure for first-order real arithmetic. But first-order logic (even without arithmetic) is only semidecidable, so we cannot handle it by QE and need to rely on the instantiation rules ∀r,∀l,∃r,∃l, which are complete for first-order logic. Nevertheless, from previous work [Pla08a], we obtain the following result on how to lift QE to the presence of function symbols:

Lemma 6.2. *(*Quantifier elimination lifting [Pla08a]*). Quantifier elimination can be lifted to instances of formulas of first-order theories that admit quantifier elimination, i.e., to formulas that result from the base theory by substitution.*

For example, $\forall y\,(a(i) < y^2)$ is a formula of first-order real arithmetic augmented with function symbols such that quantifier elimination in real-closed fields due to Tarski [Tar51] is not (directly) applicable. By Lemma 6.2, however, QE can be lifted to this formula, because it is an instance of $\forall y\,(Z < y^2)$, for Z replaced with $a(i)$. Hence,

$$\mathrm{QE}(\forall y\,(a(i) < y^2)) \equiv (\mathrm{QE}(\forall y\,(Z < y^2)))^{a(i)}_Z \equiv (Z < 0)^{a(i)}_Z \equiv a(i) < 0$$

Global Rules. The proof rules in the last block of Fig. 2 depend on the truth of their premises in all states, thus the context Γ, Δ cannot be used in the premise, because it may be specific to the current state. The rules are given in a form that best displays their underlying logical principles. The general pattern for applying these rules to prove that the succedent of their conclusion holds is to prove that both their premise and the antecedent of their conclusion hold. In particular, the antecedent can be thought of as holding in the current state, whereas the premise can be thought of as holding in all states because the context Γ, Δ is gone.

Rules $[]gen$,$\langle\rangle gen$ are Gödel generalization rules and can be used to strengthen postconditions: antecedent $[\alpha]\phi$ is sufficient for proving succedent $[\alpha]\psi$ when postcondition ϕ entails ψ in all states, as shown in the premise of $[]gen$. Clearly, for rule $[]gen$, if all states reachable by α satisfy ϕ (antecedent $[\alpha]\phi$) and ϕ implies ψ in all states (premise $\phi\rightarrow\psi$), then ψ also holds in all states reachable by α (succedent $[\alpha]\psi$). Similarly, for rule $\langle\rangle gen$, if some state reachable by α satisfies ϕ (antecedent $\langle\alpha\rangle\phi$) and ϕ implies ψ in all states (premise $\phi\rightarrow\psi$), then ψ also holds in some state reachable by α (succedent $\langle\alpha\rangle\psi$).

Rule *ind* is an induction schema for loops with *inductive invariant* ϕ [HKT00, Pla08a]. Rule *ind* says that ϕ holds after any number of repetitions of α if it holds initially (antecedent) and, for all states, invariant ϕ remains true after one iteration of α (premise). If ϕ is true after executing α whenever ϕ has been true before (premise), then, if ϕ holds in the beginning, ϕ will continue to hold, no matter how often we repeat α in $[\alpha^*]\phi$.

Similarly, *con* generalizes Harel's convergence rule [HKT00] to the hybrid case with decreasing *variant* φ [Pla08a]. Rule *con* expresses that the variant $\varphi(v)$ holds for some real number $v \leq 0$ after repeating α sufficiently often (succedent) if $\varphi(v)$ holds for some real number at all in the beginning (antecedent) and, by premise, $\varphi(v)$ can decrease after every execution of α by 1 (or another positive real constant). This rule can be used to show positive progress (by 1) with respect to $\varphi(v)$ by executing α.

Example. As a simple example illustrating how the QdL proof calculus works, we consider the QdL derivation in Fig. 4 for a simple QdL formula. The QdL formula that we consider here follows the pattern of the running example formula in (3.1). But we simplify the formula to consider just one case and postpone the discussion of the full system to Section 9. Here we consider the QdL formula:

$$\forall i{\neq}j\, x(i){\neq}x(j) \rightarrow [\forall i\, (x(i)' = v(i), v(i)' = -b)]\, \forall j{\neq}k\, x(j){\neq}x(k) \tag{6.1}$$

The difference of the simpler QdL formula (6.1) compared to the full QdL formula (3.1) is that the simpler formula considers only the case of the QHP dynamics where all cars are braking. Certainly, if the system would not be safe when all cars are braking (which is one possible behavior of *DCCS*), then it would not be safe always. The other difference is that (6.1) has a weaker assumption in the precondition. It only assumes that cars start from different positions ($\forall i{\neq}j\, x(i){\neq}x(j)$), not that they respect the compatibility constraint $\forall i, j : C\ \mathcal{M}(i, j)$. In fact, we are using the derivation in Fig. 4 to find out how we need to choose $\mathcal{M}(i, j)$ to ensure collision freedom, because $\mathcal{M}(i, j)$ needs to imply at least that all cars would remain safe when braking.

The derivation in Fig. 4 can be used to find out under which circumstances the QdL formula (6.1), from which we start the derivation at the bottom of Fig. 4, is true. Formula (6.1) claims that cars would never crash if they start at different positions ($\forall i{\neq}j\, x(i){\neq}x(j)$) and all cars brake by following the dynamics $\forall i\, x(i)'' = -b$. Since braking is the safest operation for cars, we might think that car control would always be safe in this most conservative scenario. But that is not the case. If the cars start with incompatible velocities and distances, then not even braking can prevent a crash. The premise discovered by the QdL derivation in Fig. 4 reveals that collisions will only be prevented by braking if the initial velocities and positions satisfy the monotonicity condition $\mathcal{M}(j, k)$ that we have already shown in (3.2).

$$\dfrac{}{\mathrm{QE}(\forall X, Y, V, W\, (j \neq k \wedge X \neq Y \rightarrow X{\leq}Y \wedge V{\leq}W \vee X{\geq}Y \wedge V{\geq}W))}\ \mathrm{i}\forall$$

$$\dfrac{\forall i{\neq}j\, x(i){\neq}x(j) \rightarrow (j{\neq}k \rightarrow (x(j){\leq}x(k) \wedge v(j){\leq}v(k) \vee x(j){\geq}x(k) \wedge v(j){\geq}v(k)))}{\forall i{\neq}j\, x(i){\neq}x(j) \rightarrow \mathrm{QE}(\forall s{\geq}0\, (j{\neq}k \rightarrow -\tfrac{b}{2}s^2 + v(j)s + x(j) \neq -\tfrac{b}{2}s^2 + v(k)s + x(k)))}\ \mathrm{QE}$$

$$\dfrac{\forall i{\neq}j\, x(i){\neq}x(j), s{\geq}0 \rightarrow (j{\neq}k \rightarrow -\tfrac{b}{2}s^2 + v(j)s + x(j) \neq -\tfrac{b}{2}s^2 + v(k)s + x(k))}{\forall i{\neq}j\, x(i){\neq}x(j), s{\geq}0 \rightarrow \forall j{\neq}k\, (-\tfrac{b}{2}s^2 + v(j)s + x(j) \neq -\tfrac{b}{2}s^2 + v(k)s + x(k))}\ \mathrm{i}\forall$$

$$\dfrac{\forall i{\neq}j\, x(i){\neq}x(j), s{\geq}0 \rightarrow \forall j{\neq}k\, [\forall i\, x(i) := -\tfrac{b}{2}s^2 + v(i)s + x(i)]\, x(j){\neq}x(k)}{\forall i{\neq}j\, x(i){\neq}x(j), s{\geq}0 \rightarrow [\forall i\, x(i) := -\tfrac{b}{2}s^2 + v(i)s + x(i)]\, \forall j{\neq}k\, x(j){\neq}x(k)}\ \forall\mathrm{r}$$

$$\dfrac{\forall i{\neq}j\, x(i){\neq}x(j) \rightarrow (s{\geq}0 \rightarrow [\forall i\, x(i) := -\tfrac{b}{2}s^2 + v(i)s + x(i)]\, \forall j{\neq}k\, x(j){\neq}x(k))}{\forall i{\neq}j\, x(i){\neq}x(j) \rightarrow \forall t{\geq}0\, [\forall i\, x(i) := -\tfrac{b}{2}t^2 + v(i)t + x(i)]\, \forall j{\neq}k\, x(j){\neq}x(k)}\ \rightarrow\mathrm{r}$$

$$\dfrac{\forall i{\neq}j\, x(i){\neq}x(j) \rightarrow \forall t{\geq}0\, [\forall i\, x(i) := -\tfrac{b}{2}t^2 + v(i)t + x(i)]\, \forall j{\neq}k\, x(j){\neq}x(k)}{\forall i{\neq}j\, x(i){\neq}x(j) \rightarrow [\forall i\, (x(i)' = v(i), v(i)' = -b)]\, \forall j{\neq}k\, x(j){\neq}x(k)}\ \forall\mathrm{r}$$

$$\dfrac{\forall i{\neq}j\, x(i){\neq}x(j) \rightarrow [\forall i\, (x(i)' = v(i), v(i)' = -b)]\, \forall j{\neq}k\, x(j){\neq}x(k)}{}\ [']$$

Figure 4: Example of a QdL derivation to prove collision-freedom of simple car control.

The proof in Fig. 4 starts with the conjecture at the bottom (goal). The proof uses rule $[']$ to turn the quantified differential equation system into a quantified assignment with an extra quantifier for the duration t of the evolution. The quantified differential equation system is easy to solve. The quantified assignment $\forall i\, x(i) := -\frac{b}{2}t^2 + v(i)t + x(i)$ solving it can be obtained easily from the solution $x := -\frac{b}{2}t^2 + vt + x$ of the deparametrized

differential equation system $x' = v, v' = -b$, just by adding the parameter i back in and checking that the resulting terms solve the quantified differential equation. Now the top-most logical operator in the succedent is the quantifier $\forall t$. Even though it is a quantifier over a real variable, we cannot use the decision procedure of quantifier elimination for real-closed fields [Tar51] to handle it, because we do not have a formula of first-order real arithmetic, but still a $\mathsf{Qd}\mathcal{L}$ formula with a modality expressing a property of all reachable states. Instead, we use rule $\forall$r to postpone quantifier elimination and turn variable t into a Skolem function s. This Skolem function has no arguments, because no free (existential) logical variables occur in the formula [Pla08a]. After that, we use the standard propositional sequent rule $\rightarrow$r to normalize implications in the succedent into sequent form by moving their left-hand side to the antecedent.

The resulting quantified assignment to $x(i)$ (for all i) takes effect on the postcondition $\forall j{\neq}k\, x(j) \neq x(k)$ by skipping over the quantifier $\forall j{\neq}k$ with rule [:] and then affecting $x(j)$ and $x(k)$ subsequently by rule [:=] (and another application of [:] to skip over $\neq$, which is not shown in Fig. 4).

At this point (the top-most use of rule $\forall$r in Fig. 4), we already have a first-order formula and it may seem as if we could apply i$\forall$ directly instead of $\forall$r. This would not work, however, because quantifier elimination works from inside out and will have to eliminate the inner quantifier $\forall j{\neq}k$ before the outer quantifier $\forall s$. Yet, the resulting formula is not an instance of first-order real arithmetic (not even when using Lemma 6.2), because there are dependencies on the quantified variables j, k in function arguments of the resulting formula:

$$\forall s{\geq}0\, \forall j{\neq}k \left(-\frac{b}{2}s^2 + v(j)s + x(j) \neq -\frac{b}{2}s^2 + v(k)s + x(k)\right)$$

Instead, the proof in Fig. 4 uses rule $\forall$r to turn the quantified variables j, k into Skolem functions, which, for simplicity, we again denote by j and k. Subsequently, we can use rule i$\forall$ to reintroduce a quantifier for the Skolem function s. Rule i$\forall$ does not produce an if-then-else, because s has no arguments. This time, the formula is still not in first-order real arithmetic, because function symbols like $v(j)$ occur. However, it is an instance ($v(j)$ for V and $x(j)$ for X and $v(k)$ for W and $x(k)$ for Y) of the following formula of first-order real arithmetic:

$$\forall s{\geq}0 \left(j{\neq}k \rightarrow -\frac{b}{2}s^2 + Vs + X \neq -\frac{b}{2}s^2 + Ws + Y\right) \tag{6.2}$$

and thus quantifier elimination can be lifted by Lemma 6.2. The result of quantifier elimination is an instance (with the same instantiation as above) of the result of applying QE to (6.2). To improve traceability, we show the application of QE as a separate proof step (indicated by QE).

Finally (the top-most rule), we use rule i$\forall$ to finish the deduction. We still cannot yet use rule i$\forall$ for j, k, but we can use rule i$\forall$ for the (non-Skolem) function symbols x and v. This time, the use of rule i$\forall$ is more involved than before, because the functions x and v have arguments. When using rule i$\forall$ on

$$\forall i{\neq}j\, x(i){\neq}x(j){\rightarrow}(j{\neq}k \rightarrow (x(j){\leq}x(k){\wedge}v(j){\leq}v(k) \vee x(j){\geq}x(k){\wedge}v(j){\geq}v(k)))$$

we formally obtain

$$\mathrm{QE}\big(\forall X,Y,V,W\ \big(\mathsf{if}\ j=k\ \mathsf{then}\ j\neq k\wedge X\neq X\to X{\leq}X\wedge V{\leq}V\vee X{\geq}X\wedge V{\geq}V\ \mathsf{else}\ j\neq k\wedge X\neq Y\to X{\leq}Y\wedge V{\leq}W\vee X{\geq}Y\wedge V{\geq}W\big)\big)$$

Since the condition $\mathsf{if}\ j=k$ contradicts the assumption $j\neq k$, this formula simplifies to:

$$\mathrm{QE}(\forall X,Y,V,W\ (j\neq k\wedge X\neq Y\to X{\leq}Y\wedge V{\leq}W\vee X{\geq}Y\wedge V{\geq}W))$$

Simplifications like those arise often and can be exploited for automated theorem proving. Applying QE in the above formula yields *false*, so the derivation in Fig. 4 does not result in a closed proof. This is good news, however, because the conjecture at the bottom of Fig. 4 is not true under all interpretations. The constraints at the top of Fig. 4 can be used to construct the constraints required for safety, which coincide with $\mathcal{M}(j,k)$ from (3.2).

Derived Rules. Several useful rules can be derived from the QdL rules in Fig. 2 to shortcut common reasoning cases. For instance, the following derived rules characterize the effect of creating objects of type C on actualist quantifiers over type $C!$ (where n is of type C):

$$(\nu\forall)\ \frac{[\mathsf{E}(n):=1]\phi(n)\wedge\forall i:C!\ [\mathsf{E}(n):=1]\phi(i)}{[\mathsf{E}(n):=1]\forall i:C!\ \phi(i)}\qquad(\nu\exists)\ \frac{[\mathsf{E}(n):=1]\phi(n)\vee\exists i:C!\ [\mathsf{E}(n):=1]\phi(i)}{[\mathsf{E}(n):=1]\exists i:C!\ \phi(i)}$$

They commute the effect $[\mathsf{E}(n):=1]$ of object creation with quantification, retaining the effect on the new object explicitly. Rule $\nu\forall$ states that the new object denoted by n—which may not have been created before—needs to satisfy $\phi(n)$ too in order for $\forall i:C!\ \phi(i)$ to hold after $\mathsf{E}(n):=1$ ensures that n is created. Dually, rule $\nu\exists$ states that created object n is an alternative choice for i, in addition to the previous domain of $C!$.

A similar derived rule νA states that, after creating an object of type C, this created object will be affected by actualist quantified assignments ranging over $C!$, so that commuting has to take care of the effect on the new object explicitly.

$$(\nu A)\ \frac{[\forall i:C!\cup\{n\}\ f(\vec{s}):=\theta][\mathsf{E}(n):=1]\phi}{[\mathsf{E}(n):=1][\forall i:C!\ f(\vec{s}):=\theta]\phi}$$

For this situation where n is adjoined to the range of quantification (n might even have been in the range before, so the union is not necessarily disjoint), we use the following mnemonic abbreviation in the premise of νA:

$$\forall i:C!\cup\{n\}\ f(\vec{s}):=\theta\ \equiv\ \forall i:C\ (f(\vec{s}):=\mathsf{if}\ i=n\vee\mathsf{E}(i)\ \mathsf{then}\ \theta\ \mathsf{else}\ f(\vec{s})\ \mathsf{fi})$$

Note that we cannot simply apply the assignment to n separately before $\forall i:C!\ f(\vec{s}):=\theta$ as in $i:=n;\ f(\vec{s}):=\theta;\ \forall i:C!\ f(\vec{s}):=\theta$, because that would change f twice if n already existed initially.

7. Soundness

We have presented a proof calculus for QdL in Section 6. One of the most important questions about it is whether we can rely on the proofs and know that every QdL formula proven in the QdL calculus is really a valid formula. That is, the question is whether the QdL calculus is sound. An unsound calculus would be disastrous, because we could use it

to "prove" counterfactual properties. We need to make sure that the proof calculus fits to the semantics of QdL. Indeed it does.

Theorem 7.1. *(*Soundness*). The* QdL *calculus is* sound*: every* QdL *formula that can be proven in the* QdL *calculus is valid, i.e., true in all states.*

Proof. The calculus is sound if each rule instance is sound. Some of the rules of the QdL calculus are even *locally sound*, i.e., their conclusion is true at state σ if all its premises are true at σ, which implies soundness. The proofs for the propositional rules, and regular rules [;],⟨;⟩,[∪],⟨∪⟩,[?],⟨?⟩ are as usual [Pla10b]. We refer to previous work [Pla08a, Pla10b] for the soundness proofs for ∃r,∀l,∀r,∃l,i∃, which are more involved.

i∀ Rule i∀ is locally sound. For this, we assume that the premise holds, i.e., we assume $\sigma \models \mathrm{QE}(\forall X, Y\ (\mathsf{if}\ \vec{s}=\vec{t}\ \mathsf{then}\ \Phi(X)\to\Psi(X)\ \mathsf{else}\ \Phi(X)\to\Psi(Y)\ \mathsf{fi}))$. Since QE yields an equivalence, we conclude $\sigma \models \forall X, Y\ (\mathsf{if}\ \vec{s}=\vec{t}\ \mathsf{then}\ \Phi(X)\to\Psi(X)\ \mathsf{else}\ \Phi(X)\to\Psi(Y)\ \mathsf{fi})$. This is equivalent to $\sigma \models \mathsf{if}\ \vec{s}=\vec{t}\ \mathsf{then}\ \forall X\ (\Phi(X)\to\Psi(X))\ \mathsf{else}\ \forall X, Y\ (\Phi(X)\to\Psi(Y))\ \mathsf{fi}$, because the fresh variables X, Y do not occur in $\vec{s}$ or $\vec{t}$. Then we assume the antecedent of the conclusion is true, i.e., $\sigma \models \Phi(f(\vec{s}))$. We conclude that the succedent of the conclusion is true, $\sigma \models \Psi(f(\vec{t}))$, by choosing $\sigma[\![f(\vec{s})]\!]$ for X and $\sigma[\![f(\vec{t})]\!]$ for Y in the premise. If $\sigma \models \neg(\vec{s}=\vec{t})$ then $\sigma \models \Psi(f(\vec{t}))$ follows directly from the premise. If, otherwise, $\sigma \models \vec{s}=\vec{t}$, then $\sigma \models \Psi(f(\vec{t}))$ also follows, because the choice $\sigma[\![f(\vec{s})]\!]$ for X is identical to the choice $\sigma[\![f(\vec{t})]\!]$ for Y in the premise. By admissibility of substitutions, any variables occurring in terms $\vec{s}$ and $\vec{t}$ are free at all occurrences of $f(\vec{s})$ and $f(\vec{t})$, hence their value is the same in all occurrences.

⟨:=⟩ Rule ⟨:=⟩ is locally sound for injective $\forall i : C\ f(\vec{s}) := \theta$, which we abbreviate as $\mathcal{A}$. Injective $\mathcal{A}$ give a deterministic transition. We assume that the premise holds $\sigma \models \mathsf{if}\ \exists i : C\ \vec{s} = \langle\mathcal{A}\rangle\vec{u}\ \mathsf{then}\ \exists i : C\ (\vec{s} = \langle\mathcal{A}\rangle\vec{u} \wedge \phi(\theta))\ \mathsf{else}\ \phi(f(\langle\mathcal{A}\rangle\vec{u}))\ \mathsf{fi}$. We show that $\sigma \models \phi(\langle\forall i : C\ f(\vec{s}) := \theta\rangle f(\vec{u}))$. First assume that, with a fresh variable z, $\phi(z)$ is a first-order formula without modalities or quantifiers. Let τ be the (unique) state with $(\sigma, \tau) \in \rho(\forall i : C\ f(\vec{s}) := \theta) = \rho(\mathcal{A})$. By renaming, we can assume the quantified variable i not to occur anywhere else than in $\mathcal{A}$. We write this occurrence constraint as $i \notin \vec{u}$ and $i \notin \phi(z)$.

– Suppose $\sigma \models \exists i : C\ \vec{s} = \langle\mathcal{A}\rangle\vec{u}$, then $\sigma \models \exists i : C\ (\vec{s} = \langle\mathcal{A}\rangle\vec{u} \wedge \phi(\theta))$ by premise. That is equivalent to: there is an $e \in \sigma(C)$ with $\sigma_i^e \models \vec{s} = \langle\mathcal{A}\rangle\vec{u} \wedge \phi(\theta)$. That means $\sigma_{i\,z}^{e\,d} \models \phi(z)$ for $d := \sigma_i^e[\![\theta]\!]$ by the substitution lemma. This is equivalent to $\sigma_z^d \models \phi(z)$, because $i \notin \phi(z)$, i.e., i does not occur in $\phi(z)$, so that its value is irrelevant. We want to show that $\sigma_z^d \models \phi(z)$ also holds for $d = \sigma[\![\langle\mathcal{A}\rangle f(\vec{u})]\!]$, because this implies $\sigma \models \phi(\langle\mathcal{A}\rangle f(\vec{u}))$ by the substitution lemma. Now

$$\sigma[\![\langle\mathcal{A}\rangle f(\vec{u})]\!] = \tau[\![f(\vec{u})]\!] = \tau(f)\big(\tau[\![\vec{u}]\!]\big) = \tau(f)\big(\sigma[\![\langle\mathcal{A}\rangle\vec{u}]\!]\big) \overset{*}{=} \tau(f)\big(\sigma_i^e[\![\vec{s}]\!]\big) \overset{\rho(\mathcal{A})}{=} \sigma_i^e[\![\theta]\!] = d$$

Thus $\sigma \models \phi(\langle\mathcal{A}\rangle f(\vec{u}))$. The equality marked $*$ holds, because the premise implies $\sigma_i^e \models \vec{s} = \langle\mathcal{A}\rangle\vec{u}$, which yields

$$\sigma_i^e[\![\vec{s}]\!] = \sigma_i^e[\![\langle\mathcal{A}\rangle\vec{u}]\!] \overset{i\notin\vec{u}}{=} \sigma[\![\langle\mathcal{A}\rangle\vec{u}]\!]$$

– Suppose $\sigma \models \neg\exists i : C\ \vec{s} = \langle\mathcal{A}\rangle\vec{u}$, then $\sigma \models \phi(f(\langle\mathcal{A}\rangle\vec{u}))$ by premise. Consequently $\sigma_z^d \models \phi(z)$ for $d := \sigma[\![f(\langle\mathcal{A}\rangle\vec{u})]\!]$ by the substitution lemma. We show that $\sigma_z^d \models \phi(z)$ also holds for $d = \sigma[\![\langle\mathcal{A}\rangle f(\vec{u})]\!]$, because this implies $\sigma \models \phi(\langle\mathcal{A}\rangle f(\vec{u}))$

by the substitution lemma. This time we have

$$\sigma[\![\langle\mathcal{A}\rangle f(\vec{u})]\!] = \tau[\![f(\vec{u})]\!] = \tau(f)\big(\tau[\![\vec{u}]\!]\big) \overset{*}{=} \sigma(f)\big(\tau[\![\vec{u}]\!]\big) = \sigma(f)\big(\sigma[\![\langle\mathcal{A}\rangle\vec{u}]\!]\big) = \sigma[\![f(\langle\mathcal{A}\rangle\vec{u})]\!] = d$$

The equality marked $*$ holds, because—by assumption $\sigma \models \neg\exists i{:}C\ \vec{s} = \langle\mathcal{A}\rangle\vec{u}$—we know that for position $\tau[\![\vec{u}]\!] = \sigma[\![\langle\mathcal{A}\rangle\vec{u}]\!]$ there is no $e \in \sigma(C)$ such that

$$\sigma_i^e[\![\vec{s}]\!] = \tau[\![\vec{u}]\!] = \sigma[\![\langle\mathcal{A}\rangle\vec{u}]\!] \overset{i\notin\vec{u}}{=} \sigma_i^e[\![\langle\mathcal{A}\rangle\vec{u}]\!]$$

Thus $\mathcal{A}$ has no effect on the interpretation of f at position $\tau[\![\vec{u}]\!]$ and σ and τ agree at that position.

In both cases, equivalence of premise and conclusion can be established by following the equations and equivalences backwards, which also gives a proof for the dual rule [:=]. For the case where $\phi(z)$ contains modalities or quantifiers, the proof is accordingly using the substitution lemma and the fact that the interpretation of the symbols occurring in $\langle\mathcal{A}\rangle f(\vec{u})$ is not affected by the modalities and quantifiers in $\phi(z)$ (since all substitutions need to be admissible for QdL rules to be applicable).

[:] Local soundness of rule [:] for injective quantified assignments $\forall i{:}C\ f(\vec{s}) := \theta$ is a simple consequence of the fact that a quantified assignment to f cannot affect the evaluation of another operator $\Upsilon \neq f$, but only its arguments (assuming admissible substitutions).

$\exists$ The soundness of axiom $\exists$ (i.e., validity of the conclusion) is a simple consequence of the fact that we have assumed finite support for the createdness flag $\exists(\cdot)$ and that domains are infinite. That is, there are only finitely many $e \in \sigma(C)$ with $\sigma_i^e \models \exists(i) = 1$, yet domain $\sigma(C)$ is infinite. Consequently, in every state σ, there always is a choice e for i that has not been created yet ($\sigma_i^e \models \exists(i) \neq 1$).

$\langle'\rangle$ Rule $\langle'\rangle$ is locally sound. Let $y_{\vec{s}}(t)$ be simultaneous solutions for the respective differential equations with symbolic initial values $f(\vec{s})$. Let $\langle\forall i{:}C\ f(\vec{s}) := y_{\vec{s}}(t)\rangle$ denote the quantified assignment $\langle\forall i{:}C\ f(\vec{s}) := y_{\vec{s}}(t)\rangle$. Assume σ satisfies the premise: $\sigma \models \exists t{\geq}0\,(\bar{\chi} \wedge \langle\forall i{:}C\ f(\vec{s}) := y_{\vec{s}}(t)\rangle\phi\big)$, with $\forall 0{\leq}\tilde{t}{\leq}t\,\langle\forall i{:}C\ f(\vec{s}) := y_{\vec{s}}(\tilde{t})\rangle\chi$ abbreviated as $\bar{\chi}$. By premise, there is a real $r \geq 0$ such that $\sigma_t^r \models \bar{\chi} \wedge \langle\forall i{:}C\ f(\vec{s}) := y_{\vec{s}}(t)\rangle\phi$. Abbreviate $\forall i{:}C\ f(\vec{s})' = \theta\,\&\,\chi$ by $\mathcal{D}$. We have to show that $\sigma \models \langle\mathcal{D}\rangle\phi$. Equivalently, we show $\sigma_t^r \models \langle\mathcal{D}\rangle\phi$, because t is a fresh variable that does not occur in $\mathcal{D}$ or ϕ. Let function $\varphi{:}[0,r] \to \mathcal{S}$ be defined such that $(\sigma, \varphi(\zeta)) \in \rho(f(\vec{s}) := y_{\vec{s}}(t))$ for all $\zeta \in [0,r]$. By premise, $\varphi(0)$ is identical to σ and ϕ holds at $\varphi(r)$. Thus it only remains to be shown that φ respects the constraints for the flow function φ in the definition of the semantics of $\rho(\mathcal{D})$ in Section 4. In fact, φ obeys the continuity and differentiability properties required for well-definedness of time-derivatives by the corresponding properties of the solution $y_{\vec{s}}(t)$. Moreover, for any $e \in \sigma(C)$, $\varphi(\zeta)_i^e[\![f(\vec{s})]\!] = \sigma_{t\,i}^{\zeta\,e}[\![y_{\vec{s}}(t)]\!]$ has a derivative of value $\varphi(\zeta)_i^e[\![\theta]\!]$, because $y_{\vec{s}}$ is a solution of the quantified differential equation $\forall i{:}C\ f(\vec{s})' = \theta$ with corresponding initial values $\sigma(f(\vec{s}))$. Further, it can be shown that the evolution invariant region χ is respected along φ as follows: By premise, $\sigma_t^r \models \bar{\chi}$ holds for the initial state σ_t^r, thus $\varphi(\zeta) \models \chi$ for all $\zeta \in [0,r]$. Combining these results, we can conclude that φ is a witness for $\sigma \models \langle\mathcal{D}\rangle\phi$.

The converse direction can be shown accordingly to prove equivalence and the dual rule [′] for quantified differential equations with unique solutions (see end of Section 4). Without unique solutions, the rule is more complicated, but still works: all

parameters of all parametric solutions will need to be quantified over in addition to time $t\geq 0$.

[:*] Rules [:*],⟨:*⟩ are locally sound by a simple consequence of the fact that arbitrary nondeterministic assignment of θ for any j of type C to n is the same as corresponding quantification over C. The semantics of $[\forall j:C\ n:=\theta]$ then is equivalent to universal quantification, that of $\langle\forall j:C\ n:=\theta\rangle$ is equivalent to existential quantification.

⟨⟩*gen* Rules []*gen*,⟨⟩*gen*,*ind*,*con* are sound (but not locally sound) by a variation of the usual proofs [HKT00, Pla10b]. For ⟨⟩*gen*, let premise $\phi\rightarrow\psi$ be valid. Let the antecedent be true in a state: $\sigma\models\langle\alpha\rangle\phi$, i.e., let $(\sigma,\tau)\in\rho(\alpha)$ with $\tau\models\phi$. Hence, the premise implies $\tau\models\phi\rightarrow\psi$, thus $\tau\models\psi$, which implies $\sigma\models\langle\alpha\rangle\psi$. The proof for []*gen* is similar.

ind Let premise $\phi\rightarrow[\alpha]\phi$ be valid and let the antecedent of the conclusion be true in σ, that is $\sigma\models\phi$. By premise, $\tau\models\phi$ for all states τ with $(\sigma,\tau)\in\rho(\alpha)$. We thus conclude $\sigma\models\phi\rightarrow[\alpha^*]\phi$ by induction along the series of states reached from σ by repeating α.

con Assume that the antecedent is valid and that the premise holds in σ. By premise, we have that $\tau\models v>0\land\varphi(v)\rightarrow\langle\alpha\rangle\varphi(v-1)$ for all states τ. By antecedent, there is a $d\in\mathbb{R}$ such that $\sigma_v^d\models\varphi(v)$. Now, the proof is a well-founded induction on d. If $d\leq 0$, we have $\sigma\models\langle\alpha^*\rangle\exists v{\leq}0\,\varphi(v)$ directly for zero repetitions. Otherwise, if $d>0$, we have, by premise, that

$$\sigma_v^d\models v>0\land\varphi(v)\rightarrow\langle\alpha\rangle\varphi(v-1)$$

As $v>0\land\varphi(v)$ holds true at σ_v^d, we have for some τ with $(\sigma_v^d,\tau)\in\rho(\alpha)$ that $\tau\models\varphi(v-1)$. Thus, $\tau_v^{d-1}\models\varphi(v)$ satisfies the induction hypothesis for a smaller d and a reachable τ, because $(\sigma,\tau)\in\rho(\alpha)$ as v does not occur in α. The induction is well-founded, because d decreases by 1 up to the base case $d\leq 0$.

□

8. Completeness

The verification problem for distributed hybrid systems is extremely challenging. It has *three independent sources* of undecidability. Thus, no verification technique can be effective. Hence, Q𝖽ℒ cannot be effectively axiomatizable. The discrete fragment of Q𝖽ℒ is not effectively axiomatizable and the discrete fragment of QHPs is a computationally complete sublanguage. The continuous fragment of Q𝖽ℒ is also not effectively axiomatizable. The fragment with only structural and dimension-changing dynamics is not effective either, because it can encode two-counter machines in link data structures. As a stronger result, we give a simple proof showing that each of those fragments of Q𝖽ℒ can define first-order integer arithmetic and are, thus, affected by Gödel's incompleteness theorem [Göd31].

Theorem 8.1. *(*Incompleteness of Q𝖽ℒ*). The discrete fragment of* Q𝖽ℒ*, the continuous fragment of* Q𝖽ℒ*, and the fragment of* Q𝖽ℒ *with structural and dimension-changing dynamics are* not effectively axiomatizable*, i.e., they have no sound and complete effective calculus, because natural numbers are definable in each of those fragments.*

Proof. We prove that natural numbers are definable among the real numbers of QdℒFinterpretations in all three fragments. Then these fragments extend first-order *integer* arithmetic such that the incompleteness theorem of Gödel [Göd31] applies. Gödel's incompleteness theorem shows that no logic extending first-order integer arithmetic can have a sound and complete effective calculus. Natural numbers are definable in the discrete fragment using repetitive additions without continuous evolutions, quantified state change, or first-order function symbols:

$$nat(n) \;\leftrightarrow\; \langle x := 0; (x := x + 1)^* \rangle\, x = n.$$

In the continuous fragment, an isomorphic copy of the natural numbers is definable using linear ordinary (non-quantified) differential equations without first-order function symbols:

$$nat(n) \;\leftrightarrow\; \exists s\, \exists c\, \exists \tau\, (s = 0 \wedge c = 1 \wedge \tau = 0 \wedge \langle s' = c, c' = -s, \tau' = 1 \rangle (s = 0 \wedge \tau = n)).$$

These differential equations characterise sin and cos as unique solutions for s and c, respectively. Their zeros, as detected by τ, correspond to an isomorphic copy of natural numbers, scaled by π, i.e., $nat(n)$ holds iff n is of the form $k\pi$ for a $k \in \mathbb{N}$; see Fig. 5. The initial values for s and c prevent the trivial solution identical to 0.

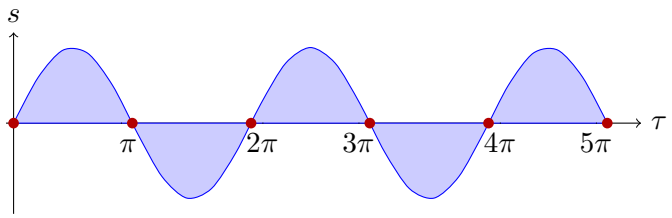


Figure 5: Characterisation of $\mathbb{N}$ as zeros of solutions of differential equations.

Integer arithmetic for natural numbers is also definable in the fragment with only structural and dimensional dynamics. The proof is somewhat more involved, because we do not consider data arithmetic to be part of that fragment. Instead, we characterize natural numbers by chains of links along the values of a function p, where we encode zero by a constant symbol z:

$$nat(n) \;\leftrightarrow\; \langle (?n \neq z;\; n := p(n))^* \rangle\, n = z.$$

We characterize addition by a QHP $plus(s, n, m)$ to express that the result of adding the natural numbers represented by n and m yields the number represented by s:

$$plus(s,n,m) \equiv s := z;\; \big(?n \neq z;\; n := p(n);\; \nu := \mathsf{new};\; p(\nu) := s;\; s := \nu\big)^*;$$
$$\big(?m \neq z;\; m := p(m);\; \nu := \mathsf{new};\; p(\nu) := s;\; s := \nu\big)^*;\; ?(n = z \wedge m = z)$$

The idea behind this characterization is to create a new chain of links along the values of p by first creating exactly as many links as we can follow along p when starting from n, and then continue creating exactly as many links as we can follow along p when starting from m, instead; see Fig. 6. The number of links of the result s then is the sum of the respective numbers of links of n and m.

We characterize multiplication by a QHP $times(s, n, m)$ to express that the result of multiplying the natural numbers represented by n and m yields the number represented by s:

$$times(s,n,m) \equiv s := z;\; \big(?n \neq z;\; n := p(n);\; plus(t,m,s);\; s := t\big)^*;\; ?n = z$$

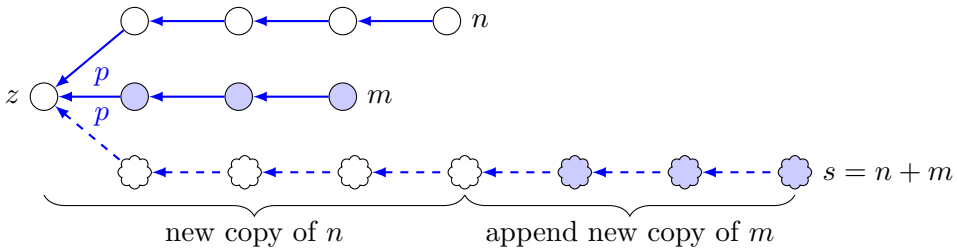


Figure 6: Characterization of $\mathbb{N}$ addition with p links in dimensional dynamics.

The idea behind this characterization is to compute multiplication by a corresponding number of additions characterized by $plus(t, m, s)$. That is, the product of n and m can be computed by adding m to an accumulator s, n times. □

The standard way to show adequacy of proof calculi for problems that are not effective is to prove completeness relative to an oracle for handling a fragment of the logic. Unlike in Cook/Harel relative completeness for discrete programs [Coo78, HKT00], however, Q$\mathsf{d}\mathcal{L}$ cannot be complete relative to the fragment of the data logic (many-sorted first-order logic with reals), because first-order real arithmetic is decidable and many-sorted first-order logic is semidecidable. If the Q$\mathsf{d}\mathcal{L}$ calculus would be complete relative to its data of many-sorted first-order logic with real arithmetic, then, since this is a semidecidable logic, the Q$\mathsf{d}\mathcal{L}$ calculus would be complete altogether, which would contradict Theorem 8.1. Thus, we need a different basis for a relative completeness argument. Unlike in conventional discrete programs, the complexity of distributed hybrid systems truly originates from the actual dynamics, not the data.

Theorem 8.1 shows that the discrete fragment, the continuous fragment, and also the structural/dimensional fragment of Q$\mathsf{d}\mathcal{L}$ each cause non-axiomatizability of Q$\mathsf{d}\mathcal{L}$. The combination of these fragments and their repeated interaction in the QHP dynamics of Q$\mathsf{d}\mathcal{L}$ cannot be any easier. We prove that, nevertheless, our Q$\mathsf{d}\mathcal{L}$ calculus is a complete axiomatization relative to the fragment of Q$\mathsf{d}\mathcal{L}$ that has only quantified differential equations in modalities. We call this sublogic FOQD, the *first-order logic of quantified differential equations*, i.e., (many-sorted) first-order logic with real arithmetic augmented with formulas expressing properties of quantified differential equations, that is, Q$\mathsf{d}\mathcal{L}$ formulas of the form $[\forall i : C\ f(\vec{s})' = \theta \,\&\, \chi]F$. The dual formula $\langle \forall i : C\ f(\vec{s})' = \theta \,\&\, \chi\rangle F$ is expressible as $\neg[\forall i : C\ f(\vec{s})' = \theta \,\&\, \chi]\neg F$. Note that the inclusion of χ in FOQD is not essential [Pla12].

Theorem 8.2. *(*Axiomatization*). The calculus in Fig. 2 is a sound and complete axiomatization of* Q$\mathsf{d}\mathcal{L}$ *relative to quantified differential equations, i.e., every valid* Q$\mathsf{d}\mathcal{L}$ *formula can be derived from valid FOQD tautologies.*

Proof Outline. The (constructive) proof, which, in full, is contained in the remainder of this section, generalizes our earlier proof for static, unquantified hybrid systems [Pla08a] to Q$\mathsf{d}\mathcal{L}$ and distributed hybrid systems. We prove that every valid Q$\mathsf{d}\mathcal{L}$ formula can be proven in the Q$\mathsf{d}\mathcal{L}$ calculus from elementary properties of quantified differential equations (valid oracle instances). The crucial step is to show that every valid property of a repetition α^* of a QHP α for a distributed hybrid system can be proven by *ind* or *con* with a sufficiently strong invariant or variant that is expressible in Q$\mathsf{d}\mathcal{L}$. For this, we show that QHP transitions can be characterized in Q$\mathsf{d}\mathcal{L}$. One decisive difference to our previous proof [Pla08a] is the need

to show that states can be characterized by a fixed-size vector of real numbers, and can thus be quantified over. This is easy in static finite-dimensional systems, but a fairly tricky challenge in unbounded varying-dimensional systems with first-order functions. □

This central result shows that properties of distributed hybrid systems can be proven to exactly the same extent to which properties of quantified differential equations can be proven. Proof-theoretically, the QdL calculus completely lifts verification techniques for quantified continuous dynamics to distributed hybrid dynamics. Even though distributed hybrid systems have numerous independent sources of undecidability, we have shown that all true QdL formulas can be proven in our QdL calculus, if only we manage to tame the complexity of the continuous dynamics. Despite these new independent sources of undecidability, we have shown that QdL can still be axiomatized completely relative to differential equations, only now they are quantified differential equations.

Another important consequence of this result is that decomposition is successful in taming the complexity of distributed hybrid systems. The QdL proof calculus is strictly compositional. All proof rules prove logical formulas or properties of QHPs by reducing them to structurally simpler QdL formulas. As soon as we understand that the distributed hybrid systems complexity comes from a combination of several simpler aspects, we can, hence, tame the system complexity by reducing it to analyzing the dynamical effects of simpler parts. This decomposition principle is exactly how QdL proofs can scale to interesting systems in practice. The relative completeness theorem 8.2 gives the theoretical evidence why this principle works in general.

In the remainder of this section, we present a fully constructive proof of Theorem 8.2. We have already shown that the QdL calculus is a sound axiomatization of QdL in Theorem 7.1. We need to prove that the QdL calculus is a complete axiomatization relative to quantified differential equations: every valid QdL formula can be derived in the QdL calculus from elementary properties of quantified differential equations. We need to prove that every valid QdL formula can be derived in the QdL calculus from a finite set of valid FOQD tautologies. A road map of the proof of Theorem 8.2 that we present here is above.

The basic structure follows that of our relative completeness proof for unquantified differential dynamic logic for fixed-dimensional static hybrid systems in previous work [Pla08a]. Here we generalize the proof to QdL. A fundamental difference to previous work is that states can be characterized trivially in fixed-dimensional static hybrid systems, but it is not obvious why a finite formula would be sufficient in varying dimensions. In (dynamic) distributed hybrid systems, we have to prove that there is a finite formula that can characterize and identify all states (see Section 8.2). In fixed-dimensional static hybrid systems, states can be characterized and identified trivially by a fixed vector of real numbers for each system variable. In QdL, instead, states are full first-order structures with interpretations of functions for all function symbols and the ability to characterize semantic states in logic is no longer obvious. States are no longer assignments of real numbers to a finite number of variables. In QdL, states are full first-order interpretations of function symbols.

Natural numbers are definable in FOQD by Theorem 8.1. Thus, we allow quantifiers over natural numbers like $\forall x:\mathbb{N}\ \phi$ and $\exists x:\mathbb{N}\ \phi$ and over integers $\forall x:\mathbb{Z}\ \phi$ as abbreviations.

8.1. **Characterizing Real Gödel Encodings.** As the central device for constructing a FOQD formula that captures the effect of unboundedly many repetitive hybrid transitions and just uses finitely many real variables, we show that a real version of Gödel encoding is

definable in FOQD. That is, we show that there is a FOQD formula that reversibly packs finite sequences of real values into a single real number.

Observe that a single differential equation system is *not* sufficient for defining these pairing functions as their solutions are differentiable, yet, as a consequence of Morayne's theorem [Mor87], there is no differentiable surjection $\mathbb{R} \to \mathbb{R}^2$, nor to any part of $\mathbb{R}^2$ of positive measure. We show that real sequences can be encoded nevertheless by chaining the effects of solutions of multiple differential equations and quantifiers.

Lemma 8.3. *(*$\mathbb{R}$-Gödel encoding*). The formula $at(Z, n, j, z)$, which holds iff Z is a real number that represents a Gödel encoding of a sequence of n real numbers with real value z at position j (for a position j with $1 \leq j \leq n$), is definable in FOQD. For a formula $\phi(z)$ we abbreviate $\exists z\,(\mathrm{at}(Z, n, j, z) \wedge \phi(z))$ by $\phi(Z_j^{(n)})$.*

Proof. The proof is an immediate corollary to a result from previous work [Pla08a, Lemma 4]. □

8.2. **First-order State Identification.** The crucial step in the proof of Theorem 8.2 is the construction of QdL (in)variants that are strong enough to characterize properties of repetition. In order to be able to characterize QHP state transitions in QdL (in)variants for the completeness proof, we first need to find formulas that characterize/identify states. For finite-dimensional systems of a fixed dimension n, states can simply be characterized completely by the values of all n real state variables. A particular state could be characterized uniquely by the formula $x = 2 \wedge y = 0.5 \wedge z = -0.382$, for example. As a trivial corollary to Lemma 8.3, states can then even be characterized uniquely by one real number when using the $\mathbb{R}$-Gödel encoding. For infinite-dimensional systems, systems with changing dimension, or systems with a dynamics that depends on evolving interpretations of function symbols $f(\vec{s})$, the situation is more difficult. After all, a state of QdL is a full first-order structure with functions as interpretations of function symbols, and these interpretations can change from state to state. Furthermore, in order to navigate among states during the completeness proof, we need to be able to characterize the current first-order state, but also to recall a previously identified first-order state and express what holds true at this state.

We show that the first-order states reachable with QHP α from an initial state can, nevertheless, be characterized uniquely by real numbers, which can thus be quantified over. Furthermore, we show that this correspondence can be axiomatized in FOQD. One key observation is that the first-order interpretations can change from state to state, but only according to the dynamics of the QHP. Intuitively, the difference of any reachable first-order state to the initial state can be characterized by a finite list of differences to the initial state. Clearly this difference concerns only finitely many symbols occurring in α. It also concerns only finitely many positions of their interpreted functions, because actualist quantified assignments and actualist quantified differential equations only change the interpretation of finitely many function symbols at finitely many positions (actual quantified domains $C!$ occurring in actualist quantifiers of QHPs are finite). Note that it is crucial for this argument that we have assumed the actual existence predicate $\mathsf{E}(i)$ to have finite support.

Lemma 8.4. *(*State identification*). Let Σ_b be a finite set of function symbols containing $\mathsf{E}(\cdot)$. The operators $\downarrow$ and @, which identify and recall states reachable by QHPs, are definable in FOQD such that:*

(1) *For every QHP α with $\mathrm{BV}(\alpha) \subseteq \Sigma_b$, every variable $\mathfrak{I} \notin \Sigma_b$ of sort $\mathbb{R}$, and every state σ, the formula $\downarrow\mathfrak{I}$ is true in at most one of the states reachable by α from σ. That is, there is at most one state ι such that $(\sigma,\iota) \in \rho(\alpha)$ and $\iota \models \downarrow\mathfrak{I}$.*

(2) *For every QHP α with $\mathrm{BV}(\alpha) \subseteq \Sigma_b$, every variable $\mathfrak{I} \notin \Sigma_b$ of sort $\mathbb{R}$, every formula ϕ, and every state σ, the formula $@\,\mathfrak{I}\,\phi$ is true in any state reachable by α from σ if and only if ϕ is true in the (unique) state that is reachable by α from σ in which $\downarrow\mathfrak{I}$ holds (provided such a state is reachable at all, otherwise the truth-value of $@\,\mathfrak{I}\,\phi$ is arbitrary). That is, suppose there is a state ι such that $(\sigma,\iota) \in \rho(\alpha)$ and $\iota \models \downarrow\mathfrak{I}$ (thus, by case 1, ι is unique with that property). Then for any state τ with $(\sigma,\tau) \in \rho(\alpha)$, it is the case that $\tau \models @\,\mathfrak{I}\,\phi$ if and only if $\iota \models \phi$. If, on the contrary, there is no state ι with $(\sigma,\iota) \in \rho(\alpha)$ and $\iota \models \downarrow\mathfrak{I}$, then this lemma makes no statement concerning the truth of formula $@\,\mathfrak{I}\,\phi$ at any state τ.*

Proof. The formulas $\downarrow\mathfrak{I}$ and $@\,\mathfrak{I}\,\phi$ are like the *here* and *at* operators of hybrid-nominal logic. We show that they can be characterized by FOQD formulas. For defining $\downarrow\mathfrak{I}$ and $@\,\mathfrak{I}\,\phi$, we use an auxiliary function $is_f(\mathfrak{I},\vec{o})$ to improve readability. The formula $\theta = is_f(\mathfrak{I},\vec{o})$ is true if the value of θ coincides with the value of f at position $\vec{o}$ according to the state characterized by $\mathfrak{I}$ (i.e., where $\downarrow\mathfrak{I}$ is true). We characterize $\theta = is_f(\mathfrak{I},\vec{o})$ by the following FOQD formula:

$$\mathsf{if}\ \exists s:\mathbb{N}\ (s < m \wedge X_s^{(m)} = \vec{o})\ \mathsf{then}\ \exists s:\mathbb{N}\ (s < m \wedge X_s^{(m)} = \vec{o} \wedge \theta = Y_s^{(m)})\ \mathsf{else}\ \theta = f(\vec{o})\ \mathsf{fi}$$

where $\mathfrak{I}$ is split into the following abbreviations $m := \mathfrak{I}_i^{(d)\,(3)}_{\ 1}, X := \mathfrak{I}_i^{(d)\,(3)}_{\ 2}, Y := \mathfrak{I}_i^{(d)\,(3)}_{\ 3}$

further d is the number of symbols in Σ_b and i is the index of f in Σ_b

The function symbol $is_f(\mathfrak{I},\vec{o})$ gives the value (θ) of function f at position $\vec{o}$ at the state characterized by the real number denoted by $\mathfrak{I}$. It can be defined easily using the real pairing function from Lemma 8.3. The basic idea is to understand $\mathfrak{I}$ via the real pairing function as a list of length m of position/value pairs $(X_s^{(m)}/Y_s^{(m)})$, which characterize changes to the value $f(\vec{o})$ for each of the finitely many function symbols $f \in \Sigma_b$. Using an arbitrary but fixed ordering, these function symbols f are identified with their index d in Σ_b. The most important insight for the proof is that, for every state reachable by α from σ, the list of changes of f compared to $f(\vec{o})$ at σ is always finite after finitely many transitions of quantified state change with finite support (see end of Section 5). Consequently, the list of changes can always be encoded by one (finite) real number according to Lemma 8.3.

Using the auxiliary definition $\theta = is_f(\mathfrak{I},\vec{o})$, we characterize cases 1 and 2, that is $\downarrow\mathfrak{I}$ and $@\,\mathfrak{I}\,\phi$ by the following FOQD formulas:

$$\downarrow\mathfrak{I} \equiv \bigwedge_{f\in\Sigma_b} \forall\vec{o}:S_f\ f(\vec{o}) = is_f(\mathfrak{I},\vec{o}) \qquad \text{where } S_f \text{ is the sort of the arguments of } f$$

$$@\,\mathfrak{I}\,\phi \equiv \langle \forall i:C\ \forall u:\mathbb{R}\ f(i)' = u\rangle(\phi \wedge \downarrow\mathfrak{I})$$

The definitions do not need recursion, so that we can consider occurrences of the defined notations as syntactic abbreviations for quantified variables satisfying the respective definitions (like for Lemma 8.3).

Case 1: The characterization for $\downarrow\mathfrak{I}$ is defined as a conjunction over all relevant function symbols $f \in \Sigma_b$ asserting that the value $f(\vec{o})$ of f at each position $\vec{o}$ of the sort S_f of f is identical to the corresponding value $is_f(\mathfrak{I},\vec{o})$ characterized by $\mathfrak{I}$.

Case 2: The characterization for $@\,\mathfrak{I}\,\phi$ uses a quantified differential equation with a variable u that only occurs on the right hand side and thus changes f at all positions i

with an arbitrary slope u. The $@\,\mathfrak{I}\,\phi$ characterization then checks if the appropriate state characterized by $\mathfrak{I}$ has been reached using $\downarrow\mathfrak{I}$ and further expresses that ϕ holds at this state. By case 1, we know that $\downarrow\mathfrak{I}$ holds in at most one of the states reachable by α from σ. In the quantified differential equation system for $@\,\mathfrak{I}\,\phi$, the second quantified variable u amounts to nondeterministically specifying a slope u for each $f(i)$. Unlike i, quantified variable u only occurs on the right hand side of the quantified differential equation. Consequently, the semantics (case 2 of the transition relation $\rho(\alpha)$ defined in Section 4) defines the states corresponding to *all choices* for u to be reachable. These respective choices for u include the choice that leads to the state characterized by $\downarrow\mathfrak{I}$, e.g., by choosing slope $u := is_f(\mathfrak{I},i) - f(i)$ for each i and evolving for 1 time unit. To simplify notation, we define $@\,\mathfrak{I}\,\phi$ only for $\Sigma_b = \{f\}$. The construction is repeated accordingly (by nesting modalities) for each $f \in \Sigma_b$, which are finitely many. The createdness flag $\mathsf{E}(\cdot)$ needs to be part of Σ_b so that object creation is taken care of on the fly. □

8.3. **Expressibility and Rendition of Quantified Hybrid Program Semantics.** In order to show that QdL is sufficiently expressive to state the invariants and variants that are needed for proving valid statements about QHP loops with *ind* and *con*, we prove an expressibility result. We give a constructive proof that the state transition relation of QHPs is definable in FOQD, i.e., there is a FOQD-formula $\mathcal{R}_\alpha(\mathfrak{I})$ characterizing the state transitions of quantified hybrid program α from the current state to the state characterized by $\mathfrak{I}$ (a real variable that characterizes a state by way of Lemma 8.4). For this, we need to characterize the dynamics of QHPs, which are dynamic distributed hybrid processes with repetitively evolving discrete, continuous, structural, and dimension-changing dynamics, equivalently by quantified differential equations in FOQD.

Lemma 8.5. *(*Program rendition*). For every QHP α with symbols among a finite set $\Sigma_b \supseteq \{\mathsf{E}(\cdot)\}$ there is a FOQD-formula $\mathcal{R}_\alpha(\mathfrak{I})$ with one additional free variable $\mathfrak{I}$ of sort $\mathbb{R}$ such that*

$$\vDash \mathcal{R}_\alpha(\mathfrak{I}) \leftrightarrow \langle\alpha\rangle\downarrow\mathfrak{I}$$

$$\begin{aligned}
\mathcal{R}_{\forall i:C\ f(\vec{s}):=\theta}(\mathfrak{I}) &\equiv \forall\vec{o}{:}\,S_f \\
&\quad \big(\mathsf{if}\ \exists i:C\ \vec{o}=\vec{s}\ \mathsf{then}\ \exists i:C\ (\vec{o}=\vec{s}\wedge\theta=is_f(\mathfrak{I},\vec{o}))\ \mathsf{else}\ f(\vec{o})=is_f(\mathfrak{I},\vec{o})\ \mathsf{fi}\big) \\
&\quad \wedge \bigwedge_{g\in\Sigma_b\setminus\{f\}} \forall\vec{o}{:}\,S_g\ g(\vec{o})=is_g(\mathfrak{I},\vec{o}) \\
\mathcal{R}_{\forall i:C\ f(\vec{s})'=\theta}(\mathfrak{I}) &\equiv \langle\forall i:C\ f(\vec{s})'=\theta\rangle\downarrow\mathfrak{I} \\
\mathcal{R}_{\forall i:C\ f(\vec{s})'=\theta\,\&\,\chi}(\mathfrak{I}) &\equiv \langle\forall i:C\ f(\vec{s})'=\theta\,\&\,\chi\rangle\downarrow\mathfrak{I} \\
\mathcal{R}_{?\chi}(\mathfrak{I}) &\equiv \chi\wedge\downarrow\mathfrak{I} \\
\mathcal{R}_{\beta\cup\gamma}(\mathfrak{I}) &\equiv \mathcal{R}_\beta(\mathfrak{I})\vee\mathcal{R}_\gamma(\mathfrak{I}) \\
\mathcal{R}_{\beta;\gamma}(\mathfrak{I}) &\equiv \exists\mathfrak{B}\,(\mathcal{R}_\beta(\mathfrak{B})\wedge @\,\mathfrak{B}\,\mathcal{R}_\gamma(\mathfrak{I})) \\
\mathcal{R}_{\beta^*}(\mathfrak{I}) &\equiv \exists\mathfrak{B}\,\exists n:\mathbb{N}\,\big(\downarrow\mathfrak{B}_1^{(n)}\wedge\mathfrak{B}_n^{(n)}=\mathfrak{I}\wedge\forall i:\mathbb{N}\,(1\le i<n\to @\,\mathfrak{B}_i^{(n)}\,\mathcal{R}_\beta(\mathfrak{B}_{i+1}^{(n)}))\big)
\end{aligned}$$

Figure 7: Explicit rendition of QHP transition semantics in FOQD

Proof. The program rendition is defined inductively in Fig. 7. The characterization of quantified assignments is a variation of the characterization of $\downarrow\mathfrak{I}$ from the proof of Lemma 8.4. The only difference is that the value θ is used instead of $f(\vec{o})$ for positions $\vec{o}$ that are affected by the quantified state change, i.e., $\vec{o}$ is of the form $\vec{s}$ for some i (where the quantified assignment matches as expressed by $\exists i : C\ \vec{o} = \vec{s}$). Quantified differential equations give FOQD-formulas already, because $\downarrow\mathfrak{I}$ is a FOQD-formula, hence no further reduction is necessary.

With a finite formula, the characterization of repetition $\mathcal{R}_{\beta^*}(\vec{v})$ in FOQD needs to capture arbitrarily long sequences of intermediate first-order states and the correct transition between successive states of such a sequence. To achieve this with first-order quantifiers, we use the real Gödel encoding from Lemma 8.3 in Fig. 7 along with the first-order state identification from Lemma 8.4 to map unbounded sequences of real first-order states reversibly to a single real variable $\mathfrak{B}$, which can be quantified over in first-order logic and identify a first-order state with it by Lemma 8.4. □

Using the QHP rendition from Lemma 8.5 to characterize modalities, we prove that every QdL formula can be expressed equivalently in FOQD by structural induction.

Lemma 8.6. *(*Expressibility*).* QdL *is* expressible *in FOQD: for all* QdL *formulas* $\phi \in \mathrm{Fml}$ *there is a FOQD-formula* $\phi^\flat \in \mathrm{Fml}_{FOQD}$ *that is equivalent, i.e.,* $\vDash \phi \leftrightarrow \phi^\flat$. *The converse holds trivially.*

Proof. The proof follows an induction on the structure of formula ϕ for which it is imperative to find an equivalent $\phi^\flat$ in FOQD. Observe that the construction of $\phi^\flat$ from ϕ is effective.

0. If ϕ is a first-order formula, then $\phi^\flat := \phi$ already is a FOQD-formula such that nothing has to be shown.
(1) If ϕ is of the form $\varphi \vee \psi$, then by induction hypothesis there are FOQD-formulas $\varphi^\flat, \psi^\flat$ such that $\vDash \varphi \leftrightarrow \varphi^\flat$ and $\vDash \psi \leftrightarrow \psi^\flat$, from which we can conclude by congruence that $\vDash (\varphi \vee \psi) \leftrightarrow (\varphi^\flat \vee \psi^\flat)$ giving $\vDash \phi \leftrightarrow \phi^\flat$ by choosing $\varphi^\flat \vee \psi^\flat$ for $\phi^\flat$. Likewise reasoning concludes the other propositional connectives or quantifiers.
(2) The case where ϕ is of the form $\langle\alpha\rangle\psi$ is a consequence of the characterization of the semantics of QHPs in FOQD. The expressibility conjecture holds by induction hypothesis using the equivalence of explicit QHP renditions from Lemma 8.5:
$$\vDash\ \langle\alpha\rangle\psi \leftrightarrow \exists\mathfrak{I}\,(\mathcal{R}_\alpha(\mathfrak{I}) \wedge @\,\mathfrak{I}\,\psi^\flat)\ .$$
(3) The case where ϕ is $[\alpha]\psi$ is again a consequence of Lemma 8.5:
$$\vDash\ [\alpha]\psi \leftrightarrow \forall\mathfrak{I}\,(\mathcal{R}_\alpha(\mathfrak{I}) \rightarrow @\,\mathfrak{I}\,\psi^\flat)\ .$$

□

8.4. **Relative Completeness of First-order Assertions.** As special cases of Theorem 8.2, we first prove relative completeness for first-order assertions about QHPs. These first-order cases constitute the basis for the general completeness proof for arbitrary QdL formulas.

In the sequel, we use the notation $\vdash_{\mathcal{D}} \phi$ to indicate that a QdL formula ϕ is derivable from a set of FOQD-tautologies, which is equivalent to saying that ϕ is derivable in the QdL calculus augmented with a single *oracle axiom* $\mathcal{D}$, that gives all valid FOQD-instances. The QdL calculus contains a complete calculus for propositional logic and for many-sorted

first-order logic. We implicitly use simple propositional reasoning (using the *cut*-rule) to glue together subproofs propositionally.

Proposition 8.7. *(*Relative completeness of first-order safety*). For every QHP α and all FOQD formulas F, G*

$$\vDash F \to [\alpha]G \text{ implies } \vdash_{\mathcal{D}} F \to [\alpha]G \ .$$

Proof. We generalize the relative completeness proof by Cook [Coo78] to $\mathsf{Qd}\mathcal{L}$ and follow an induction on the structure of program α. In the following, *IH* is short for the induction hypothesis.

(1) The cases where α is of the form $f(\vec{s}) := \theta$, $?\chi$, $\beta \cup \gamma$, or $\beta;\gamma$ are consequences of the soundness of the rules [;], [$\cup$], [?], and [:=], which are equivalence rules. Consequently, whenever their conclusion is valid, their premise is valid and of smaller complexity (the programs get simpler), hence the premise is derivable by IH. Thus, we can derive $F \to [\alpha]G$ by applying the respective rule. For [:=] and [:], respectively, the premise is simpler because the quantified assignment is only applied to structurally simpler expressions ($\vec{u}$) in the premise than in the conclusion ($f(\vec{u})$) while the program stays the same. For nondeterministic assignments, the reasoning is similar using equivalence rule [:$*$] instead of [:=]. Again, the premise is valid, and already a FOQD formula, hence derivable as an $\mathcal{D}$ axiom directly. A formal rewrite proof along these lines is a simple modification of prior work [BP06]. We explicitly show the proof for $\beta;\gamma$ as it contains an extra twist.

(2) $\vDash F \to [\beta;\gamma]G$, which implies $\vDash F \to [\beta][\gamma]G$. By Lemma 8.6, there is a FOQD-formula $G^\flat$ such that $\vDash G^\flat \leftrightarrow [\gamma]G$. From the validity of $\vDash F \to [\beta]G^\flat$, we can conclude by IH that $\vdash_{\mathcal{D}} F \to [\beta]G^\flat$ is derivable. Similarly, because of $\vDash G^\flat \to [\gamma]G$, we conclude $\vdash_{\mathcal{D}} G^\flat \to [\gamma]G$ by IH. With an application of $[]gen$, the latter derivation can be extended to a derivation of $\vdash_{\mathcal{D}} [\beta]G^\flat \to [\beta][\gamma]G$. Combining the above derivations propositionally by a cut with $[\beta]G^\flat$, we can derive $\vdash_{\mathcal{D}} F \to [\beta][\gamma]G$, from which [;] yields $\vdash_{\mathcal{D}} F \to [\beta;\gamma]G$ as desired.

(3) $\vDash F \to [\forall i : C\ f(\vec{s})' = \theta \,\&\, \chi]G$ is a FOQD-formula and hence derivable as a $\mathcal{D}$ axiom directly.

(4) $\vDash F \to [\beta^*]G$ can be derived by induction. For this, we define the invariant as a FOQD encoding of the statement that all potential poststates of β^* satisfy G according to Lemma 8.6:

$$\phi \equiv ([\beta^*]G)^\flat \equiv \forall \mathfrak{I}\, (\mathcal{R}_{\beta^*}(\mathfrak{I}) \to @\,\mathfrak{I}\, G) \ .$$

Since $F \to \phi$ and $\phi \to G$ are valid FOQD-formulas according to the semantics, they are derivable by $\mathcal{D}$. By $[]gen$, $\vdash_{\mathcal{D}} [\beta^*]\phi \to [\beta^*]G$ is derivable from the latter. Likewise, $\phi \to [\beta]\phi$ is valid according to the semantics of repetition, thus derivable by IH, since β is less complex. Now *ind* yields $\vdash_{\mathcal{D}} \phi \to [\beta^*]\phi$. Combining the above derivations propositionally by a cut with $[\beta^*]\phi$ and ϕ yields $\vdash_{\mathcal{D}} F \to [\beta^*]G$.

□

Proposition 8.8. *(*Relative completeness of first-order liveness*). For each QHP α and all FOQD-formulas F, G*

$$\vDash F \to \langle\alpha\rangle G \text{ implies } \vdash_{\mathcal{D}} F \to \langle\alpha\rangle G \ .$$

Proof. We generalize the integer arithmetic completeness proof by Harel [Har79] to the hybrid case. Most cases of the proof are simple adaptations of the corresponding cases in Proposition 8.7. What remains to be shown is the case of repetitions. Assume that $\vDash F \to \langle\beta^*\rangle G$. To derive this formula by *con*, we use a FOQD-formula $\varphi(n)$ as a variant expressing that, after n iterations, β can lead to a state satisfying G. This formula is obtained from Lemma 8.5-8.6 as $(\langle\beta^*\rangle G)^\flat \equiv \exists\mathfrak{I}\,(\mathcal{R}_{\beta^*}(\mathfrak{I}) \wedge @\,\mathfrak{I}\,G)$, *except* that the quantifier on the repetition count n is removed such that n becomes a free variable (plus index shifting to count repetitions):

$$\varphi(n-1) \;\equiv\; \exists\mathfrak{B}\,\big(\downarrow\mathfrak{B}_1^{(n)} \wedge \mathfrak{B}_n^{(n)} = \mathfrak{I} \wedge \forall i:\mathbb{N}\;(1 \le i < n \to @\,\mathfrak{B}_i^{(n)}\,\mathcal{R}_\beta(\mathfrak{B}_{i+1}^{(n)})) \wedge @\,\mathfrak{I}\,G\big)\ .$$

By Lemma 8.3, $\varphi(n)$ can only hold true if n is a natural number.

According to the loop semantics, $\vDash n > 0 \wedge \varphi(n) \to \langle\beta\rangle\varphi(n-1)$ is valid by construction: If $n > 0$ is a natural number then so is $n-1$, and if β reaches G after n repetitions, then, after executing β once, $n-1$ repetitions of β reach G. By IH, this formula is derivable, since β contains less loops. We have derived $\vdash_{\mathcal{D}} n > 0 \wedge \varphi(n) \to \langle\beta\rangle\varphi(n-1)$. Thus $\vdash_{\mathcal{D}} \exists v\,\varphi(v) \to \langle\beta^*\rangle\exists v{\le}0\,\varphi(v)$ by *con*. It only remains to show that the antecedent is derivable from F and that $\langle\beta^*\rangle G$ is derivable from the succedent. From our assumption, we conclude that the following are valid FOQD-formulas, hence $\mathcal{D}$-axioms:

- $\vDash F \to \exists v\,\varphi(v)$, because $\vDash F \to \langle\beta^*\rangle G$, and
- $\vDash (\exists v{\le}0\,\varphi(v)) \to G$, because $v{\le}0$ and the fact, that, by Lemma 8.3, $\varphi(v)$ only holds true for natural numbers, imply $\varphi(0)$. Further, $\varphi(0)$ entails G, because zero repetitions of β have no effect.

We extend the latter derivation to $\vdash_{\mathcal{D}} \langle\beta^*\rangle\exists v{\le}0\,\varphi(v) \to \langle\beta^*\rangle G$ by $\langle\rangle gen$. Now, the above derivations can be combined propositionally by a cut with $\langle\beta^*\rangle\exists v{\le}0\,\varphi(v)$ and with $\exists v\,\varphi(v)$ to yield $\vdash_{\mathcal{D}} F \to \langle\beta^*\rangle G$. □

8.5. **Relative Completeness of the QdL Calculus.** Having succeeded with the proofs of the above statements about parts of the completeness proof, we can finish the proof of Theorem 8.2.

Proof of Theorem 8.2. The proof follows a basic structure similar to that of Harel's proof for the discrete case [Har79, Theorem 3.1]. We have to show that every valid QdL formula ϕ can be proven from FOQD axioms within the QdL calculus: from $\vDash \phi$ we have to prove $\vdash_{\mathcal{D}} \phi$. The proof proceeds as follows: By propositional recombination, we inductively identify fragments of ϕ that correspond to $\phi_1 \to [\alpha]\phi_2$ or $\phi_1 \to \langle\alpha\rangle\phi_2$ logically. Next, we express subformulas ϕ_i equivalently in FOQD by Lemma 8.6, and use Proposition 8.7 and 8.8 to resolve these first-order safety or liveness assertions. Finally, we prove that the original QdL formula can be re-derived from the subproofs.

We can assume ϕ to be given in conjunctive normal form by appropriate propositional reasoning. In particular, we assume that negations are pushed inside over modalities using the dualities $\neg[\alpha]\phi \equiv \langle\alpha\rangle\neg\phi$ and $\neg\langle\alpha\rangle\phi \equiv [\alpha]\neg\phi$. The remainder of the proof follows an induction on a measure $|\phi|$ defined as the number of modalities in ϕ. For a uniform proof, we assume real quantifiers to be abbreviations for modal formulas by $\exists x:\mathbb{R}\ \phi \equiv \langle x' = 1\rangle\phi \vee \langle x' = -1\rangle\phi$ and $\forall x:\mathbb{R}\ \phi \equiv [x' = 1]\phi \wedge [x' = -1]\phi$. Following either $x' = 1$ or $x' = -1$, we can reach any real number as a value for x. Similarly, we assume quantifiers for sort $C \neq \mathbb{R}$ to be abbreviations for modal formulas by $\exists x:C\ \phi \equiv \langle\forall j:C\ x := j\rangle\phi$

and $\forall x:C\ \phi \equiv [\forall j:C\ x:=j]\phi$. We can obtain any object of sort C by an appropriate choice of j. Now the proof is by induction on the measure $|\phi|$ of ϕ.

0. $|\phi|=0$ then ϕ is a first-order formula, hence derivable by $\mathcal{D}$.
(1) ϕ is of the form $\neg\phi_1$, then ϕ_1 is first-order, as we assumed negations to be pushed inside. Hence, case 0 applies: $|\phi|=0$.
(2) ϕ is of the form $\phi_1\wedge\phi_2$, then individually deduce the simpler proofs for $\vdash_{\mathcal{D}}\phi_1$ and $\vdash_{\mathcal{D}}\phi_2$ by IH, which can be combined by $\wedge$r.
(3) ϕ is a disjunction and—without loss of generality—has one of the following forms (otherwise use associativity and commutativity to select a different order for the disjunction):

$$\phi_1\vee[\alpha]\phi_2$$
$$\phi_1\vee\langle\alpha\rangle\phi_2$$

As a unified notation for those cases we use $\phi_1\vee\langle\!\langle\alpha\rangle\!\rangle\phi_2$. Then, $|\phi_2|<|\phi|$, since ϕ_2 has less modalities. Likewise, $|\phi_1|<|\phi|$ because $\langle\!\langle\alpha\rangle\!\rangle\phi_2$ contributes one modality to $|\phi|$ that is not part of ϕ_1.

According to Lemma 8.6 there are FOQD-formulas $\phi_1^\flat,\phi_2^\flat$ that satisfy $\models\phi_i\leftrightarrow\phi_i^\flat$ for $i=1,2$. By congruence, the validity $\models\phi$ yields that $\models\phi_1^\flat\vee\langle\!\langle\alpha\rangle\!\rangle\phi_2^\flat$, which directly implies $\models\neg\phi_1^\flat\to\langle\!\langle\alpha\rangle\!\rangle\phi_2^\flat$. Then by Proposition 8.7 or 8.8, respectively, we can derive

$$\vdash_{\mathcal{D}}\neg\phi_1^\flat\to\langle\!\langle\alpha\rangle\!\rangle\phi_2^\flat\ . \tag{8.1}$$

Further $\models\phi_1\leftrightarrow\phi_1^\flat$ implies $\models\neg\phi_1\to\neg\phi_1^\flat$, which is derivable by IH, because $|\phi_1|<|\phi|$. We combine the resulting derivation $\vdash_{\mathcal{D}}\neg\phi_1\to\neg\phi_1^\flat$, with (8.1) by a cut with $\neg\phi_1^\flat$ to obtain

$$\vdash_{\mathcal{D}}\neg\phi_1\to\langle\!\langle\alpha\rangle\!\rangle\phi_2^\flat\ . \tag{8.2}$$

Likewise $\models\phi_2\leftrightarrow\phi_2^\flat$ implies $\models\phi_2^\flat\to\phi_2$, which is derivable by IH, as $|\phi_2|<|\phi|$. We can extend the derivation of $\vdash_{\mathcal{D}}\phi_2^\flat\to\phi_2$ $\vdash_{\mathcal{D}}\langle\!\langle\alpha\rangle\!\rangle\phi_2^\flat\to\langle\!\langle\alpha\rangle\!\rangle\phi_2$ by $[]gen$–$\langle\rangle gen$. Finally we combine the latter propositionally with (8.2) by a cut with $\langle\!\langle\alpha\rangle\!\rangle\phi_2^\flat$ to derive $\vdash_{\mathcal{D}}\neg\phi_1\to\langle\!\langle\alpha\rangle\!\rangle\phi_2$, from which $\vdash_{\mathcal{D}}\phi_1\vee\langle\!\langle\alpha\rangle\!\rangle\phi_2$ can be obtained, again using *cut*, to complete the proof.

□

9. Distributed Car Control Verification

With the Q$\mathsf{d}\mathcal{L}$ calculus and the compatibility condition $\mathcal{M}(i,j)$ from eqn. (3.2), we can easily prove collision freedom, i.e., formula (5.2), in the distributed car control system (5.4):

$$(\forall i,j:C!\ \mathcal{M}(i,j))\to$$
$$[(n:=\mathsf{new}\,C;?\forall i:C!\ \mathcal{M}(i,n);\forall i:C!\ (x(i)''=a(i)))^*]\ \forall i{\neq}j:C!\ x(i){\neq}x(j) \tag{9.1}$$

The biggest challenge in the proof of this Q$\mathsf{d}\mathcal{L}$ formula is that it involves continuous dynamics, discrete dynamics, and dimensional dynamics, and that all parts of the system need to interact safely for the system to stay collision-free. In particular, formula (9.1) states a safety property of unboundedly many cars driving on a road, where an unbounded number of new cars may additionally appear dynamically during the evolution of the system. See Fig. 8 for a formal Q$\mathsf{d}\mathcal{L}$ proof of this Q$\mathsf{d}\mathcal{L}$ formula, which proves collision freedom despite dynamic appearance of new cars.

$$\dfrac{\dfrac{*}{\mathrm{i}\forall\ ,\mathcal{M}(i,n),t{\geq}0{\to}\mathcal{S}_t\mathcal{M}(i,n)}}{\forall\mathrm{l}\ \ldots,\forall i:C!\ \mathcal{M}(i,n),t{\geq}0{\to}\mathcal{S}_t\mathcal{M}(i,n)}\qquad \dfrac{*}{\mathrm{i}\forall\ \mathcal{M}(i,j),,t{\geq}0{\to}\mathcal{S}_t\mathcal{M}(i,j)}$$

$$\dfrac{}{\forall\mathrm{l}\ \forall i,j:C!\ \mathcal{M}(i,j),\ldots,t{\geq}0{\to}\mathcal{S}_t\mathcal{M}(i,j)}$$

$$\wedge\mathrm{r}\ \dfrac{}{\forall i,j:C!\ \mathcal{M}(i,j),\forall i:C!\ \mathcal{M}(i,n),t{\geq}0{\to}\mathcal{S}_t\mathcal{M}(i,n)\wedge\mathcal{S}_t\mathcal{M}(i,j)}$$

$$[:=]\ \dfrac{}{\forall i,j:C!\ \mathcal{M}(i,j),\forall i:C!\ \mathcal{M}(i,n),t{\geq}0{\to}[\forall i:C!{\cup}\{n\}\ \mathcal{S}_t(i)](\mathcal{M}(i,n)\wedge\mathcal{M}(i,j))}$$

$$\forall\mathrm{r}\ \dfrac{}{\forall i,j:C!\ \mathcal{M}(i,j),\forall i:C!\ \mathcal{M}(i,n),t{\geq}0{\to}[\forall i:C!{\cup}\{n\}\ \mathcal{S}_t(i)]\forall i,j:C!\ (\mathcal{M}(i,n)\wedge\mathcal{M}(i,j))}$$

$$\nu\forall\ \dfrac{}{\forall i,j:C!\ \mathcal{M}(i,j),\forall i:C!\ \mathcal{M}(i,n),t{\geq}0{\to}[\forall i:C!{\cup}\{n\}\ \mathcal{S}_t(i)][\mathbb{E}(n)]\forall i,j:C!\ \mathcal{M}(i,j)}$$

$$\nu A\ \dfrac{}{\forall i,j:C!\ \mathcal{M}(i,j),\forall i:C!\ \mathcal{M}(i,n),t{\geq}0{\to}[\mathbb{E}(n)][\forall i:C!\ \mathcal{S}_t(i)]\forall i,j:C!\ \mathcal{M}(i,j)}$$

$$\forall\mathrm{r},{\to}\mathrm{r}\ \dfrac{}{\forall i,j:C!\ \mathcal{M}(i,j),\forall i:C!\ \mathcal{M}(i,n){\to}[\mathbb{E}(n)]\forall t{\geq}0\,[\forall i:C!\ \mathcal{S}_t(i)]\forall i,j:C!\ \mathcal{M}(i,j)}$$

$$[']\ \dfrac{}{\forall i,j:C!\ \mathcal{M}(i,j),,\forall i:C!\ \mathcal{M}(i,n){\to}[\mathbb{E}(n)][\forall i:C!\ (x(i)''=a(i))]\forall i,j:C!\ \mathcal{M}(i,j)}$$

$$\nu\forall,\wedge\mathrm{l}\ \dfrac{\forall i,j:C!\ \mathcal{M}(i,j),[\mathbb{E}(n)]\forall i:C!\ \mathcal{M}(i,n){\to}[\mathbb{E}(n)][\forall i:C!\ (x(i)''=a(i))]\forall i,j:C!\ \mathcal{M}(i,j)}{}$$

$${\to}\mathrm{r}\ \dfrac{}{\forall i,j:C!\ \mathcal{M}(i,j){\to}[\mathbb{E}(n)](\forall i:C!\ \mathcal{M}(i,n)\to[\forall i:C!\ (x(i)''=a(i))]\forall i,j:C!\ \mathcal{M}(i,j))}$$

$$[?]\ \dfrac{}{\mathbb{E}(n)=0,\forall i,j:C!\ \mathcal{M}(i,j){\to}[\mathbb{E}(n)][?\forall i:C!\ \mathcal{M}(i,n);\forall i:C!\ (x(i)''=a(i))]\forall i,j:C!\ \mathcal{M}(i,j)}$$

$$new\ \dfrac{}{\forall i,j:C!\ \mathcal{M}(i,j){\to}[n:=\mathsf{new}\,C][?\forall i:C!\ \mathcal{M}(i,n);\forall i:C!\ (x(i)''=a(i))]\forall i,j:C!\ \mathcal{M}(i,j)}$$

$$[;]\ \dfrac{}{\forall i,j:C!\ \mathcal{M}(i,j){\to}[DCCS]\forall i,j:C!\ \mathcal{M}(i,j)}$$

$$ind\ \dfrac{}{\forall i,j:C!\ \mathcal{M}(i,j){\to}[(DCCS)^*]\forall i{\neq}j:C!\ x(i){\neq}x(j)}$$

Figure 8: QdℒL proof for collision freedom in distributed car control with dynamic appearance.

The proof in Fig. 8 uses induction (rule *ind*) with invariant $\forall i,j:C!\ \mathcal{M}(i,j)$. Figure 8 does not show the branch proving that the invariant $\forall i,j:C!\ \mathcal{M}(i,j)$ implies the postcondition $\forall i{\neq}j:C!\ x(i){\neq}x(j)$, which is easy to prove.

The proof step marked by *new* uses the definition of $\mathsf{new}\,C$ from eqn. (5.1). To save space, we abbreviate $[\mathbb{E}(n):=1]$ by $[\mathbb{E}(n)]$ in Fig. 8. The proof uses the derived rules $\nu\forall$ and νA from Section 6 to propagate the effect of object creation on actualist quantifiers and actualist quantified assignments respectively. In rule νA, the shorthand notation $\forall i:C!{\cup}\{n\}\ \mathcal{S}_t(i)$ in the resulting formula indicates that the new object n is also updated according to the solution $\mathcal{S}_t(n)$, not just the previously existing objects ($\forall i:C!\ \mathcal{S}_t(i)$). Here, we abbreviate by $\mathcal{S}_t(i)$ the solution $x(i):=x(i)+v(i)t+\frac{a(i)}{2}t^2, v(i):=v(i)+a(i)t$ of the quantified differential equation $\forall i:C\ x(i)''=a(i)$, which rule $[']$ introduces. For the topmost application of rule $[:=]$, we denote by $\mathcal{S}_t\mathcal{M}(i,j)$ the result of substituting $\forall i:C!\ \mathcal{S}_t(i)$ into $\mathcal{M}(i,j)$ according to rule $[:=]$. In Fig. 8, we leave out some irrelevant formulas, indicated by ellipsis ($\ldots$) or gray print. The proof closes (indicated by $*$) by QE with rule $\mathrm{i}\forall$. Hence, QdℒL formula (9.1) is valid by Theorem 7.1.

In a similar way, the QdℒL proof rules can prove collision freedom in an advanced distributed car control system that has both dynamic appearance of cars on the road as in (5.4) and more flexibility in acceleration and braking choices of the individual cars as in (5.3). For this, we choose a weaker constraint for $\mathcal{M}(i,j)$ that allows cars that move with quite different accelerations, if only the respective safety distances are compatible with the different velocities:

$$\begin{aligned} i\neq j\ \to\big((x(i)<x(j)\wedge v(i)^2<v(j)^2+2b(x(j)-x(i))\wedge v(i)\geq 0\wedge v(j)\geq 0)\\ \vee\ (x(i)>x(j)\wedge v(j)^2<v(i)^2+2b(x(i)-x(j))\wedge v(i)\geq 0\wedge v(j)\geq 0)\big)\end{aligned}$$

With this choice for $\mathcal{M}(i,j)$, the Q$\mathsf{d}\mathcal{L}$ proof calculus can be used to prove the following Q$\mathsf{d}\mathcal{L}$ formula with a proof very similar to that in Fig. 8:

$$\begin{aligned}&\forall i,j:C!\ \mathcal{M}(i,j) \rightarrow \\ &\quad \big[(n := \mathsf{new}\, C;\ ?\forall i:C!\ \mathcal{M}(i,n); \\ &\quad \forall i:C!\ a(i) := \mathsf{if}\, \forall j:C!\ far(i,j)\, \mathsf{then}\, a\, \mathsf{else}\ -b\, \mathsf{fi}; \\ &\quad \tau := 0;\ \forall i:C!\ (x(i)' = v(i), v(i)' = a(i), \tau' = 1 \,\&\, v(i) \geq 0 \wedge \tau \leq \varepsilon))^* \\ &\quad \big]\ \forall i{\neq}j:C!\ x(i){\neq}x(j)\end{aligned} \tag{9.2}$$

The QHP in Q$\mathsf{d}\mathcal{L}$ formula (9.2) allows all cars to change their respective acceleration freely when all other cars are sufficiently far away like in (5.3). For this, we choose a condition characterizing that the distributed car control system stays controllable for at least ε time units (which is the maximum reaction time of the controller):

$$far(i,j) \ \equiv\ x(j) > x(i) \rightarrow x(j) > x(i) + \frac{v(i)^2 - v(j)^2}{2b} + \Big(\frac{a}{b}+1\Big)\Big(\frac{a}{2}\varepsilon^2 + \varepsilon v(i)\Big)$$

The continuous dynamics in (9.2) is bounded by the evolution domain constraint $\tau \leq \varepsilon$ to evolve for at most ε time units, at which point, at the latest, the discrete controllers will have a chance to react to situation changes again (i.e., the control loop repeats). The Q$\mathsf{d}\mathcal{L}$ proof of (9.2) has the same structure as that in Fig. 8 except that the arithmetic is more involved to handle the resulting nonlinear and nonmonotonic arithmetic constraints, see [Pla10d].

For a Q$\mathsf{d}\mathcal{L}$ proof extending the above ideas to a proof of collision-freedom for a more realistic distributed car control system having arbitrarily many cars switching between arbitrarily many lanes with dynamic appearance and disappearance of arbitrarily many cars, we refer to follow-up work [LPN11]. Unlike our simplified system model, this follow-up work does not assume that all cars use the same braking power.

10. Conclusions

We have introduced a formal system model and semantics for dynamic distributed hybrid systems together with a compositional verification logic and proof calculus. We believe this is the *first formal verification approach for distributed hybrid dynamics*, where structure and dimension of the system can evolve jointly with the discrete and continuous dynamics. Our approach handles *distributed hybrid systems* with interacting discrete dynamics, continuous dynamics, structural dynamics, and dimensional dynamics. We have proven our calculus to be a *sound and complete axiomatization* relative to quantified differential equations. Our calculus proves collision avoidance in distributed car control with dynamic appearance of new cars on the road, which is out of scope for other approaches.

Future work includes full modular concurrency in distributed hybrid systems, which is already challenging in discrete programs.

Acknowledgement

I thank Frank Pfenning for his helpful comments and the reviewers for their feedback.

References

[ACHH92] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Grossman et al. [GNRR93], pages 209–229.

[AdBO10] Krzysztof R. Apt, Frank S. de Boer, and Ernst-Rüdiger Olderog. *Verification of Sequential and Concurrent Programs*. Springer, 3rd edition, 2010.

[AHS96] Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors. *Hybrid Systems III: Verification and Control, Proceedings*, volume 1066 of *LNCS*. Springer, 1996.

[AL01] Paul C. Attie and Nancy A. Lynch. Dynamic input/output automata: A formal model for dynamic systems. In Kim Guldstrand Larsen and Mogens Nielsen, editors, *CONCUR*, volume 2154 of *LNCS*, pages 137–151. Springer, 2001.

[BBM98] Michael S. Branicky, Vivek S. Borkar, and Sanjoy K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE T. Automat. Contr.*, 43(1):31–45, 1998.

[Bog95] Vladimir I. Bogachev. Deterministic and stochastic differential equations in infinite-dimensional spaces. *Acta Appl. Math.*, 40(1):25–93, Jul 1995.

[BP06] Bernhard Beckert and André Platzer. Dynamic logic with non-rigid functions: A basis for object-oriented program verification. In Ulrich Furbach and Natarajan Shankar, editors, *IJCAR*, volume 4130 of *LNCS*, pages 266–280. Springer, 2006.

[Bra95] Michael S. Branicky. General hybrid dynamical systems: Modeling, analysis, and control. In Alur et al. [AHS96], pages 186–200.

[CH91] George E. Collins and Hoon Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *J. Symb. Comput.*, 12(3):299–328, 1991.

[CJR95] Zhou Chaochen, Wang Ji, and Anders P. Ravn. A formal description of hybrid systems. In Alur et al. [AHS96], pages 511–530.

[Coo78] Stephen A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM J. Comput.*, 7(1):70–90, 1978.

[DGV96] Akash Deshpande, Aleks Göllü, and Pravin Varaiya. SHIFT: A formalism and a programming language for dynamic networks of hybrid automata. In Panos J. Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems*, volume 1273 of *LNCS*, pages 113–133. Springer, 1996.

[DMC05] Gilles Dowek, César Muñoz, and Víctor A. Carreño. Provably safe coordinated strategy for distributed conflict resolution. In *Proceedings of the AIAA Guidance Navigation, and Control Conference and Exhibit 2005, AIAA-2005-6047*, 2005.

[Fit96] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, New York, 2nd edition, 1996.

[FM99] Melvin Fitting and Richard L. Mendelsohn. *First-Order Modal Logic*. Kluwer, Norwell, MA, USA, 1999.

[GNRR93] Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors. *Hybrid Systems*, volume 736 of *LNCS*. Springer, 1993.

[Göd31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Mon. hefte Math. Phys.*, 38:173–198, 1931.

[Har79] David Harel. *First-Order Dynamic Logic*. Springer, New York, 1979.

[Hen96] Thomas A. Henzinger. The theory of hybrid automata. In *LICS*, pages 278–292, Los Alamitos, 1996. IEEE Computer Society.

[HESV91] Ann Hsu, Farokh Eskafi, Sonia Sachs, and Pravin Varaiya. Design of platoon maneuver protocols for IVHS. PATH Research Report UCB-ITS-PRR-91-6, Institute of Transportation Studies, University of California, Berkeley, 1991.

[HKT00] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic logic*. MIT Press, Cambridge, 2000.

[Hoa69] Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.

[HS94] Reiner Hähnle and Peter H. Schmitt. The liberalized δ-rule in free variable semantic tableaux. *J. Autom. Reasoning*, 13(2):211–221, 1994.

[HT06] João P. Hespanha and Ashish Tiwari, editors. *Hybrid Systems: Computation and Control, 9th International Workshop, HSCC 2006, Santa Barbara, CA, USA, March 29-31, 2006, Proceedings*, volume 3927 of *LNCS*. Springer, 2006.

[Koz97] Dexter Kozen. Kleene algebra with tests. *ACM Trans. Program. Lang. Syst.*, 19(3):427–443, 1997.

[KSPL06] Fabian Kratz, Oleg Sokolsky, George J. Pappas, and Insup Lee. R-Charon, a modeling language for reconfigurable hybrid systems. In Hespanha and Tiwari [HT06], pages 392–406.

[LPN11] Sarah M. Loos, André Platzer, and Ligia Nistor. Adaptive cruise control: Hybrid, distributed, and now formally verified. In Michael Butler and Wolfram Schulte, editors, *FM*, volume 6664 of *LNCS*, pages 42–56. Springer, 2011.

[Lyn96] Nancy Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[Mor87] Michał Morayne. On differentiability of Peano type functions. *Colloquium Mathematicum*, LIII:129–132, 1987.

[MS06] José Meseguer and Raman Sharykin. Specification and analysis of distributed object-based stochastic hybrid systems. In Hespanha and Tiwari [HT06], pages 460–475.

[Pla08a] André Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reas.*, 41(2):143–189, 2008.

[Pla08b] André Platzer. *Differential Dynamic Logics: Automated Theorem Proving for Hybrid Systems*. PhD thesis, Department of Computing Science, University of Oldenburg, Dec 2008. Appeared with Springer.

[Pla10a] André Platzer. Differential-algebraic dynamic logic for differential-algebraic programs. *J. Log. Comput.*, 20(1):309–352, 2010.

[Pla10b] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010.

[Pla10c] André Platzer. Quantified differential dynamic logic for distributed hybrid systems. In Anuj Dawar and Helmut Veith, editors, *CSL*, volume 6247 of *LNCS*, pages 469–483. Springer, 2010.

[Pla10d] André Platzer. Quantified differential dynamic logic for distributed hybrid systems. Technical Report CMU-CS-10-126, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2010.

[Pla11] André Platzer. Quantified differential invariants. In Emilio Frazzoli and Radu Grosu, editors, *HSCC*, pages 63–72. ACM, 2011.

[Pla12] André Platzer. The complete proof theory of hybrid systems. In *LICS*. IEEE Computer Society, 2012.

[Pra76] Vaughan R. Pratt. Semantical considerations on Floyd-Hoare logic. In *FOCS*, pages 109–121. IEEE, 1976.

[PSFB07] L. Pallottino, V. G. Scordio, E. Frazzoli, and A. Bicchi. Decentralized cooperative policy for conflict resolution in multi-vehicle systems. *IEEE Trans. on Robotics*, 23(6):1170–1183, 2007.

[Rou04] William C. Rounds. A spatial logic for the hybrid π-calculus. In Rajeev Alur and George J. Pappas, editors, *HSCC*, volume 2993 of *LNCS*, pages 508–522. Springer, 2004.

[Rüm06] Philipp Rümmer. Sequential, parallel, and quantified updates of first-order structures. In Miki Hermann and Andrei Voronkov, editors, *LPAR*, volume 4246 of *LNCS*, pages 422–436. Springer, 2006.

[SRS+06] Raja Sengupta, Shahram Rezaei, Steven E. Shladover, Delphine Cody, Susan Dickey, and Hariharan Krishnan. Cooperative collision warning systems: Concept definition and experimental implementation. PATH Research Report UCB-ITS-PRR-2006-6, Institute of Transportation Studies, University of California, Berkeley, 2006.

[Tar51] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, 2nd edition, 1951.

[vBMR+06] D. A. van Beek, Ka L. Man, Michel A. Reniers, J. E. Rooda, and Ramon R. H. Schiffelers. Syntax and consistent equation semantics of hybrid Chi. *J. Log. Algebr. Program.*, 68(1-2):129–210, 2006.

[Wal98] Wolfgang Walter. *Ordinary Differential Equations*. Springer, 1998.

[ZPC10] Paolo Zuliani, André Platzer, and Edmund M. Clarke. Bayesian statistical model checking with application to Simulink/Stateflow verification. In Karl Henrik Johansson and Wang Yi, editors, *HSCC*, pages 243–252. ACM, 2010.

[ZRH92] Chaochen Zhou, Anders P. Ravn, and Michael R. Hansen. An extended duration calculus for hybrid real-time systems. In Grossman et al. [GNRR93], pages 36–59.