Differential Refinement Logic

Thesis Proposal

Sarah M. Loos

December 2014

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
sloos@cs.cmu.edu

Thesis Committee:
André Platzer (chair)
Bruce Krogh
Frank Pfenning
Dexter Kozen (Cornell University)
George Pappas (University of Pennsylvania)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Sarah M. Loos is supported by a Department of Energy Computation Sciences Graduate Fellowship and a National Science Foundation Graduate Research Fellowship.
This material is based upon work supported by the National Science Foundation under NSF CAREER Award CNS-1054246, NSF EXPEDITION CNS-0926181, grant no. CNS-0931985, CNS-1035813, ONR N00014-10-1-0188, and CNS-1035800. This research was also supported by the US Department of Transportation's University Transportation Center's TSET grant, award no. DTRT12GUTC11. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution or government. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of any sponsoring institution or government.

Keywords: Differential Refinement Logic, Differential Dynamic Logic, Cyber-Physical Systems, Hybrid Systems, Distributed Hybrid Systems, Formal Verification, Formal Methods, Theorem Proving

Abstract

This thesis is focused on formal verification of cyber-physical systems. Cyber-physical systems (CPSs), such as computer-controlled cars, airplanes or robots play an increasingly crucial role in our daily lives. They are systems that we bet our lives on, so they need to be safe. However, ensuring that CPSs are safe is an intellectual challenge due to their intricate interactions of complex control software with physical behavior. Formal verification techniques, such as theorem proving, can provide strong guarantees for these systems by returning proofs that safety is preserved throughout the continuously infinite space of their possible behaviors.

Previously completed work has provided: the first formal verification of distributed car control; the first formal verification of distributed, flyable, collision avoidance protocols for aircraft; and an exploration of control choices within a well-defined safety envelope. Each of these systems presented new verification challenges and required new techniques for proving safety. However, the authors identified a unifying hurdle for each case study that has thus far remained unaddressed: *it is difficult to compare hybrid systems, even when their behaviors are very similar.*

We introduce *differential refinement logic* (dR$\mathcal{L}$), a logic with first-class support for refinement relations on hybrid systems, and a proof calculus for verifying such relations. dR$\mathcal{L}$ simultaneously solves several seemingly different challenges common in theorem proving for hybrid systems:

1. When hybrid systems are complicated, it is useful to prove properties about simpler and related subsystems before tackling the system as a whole.
2. Some models of hybrid systems can be implementation-specific. Verification can be aided by abstracting the system down to the core components necessary for safety, but only if the relations between the abstraction and the original system can be guaranteed.
3. One approach to taming the complexities of hybrid systems is to start with a simplified version of the system, prove it safe, and then iteratively expand it. However, this approach can be costly, since every iteration has to be proved safe from scratch, unless refinement relations can be leveraged in the proof.
4. When proofs become large, it is difficult to maintain a modular or comprehensible proof structure. By using a refinement relation to arrange proofs hierarchically according to the structure of natural subsystems, we can increase the readability, modularity, and reusability of the resulting proof.

dR$\mathcal{L}$ extends an existing specification and verification language for hybrid systems (differential dynamic logic, d$\mathcal{L}$) by adding a refinement relation to directly compare hybrid systems. This thesis will give a syntax, semantics, and proof calculus for dR$\mathcal{L}$.

We will also demonstrate the usefulness of dR$\mathcal{L}$ on several examples which the author has previously completed. We plan to show that using refinement results in easier and better-structured proofs than those using d$\mathcal{L}$.

Contents

Chapter 1

Introduction

1.1 Proposal Overview

Differential dynamic logic ($\mathsf{d}\mathcal{L}$) is an existing specification and verification logic for hybrid programs [1–4]. We have had many successes using $\mathsf{d}\mathcal{L}$ and its associated proof calculus in proving safety for complicated cyber-physical systems. Some examples of systems that we have modeled as hybrid programs in $\mathsf{d}\mathcal{L}$ and verified with its theorem prover KeYmaera include: distributed car control [5], distributed aircraft control [6], collision avoidance at intersections [7], and designing safe PID controllers for adaptive cruise control [8, 9].

While these examples have been impressive in their own right, we faced many barriers while exploring these case studies and often our verification results relied on complicated and creative proofs [5, 6]. Even more frustrating were the occasions where it was too challenging to verify a system as-is, and so we instead verified a simpler variant and then relied on informal reasoning to relate results back to the original system [7–9]. *We observed that if only we had direct proof support for (automatically) relating two systems, such proofs could be entirely formal and greatly simplified.*

For a simple illustration, consider an adaptive cruise controller tasked to set a car's acceleration and braking in a way that optimizes fuel consumption. Of course, even though it is designed for fuel consumption, the controller must also maintain a safe following distance behind other vehicles. Verifying safety for this controller means proving that this safe following distance is indeed maintained. But complicated optimizations designed for fuel efficiency can obfuscate the part of the code that guarantees safety. It would be easier to instead verify a simpler model of cruise control that is designed purely to satisfy the safety requirement. If this simpler model also provably contains all possible behaviors of the original fuel-efficient controller, then the fuel-efficient controller inherits the safety proof. However, in the formal syntax of $\mathsf{d}\mathcal{L}$, the possible behaviors of the two hybrid programs can not be directly compared without significant proving effort, often equal to proving the original system directly.

These and other observed challenges usually fit into one or more of the following categories:

- **Breaking the system into parts.** Some systems naturally decompose into smaller subsystems. We are able to leverage this, for example, when verifying safety for a highway of cars all operating under distributed controllers in [5]. A two-car system makes the basic

building block for a lane of cars, then a lane of cars is the building block for the highway, as illustrated in Fig. 1.1. In this case, we lucked out that the system could in fact be decomposed in this way. On top of that, we were very careful to write the model in such a way that this decomposition would occur early in the proof, since the hybrid programs would need to be unrolled by proof rules until the point where they differ. This decomposition technique can only be effective in cases where the subsystem is nearly identical to the original hybrid program. And, even in cases where decomposition is possible, figuring out the how to model the system to take advantage of this decomposition requires a considerable amount of foresight.

- **Abstracting implementation-specific designs.** When a hybrid program describes a specific system implementation which is not directly related to the safety property (e.g. a PID controller, or a fuel-efficient adaptive cruise controller), it is significantly harder to verify than an abstraction of the system which is only designed for the safety-critical aspects of the system. An over-approximation of the system may be easier to prove, but without a refinement relation in $\mathsf{d}\mathcal{L}$, relating the abstract model to the original model requires either a significant proving effort or must be reasoned about informally.
- **Leveraging iterative system design.** Due to the many subtleties of programming discrete events to control continuous physical systems, it is almost impossible to design a safe cyber-physical system on the first attempt. Even small cyber-physical systems are too hard to get right without careful study. To aid in system design, we almost always require an iterative approach [5–9]. We verify a simplified version of the system first in order to learn about it's behavior, and then extend it to be closer to the original goal. With each iteration, we can localize where new bugs can occur and we can reapply the proof tricks we learned in the simplified system. Currently, using iterative system design requires restarting proofs from scratch with every increase in complexity. While iterative design is extremely useful and sometimes crucial for getting complicated systems right, it is also very costly.
- **Maintaining a modular proof structure.** As proofs and systems become larger and more complex, there is a growing need for a formal way to inject more modularity into the proof structure. For example, the key to proving safe separation for an arbitrary number

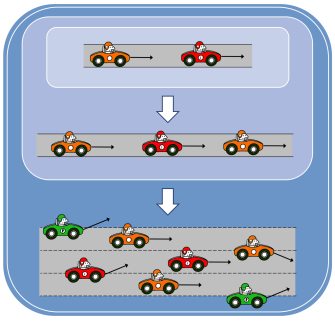

Figure 1.1: Breaking a large system into smaller building blocks was the key insight for verifying distributed car control in [5].

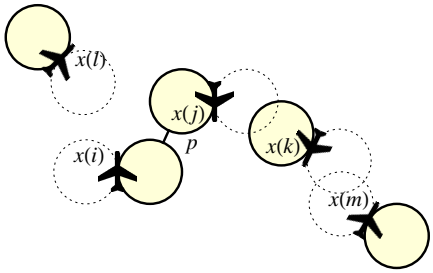

Figure 1.2: Maintaining a modular proof structure was the key insight for verifying this distributed aircraft protocol in [6].

of aircraft in [6] is to first prove that each plane stays on the edge of its own disc and then prove that the discs stay safely separated, as illustrated in Fig. 1.2.

The insight in the proposed thesis is that all of these seemingly different problems, from controlling the structure of large proofs to the high cost of iteratively designing safe systems, can be addressed with the same technique: providing direct proof support for relating two hybrid programs. By adding this technique to our toolbox, we can systemize on a technical level what we have done before in an ad hoc way.

1.1.1 Proposed Work

Previously completed verification work in the fields of car control and aircraft control have provided a solid understanding of several major challenges in CPS verification (see Chapter 3). In that work, many of the challenges were addressed for the first time in theorem proving for hybrid systems, so our approaches were understandably tailored to each specific problem. Now that we are looking at our solutions retrospectively, there is a clear missing piece: how can we efficiently compare the behavior of a panoply of hybrid programs without losing the strong verification guarantees that come with a proof? Table 1.1 lists the challenges we faced in each case study that can be addressed by dR$\mathcal{L}$.

In the proposed thesis we will introduce *differential refinement logic* (dR$\mathcal{L}$), an extension of d$\mathcal{L}$ which will allow for the direct comparison of hybrid programs. We accomplish this by adding a *refinement* relation to the grammar of d$\mathcal{L}$. We say that α refines β (written $\alpha \leq \beta$) if and only if all possible transitions in hybrid program α are also possible in β. Whenever a refinement relation has been proved between two hybrid programs, the more restrictive program automatically inherits all safety properties proved about the program that is more permissive. The thesis will introduce a syntax, semantics, and proof calculus for dR$\mathcal{L}$ (Chapter 4).

We will test the usefulness of dR$\mathcal{L}$ by revisiting the case studies verified using d$\mathcal{L}$ and exploring where dR$\mathcal{L}$ simplifies and strengthens these proofs. They include: verifying the distributed car control system (Section 3.1), verifying distributed aircraft protocols (Section 3.2), verifying safety envelopes and PID controllers for a single car controller (Section 3.3), and exploring efficiency analysis within these safety envelopes (Section 3.4). We then hope to provide a decision procedure for the *-free fragment of the logic (Section 4.3).

1.1.2 Future Work

In addition to addressing many challenges we have already encountered in theorem proving for hybrid systems, dR$\mathcal{L}$ has the potential to make an impact in areas from proof search heuristics to code generation and synthesis. We have observed that it is often significantly easier to verify models with implicit controllers. In fact, if an implicit controller is taken to an extreme where the requirements of the controller include a nested model of the physical behavior of the system, verification becomes trivial. We also hope to explore new proof search heuristics that leverage the refinement relation in a way that automatically enforces hierarchical proof structures, possibly resulting in more efficient proof search. Additionally, we believe that refinement could reduce some of the challenges faced by synthesis, as each step of the synthesis process could be checked

that it is truly a refinement of the system that is verified to adhere to safety requirements. If time allows, we hope to include research in these directions in the thesis.

Challenges addressed by dRℒ				
	break systems into parts	implementation-specific design	iterative system design	proof modularity
Distributed Car Control (Section 3.1)	✓	—	✓	✓
Smart Intersections (Section 3.1	—	—	✓	✓
Distributed Aircraft (Section 3.2)	—	—	✓	✓
PID Controller (Section 3.3)	—	✓	✓	—
Safety Envelopes (Section 3.3)	✓	✓	✓	—
Efficiency Analysis (Section 3.4)	—	—	✓	—

Table 1.1: This table lists specific challenges encountered in previously completed case studies. dRℒ can address all of them with a single, unified approach.

1.2 Thesis Statement

The goal of this thesis is to provide a formal framework for comparing hybrid programs so that CPS verification can benefit from arguments reasoning explicitly about the relations of multiple hybrid systems. In support of this, we introduce differential refinement logic (dRℒ), develop a proof calculus for it, and demonstrate its usefulness in a variety of CPS domains.

1.3 Timeline

The following is a summary of the intended dates of completion of the proposed work. **I plan to be done with the proposed thesis before August 2015.**

1. **Dec 2014:** Thesis proposal.

2. **February 2015:** Complete proposed work on dRℒ (Chapter 4), conference or journal submission.
3. **Spring 2015:** Complete additional work on applications of dRℒ. Starting by examining the benefits of dRℒ for previously completed and well-understood case studies from Chapter 3. Then followed by an exploration of MPC-style proving (Section 5.1) and an investigation of hierarchical proof structures using dRℒ (Section 5.2). Exploring program synthesis and implementation (Section 5.3) is a stretch goal, also to be completed by the end of May 2015 if time permits.
4. **Summer 2015:** Thesis writing.
5. **July/August 2015:** Thesis defense.

Chapter 2

Related Work

2.1 Hybrid Systems

A comprehensive discussion of the current landscape of verification for cyber-physical systems can be found in [10, 11]. In this section, we will focus on related work in verification techniques specific to car control and aircraft control, as they account for many of the examples we study.

2.2 Refinement Relations

The proposed research extends differential dynamic logic $\mathsf{d}\mathcal{L}$, originally proposed by Platzer [3, 4]. $\mathsf{d}\mathcal{L}$ has had many successes in proving safety for challenging CPS. This work seeks to improve upon those successes by allowing existing proofs (see Chapter 3) to be streamlined, extended, or made fully formal. It is also our hope that $\mathsf{dR}\mathcal{L}$ will enable users to prove properties that were previously out of scope for $\mathsf{d}\mathcal{L}$ due to user interactions being too challenging or numerous.

In [12], Mitsch et al. present proof-aware refactorings for hybrid programs in $\mathsf{d}\mathcal{L}$. By examining a number of common refactoring transformations on hybrid programs, they can make corresponding changes in their correctness proofs, thereby increasing proof reuse between verifications of similar hybrid programs. They introduce the semantic concept of a refinement relation over hybrid programs, which can then be leveraged in meta-level analysis to show that program transformations preserve safety or liveness properties. The analysis is limited to finitely many use cases. We share many goals with this research, and commend it as the first step toward defining refinement based on reachable states and increasing the flexibility of which systems can be considered refinements of each other. In this thesis, we hope to make complementary progress toward these goals by introducing a refinement relation formally into the grammar of $\mathsf{dR}\mathcal{L}$. With an associated proof calculus, we hope to leverage syntactic automation to reduce the level of proof awareness required of the user. We also aim to decrease the user's reliance on meta analysis, which is more susceptible to human error. Additionally, we examine refinement within a given context, so $\alpha \leq \beta$ may hold under the current context while not holding generally. Mitsch et al. achieve this by introducing a formula F which can be used to describe the context in which the refinement holds, but by including the refinement directly in the logic, context is

automatically defined and updated along various proof branches.

While the refinement relation as a first-class member of dRℒ is new, the concept of refinement for cyber-physical systems has been in use for quite some time. Model checking, which often suffers from statespace explosion, has seen tremendous success in using abstraction to keep the statespace small, then iteratively refining the model to exclude spurious counter examples (CEGAR [13]). Some verification tools (Rodin [14], CHARON [15]) have had success defining refinement based on trace inclusion [16], which is more restrictive than our approach of comparing the reachable states.

Because hybrid programs in dRℒ form a Kleene algebra with tests (KAT) [17], we draw heavily on research done on these algebras when designing the corresponding proof calculus for dRℒ. For example, the proof rules presented in Fig. 4.2 are derived directly from KAT axioms. But this is not the end of the story for dRℒ. Additional rules are required to handle the many complexities that arise from differential equations. Even simple hybrid programs that would seem trivially unrelated when examined through the lens of KAT, may in fact satisfy the refinement relation. For example, consider the hybrid programs $x := 10$ and $x := 1; x' = x$. The program $x := 10$ simply assigns the value 10 to variable x. The program $x := 1; x' = x$ first sets x to 1 and then follows the solution to the differential equation $x' = x$ for a nondeterministic amount of time. While these hybrid programs appear unrelated, the first hybrid program is actually a refinement of the second. The continuous evolution that follows $x' = x$ evolves for a nondeterministic period of time, thus allowing x to take any value greater than 1, which means it includes a transition where x takes the value 10.

Even with these differences, we expect many results from research related to KAT to be relevant to dRℒ. For example, as we research possible decision procedures for dRℒ we will further examine work on bisimulation-based decision procedures for NetKAT, a logic for reasoning about packet switching networks [18, 19]. There still remain many open questions, particularly on the proper handling of differential equations.

2.3 Car Control

Because of its societal relevance, numerous aspects of car control have been previously studied [20–37]. Major initiatives have been devoted to developing next generation individual ground transportation solutions, including the California PATH project, the SAFESPOT and PReVENT initiatives, the CICAS-V system, and many others. The societal relevance of vehicle cooperation for CICAS intersection collision avoidance [30] and for automated highway systems [24, 27] has been emphasized. Horowitz et al. [29] proposed a lane change maneuver within platoons. Varaiya [32] outlines the key features of an IVHS (Intelligent Vehicle/Highway System). A significant amount of work has been done in the pioneering California PATH Project.

Our interest in car control comes first from the challenges that arise when a large system of cars is modeled as a distributed system, with safety properties (e.g. ensuring there are no collisions) that must be guaranteed globally. The distributed nature of car control problems lend them to being broken down into subsystems for the purpose of verification. Secondly, many car control systems are implementation-specific: they should never collide with other cars, but that is not the *only* thing they are designed to do.

Dao et al. [22, 23] developed an algorithm and model for lane assignment. Their simulations suggest [22] that traffic safety can be enhanced if vehicles are organized into platoons, as opposed to having random space between them. Our approach considers an even more general setting: we not only verify safety for platoon systems, but also when cars are driving on a lane without following platooning controllers. Hall et al. [25] also used simulations to find out what is the best strategy of maximizing traffic throughput. Chee et al. [34] showed that lane change maneuvers can be achieved in automated highway systems using the signals available from on-board sensors. Jula et al. [28] used simulations to study the conditions under which accidents can be avoided during lane changes and merges. They have only tested safety partially. In contrast to [22, 23, 25, 28, 34], we do not use simulation but formal verification to validate our hypotheses.

Hsu et al. [26] propose a control system for IVHS that organizes traffic in platoons of closely spaced vehicles. They specify this system by interacting finite state machines. Those cannot represent the actual continuous movement of the cars. We use differential equations to model the continuous dynamics of the vehicles and thus consider more realistic models of the interactions between vehicles, their control, and their movement.

Stursberg et al. [31] applied counterexample-guided verification to a cruise control system with two cars on one lane. Their technique can not scale to an arbitrary number of cars. Althoff et al. [36] use reachability analysis to prove the safety of evasive maneuvers with constant velocity. They verify a very specific situation: a wrong way driver threatens two autonomously driving vehicles on a road with three lanes.

There are several challenges that still need to be solved to make next generation car control a reality. The most interesting challenge for us is that it only makes sense to introduce any of these systems after its correct functioning and reliability has been ensured. Otherwise, the system might do more harm than good. This is the formal verification problem for distributed car control. What makes this problem particularly challenging is its complicated dynamics. Distributed car control follows a hybrid dynamics, because cars move continuously along differential equations and their behavior is affected by discrete control decisions like when and how strongly to brake or to accelerate and to steer.

In [38] we develop a distributed car control system and a formal proof that this system is collision-free for arbitrarily many cars, even when new cars enter or leave a multi-lane highway with arbitrarily many lanes. This was done by explicitly developing the model to take advantage of the compositional aspects of the system (see Fig. 1.1). We reduce the proof to subsystems that can be verified without awareness of details about the system as a whole. We believe the principles behind this modular structure and verification techniques are useful for other systems and that they can be formalized by using the refinement operator in dR$\mathcal{L}$.

2.4 Aircraft Control

Verification of air traffic control is particularly challenging because it lies at the intersection of many fields which already give tough verification problems when examined independently. It is a distributed system, with a large number of aircraft interacting over an unbounded time horizon. Each aircraft has nonlinear continuous dynamics combined with complex discrete controllers. And finally, every protocol must be flyable (i.e. not cause the aircraft to enter a stall, bank too

sharply, or require it to turn on sharp corners). The complexity of curved flight dynamics has been difficult for many analysis techniques [39–46], which often resort to unflyable approximations of flight trajectories that require aircraft to turn on corners.

These strong guarantees are especially important in a distributed system with a large number of interacting participants. As in [39, 42, 45, 47, 48], many previous approaches to aircraft control have looked into a relatively small number of agents. But with thousands of aircraft flying through airspace daily, this system is already far too complex for humans to predict every scenario by looking at interactions of only a few aircraft.

Verification methods for systems with an arbitrary number of agents behaving under distributed control fall primarily into one of two categories: theorem proving and parameterized verification. Johnson and Mitra [46] use parameterized verification to guarantee that a distributed air traffic landing protocol (SATS) is collision free. Using backward reachability, they prove safety in the SATS landing maneuver given a bound on the number of aircraft that can be engaged in the landing maneuver. The protocol divides the airspace into regions and models the aircraft flight trajectory within each region by a clock.

Other provably safe systems with a specific (usually small) number of agents are presented in [39, 42, 45, 47]. The work by Umeno and Lynch [42, 45] considers real-time properties of airport protocols using Timed I/O Automata. Duperret et al. [47] verify a roundabout maneuver with three vehicles. Each vehicle is constrained to a specific, pre-defined path, so physical dynamics are simplified to one dimension. Tomlin et al. [39] analyze competitive aircraft maneuvers game theoretically using numerical approximations of partial differential equations. As a solution, they propose roundabout maneuvers and give bounded-time verification results for up to four aircraft using straight-line approximations of flight dynamics.

Flyability is identified as a major challenge in Košecká et al. [49], where planning based on superposition of potential fields is used to resolve air traffic conflicts. This planning does not guarantee flyability but, rather, defaults to classical vertical altitude changes whenever a nonflyable path is detected. The resulting maneuver has not yet been verified. The planning approach has been pursued by Bicchi and Pallottino [50] with numerical simulations.

Numerical simulation algorithms approximating discrete-time Markov Chain approximations of aircraft behavior have been proposed by Hu et al. [40]. They approximate bounded-time probabilistic reachable sets for one initial state. We consider hybrid systems combining discrete control choices and continuous dynamics instead of uncontrolled, probabilistic continuous dynamics. Hwang et al. [44] have presented a straight-line aircraft conflict avoidance maneuver involving optimization over complicated trigonometric computations, and validate it using random numerical simulation and informal arguments. The work of Dowek et al. [41] and Galdino et al. [43] provides formal geometrical proofs in PVS, but considers straight-line maneuvers, which are unflyable.

Pallottino et al. [51] proposed a distributed collision avoidance policy that provides a thorough empirical description of the system's behavior, emphasizing simulation and physical experiment. They formulate a liveness property and give probabilistic evidence for it using Monte Carlo methods. They also provide an informal proof of safety that is similar in high-level ideas to the proofs in [6], but does not consider a model for flight dynamics.

The collision avoidance maneuvers we present in [6] use curved flight, which is *flyable*, but much more difficult to analyze. We produce *formal*, deductive proofs to verify uncountably many

initial states and give unbounded-time horizon verification results. We use symbolic computation so that numerical and floating point errors can not violate soundness. We analyze hybrid system dynamics directly, rather than approximations like clocks. We verify the case of *arbitrarily many* aircraft, which is crucial for dense airspace.

In Section 3.2, we propose that the proofs presented in [6] may be simplified using differential refinement logic and we propose them as good case studies to be included in the thesis.

Chapter 3

Completed Work

3.1 Distributed Car Control

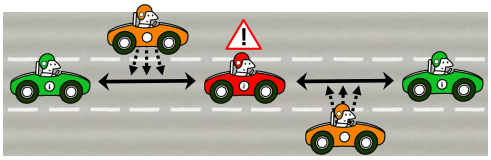

Figure 3.1: Multilane highway verified for an arbitrary number of cars. Cars may change lanes, and enter and exit the highway.

Car safety measures can be most effective when the cars on a street coordinate their control actions using distributed cooperative control. While each car optimizes its navigation planning locally to ensure the driver reaches his destination, all cars coordinate their actions in a distributed way in order to minimize the risk of safety hazards and collisions. These systems control the physical aspects of car movement using cyber technologies like local and remote sensor data and distributed V2V and V2I communication. They are thus cyber-physical systems.

In [38], we considered a distributed car control system inspired by the ambitions of the California PATH project, the CICAS system, SAFESPOT and PReVENT initiatives. We developed a formal model of a distributed car control system in which every car is controlled by adaptive cruise control. One of the major technical difficulties is that faithful models of distributed car control have both distributed systems and hybrid systems dynamics. They form distributed hybrid systems, which makes them very challenging for verification. In a formal proof system, we verify that the control model satisfies its main safety objective and guarantees collision freedom for arbitrarily many cars driving on a street, even if new cars enter the lane from on-ramps or multi-lane streets. The proof of safety for this system relies heavily on a hierarchical proof structure, with lemmas proving safety first for two cars on a single lane, then for an arbitrary number of cars on a single lane, and finally for an arbitrary number of cars on an arbitrary number of lanes.

Our main contribution in [38] is that we develop a distributed car control system and a formal proof that this system is collision-free for arbitrarily many cars, even when new cars enter or leave a multi-lane highway with arbitrarily many lanes. Another contribution is that we develop a proof structure that is strictly modular. We reduce the proof to modular stages that can be verified without the details in lower levels of abstraction. We believe the principles behind this

modular proof structure can be formalized using dR$\mathcal{L}$ and that such verification techniques are useful for other systems beyond the automotive domain. Further contributions are:

- This is the first case study in distributed hybrid systems to be verified with a generic and systematic verification approach that is not specific to the particular problem.
- We identify a simple invariant that all cars have to obey and show that it is sufficient for safety, even for emergent behavior of multiple distributed car maneuvers.
- We identify generic and static constraints on the input output parameters that any controller must obey to ensure that cars always stay safe.
- We demonstrate the feasibility of distributed hybrid systems verification.

In [7] we present a model for the interaction of two cars and a traffic light at a two lane intersection and verify with a formal proof that our system always ensures collision freedom and that our controller always prevents cars from running red lights. Naïvely, this may appear to be a simple problem where all we have to do is to ensure that at most one of the traffic lights at an intersection is red. But this does not work, because the overall system would still be unsafe if the car controllers disobey the red lights or if the traffic lights switch in a way that the car controllers have no way of complying with. We develop a controller for a stoplight intersection, identify the crucial safety constraints, and formally verify that collision freedom is guaranteed. Ideally, this model could be used as a building block to piece together increasingly complicated intersections with multiple lanes (proposed as future work in [7]).

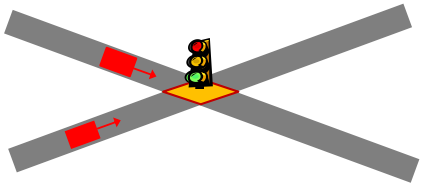

Figure 3.2: Verified traffic light for two lane intersection.

Both of these case studies benefit from *breaking the system into parts* (both within the formal proof structure as in [38] and as meta analysis in [7]). We will examine how using dR$\mathcal{L}$ might simplify and strengthened these proofs.

3.2 Distributed Aircraft Control

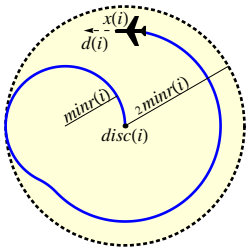

Figure 3.3: Verified *big disc* collision avoidance maneuver.

As airspace becomes ever more crowded, air traffic management must reduce both space and time between aircraft to increase throughput, making onboard collision avoidance systems ever more important. These safety-critical systems must be extremely reliable, and as such, many resources are invested into ensuring that the protocols they implement are accurate. Still, it is challenging to guarantee that such a controller works properly under every circumstance. In tough scenarios where a large number of aircraft must execute a collision avoidance maneuver, a human pilot under stress is not necessarily able to un-

derstand the complexity of the distributed system and may not take the right course, especially if actions must be taken quickly.

In this paper we specify and verify two control policies for planar aircraft avoidance maneuvers using automated theorem prover KeYmaeraD to produce a proof of safety for each of them. We design these policies such that all aircraft adhere to a simple and easy-to-implement separation principle: associated with each aircraft is a disc, within which the aircraft must remain. By *breaking the system into parts* in this way, the problem reduces to proving that i) sufficient separation is maintained between pairs of discs, and ii) individual aircraft always remain inside their associated disc. We model 2D flight dynamics since they are the relevant dynamics for planar maneuvers, but investigating 3D maneuvers and dynamics may make interesting future work.

The complexities which arise from the curved flight trajectories of an arbitrary number of aircraft interacting in a distributed manner, along with the tight coupling of discrete control and continuous dynamics presently make KeYmaeraD the only verification tool capable of proving safety for this system. Our contributions are:

- We provide the first formally verified distributed system of aircraft with *curved flight* dynamics.
- Our controller requires only flyable aircraft trajectories with no corners or instantaneous changes of ground speed.
- We prove our controller is safe for an arbitrarily large number of aircraft. This guarantee is necessary for high-traffic applications such as crowded commercial airspace, unmanned aerial vehicle maneuvers, and robotic swarms.
- Other aircraft may enter an avoidance maneuver already in progress and safety for all aircraft is guaranteed still.
- We prove that even when the interactions of many aircraft cause unexpected emergent behaviors, all resulting control choices are still safe.
- We present hierarchical and compositional techniques to reduce a very complex system into smaller, provable pieces.

In the design and verification process for this system, it was clear that *leveraging iterative system design* had the potential to substantially reduce the challenge of proving safety for the entire system, as we first proved the controller safe for two aircraft before moving to the distributed space. In this thesis, we will investigate whether using dR$\mathcal{L}$ could accomplish this.

3.3 Verifying Safety Envelopes, Implementing PID Controllers

Theorem provers are often most efficient when using generic models that abstract away many of the controller details. These abstract models use very general conditions that can then be used as *safety envelopes* when designing more detailed controllers.

In [9] we proposed an approach in which the verified safety envelopes were used as static conditions to be checked for specific controller designs, without requiring the inclusion of any physical dynamics of the system. We demonstrated this approach using the KeYmaera theorem prover for d$\mathcal{L}$ for two examples: adaptive cruise control with two cars and a cooperative intersec-

tion collision avoidance system (CICAS) for left-turn assist. In each case, a proof of safety for the closed-loop system provided static safety envelopes that are then used informally to check the controller design.

In [8, 9], we used the concept of refinement to reason informally outside of dℒ that a deterministic PID implementation of a non-deterministic hybrid program can inherit the original safety proofs. Revisiting these projects, this time with dRℒ, could allow us to *abstract implementation-specific designs*, like a PID controller, and provide a verification of the system which does not rely on meta analysis.

3.4 Symbolic Verification Allows Efficiency Analysis

In [52] we considered an adaptive cruise control system in which control decisions are made based on position and velocity information received from other vehicles via V2V wireless communication. If the vehicles follow each other at a close distance, they have better wireless reception but collisions may occur when a follower car does not receive notice about the decelerations of the leader car fast enough to react before it is too late. If the vehicles are farther apart, they would have a bigger safety margin, but the wireless communication drops out more often, so that the follower car no longer receives what the leader car is doing. In order to guarantee safety, such a system must return control to the driver if it does not receive an update from a nearby vehicle within some timeout period. The value of this timeout parameter encodes a tradeoff between the likelihood that an update is received and the maximum safe acceleration. Combining formal verification techniques for hybrid systems with a wireless communication model, we analyzed how the expected efficiency of a provably-safe adaptive cruise control system is affected by the value of this timeout.

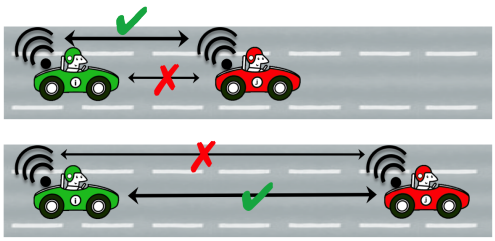

Figure 3.4: There is a tradeoff between signal strength and keeping a safe following distance with a higher tolerance for dropped packets.

Because this controller needed to be optimally permissive in order for the efficiency analysis to work out, the process of designing the controller required many iterative steps in order to get it to its final state. We will investigate whether using dRℒ could use the refinement relation to simplify this *iterative system design* process.

Chapter 4

Differential Refinement Logic

In this chapter we will discuss how we propose to develop the differential refinement logic that will be presented in the thesis. In addition to defining the syntax (Section 4.1) and semantics (Section 4.2) of dRL, we present an initial (and incomplete) set of sound proof rules for the logic (Section 4.4). We give a brief proposal for developing a decision procedure for the *-free fragment of dRL in Section 4.3. This chapter should be taken as a preliminary outline for a bulk of the work that will be completed in the thesis.

4.1 Syntax

In this section, we extend differential dynamic logic (dL), a specification and verification language for hybrid systems, and therefore we make heavy use of constructs introduced in [3]. Both dL and dRL model cyber-physical systems as *hybrid programs* (HPs). HPs combine differential equations with traditional program constructs and discrete assignments. As has been previously noted [4], HPs form a Kleene algebra with tests [17].

Definition 1 (Hybrid program)**.** HPs are defined by the following grammar (where α, β are HPs, x a variable, θ a term possibly containing x, and H a formula of dRL):

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

The effect of *assignment* $x := \theta$ is an instantaneous discrete jump assigning θ to x. The effect of *differential equation* $x' = \theta$ is a continuous evolution where the differential equation $x' = \theta$ holds *and* (written & for clarity) formula H holds throughout the evolution (the state remains in the region described by H).

The effect of *test* $?H$ is a *skip* (i.e., no change) if formula H is true in the current state and *abort* (blocking the system run by a failed assertion), otherwise. *Non-deterministic choice* $\alpha \cup \beta$ is for alternatives in the behavior of the distributed hybrid system. In the *sequential composition* $\alpha;\beta$, HP β starts after α finishes (β never starts if α continues indefinitely). *Non-deterministic repetition* α^* repeats α an arbitrary number of times, including zero times.

Except for the changes to formulas (addressed in Definition 2), the grammar of hybrid programs is unchanged from that used by dL.

Definition 2 (dRL formula)**.** Formulas in dRL are defined by the following grammar (where ϕ, ψ are dRL formulas, x is a variable, θ_1, θ_2 are terms, and α, β are HPs):

$$\phi, \psi \ ::= \ \theta_1 \leq \theta_2 \mid \neg\phi \mid \phi \wedge \psi \mid \forall x\, \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi \mid \boldsymbol{\beta \leq \alpha}$$

In addition to all formulas of first-order real arithmetic, dRL allows formulas of the form $\beta \leq \alpha$ with HPs α and β. Formala $\beta \leq \alpha$ is true in a state ν iff all states reachable from ν by following the transitions of β could also be reached from state ν by following transitions of α (the transition semantics of HPs are formally defined in Definition 3). Less formally, the behaviors of α subsume those of β, or we say that β *refines* α.

Just as in dL, we may write formula $[\alpha]\phi$ with an HP α and a formula ϕ in dRL. Formula $[\alpha]\phi$ is true in a state ν iff formula ϕ is true in all states that are reachable from ν by following the transitions of α.

The formulas $\beta \leq \alpha$ and $[\alpha]\phi$ make a powerful pair; when both are true, then we know that formula ϕ is true in all states reachable from ν by following the transitions of β, i.e. $[\beta]\phi$.

We use $[\alpha]\phi, \beta \leq \alpha$, and other formulas in dRL for stating and proving properties of HPs.

4.2 Semantics

The semantics of HPs are defined as a reachability relation. A *state* ν is a mapping from a set V of logical and state variables to $\mathbb{R}$. The set of states is denoted $\mathcal{S}$. The value of term θ in state ν is denoted by $[\![\theta]\!]_\nu$. The transition semantics of HPs remain the same as in dL [3, 4].

Definition 3 (Transition semantics of HPs)**.** Each HP α is interpreted semantically as a binary reachability relation $\rho(\alpha) \subseteq \mathcal{S}_\alpha \times \mathcal{S}_\alpha$ over states, defined inductively and as usual in differential dynamic logic (dL):

- $\rho(x := \theta) = \{(\nu, \omega) \ : \ \omega = \nu$ except that $[\![x]\!]_\omega = [\![\theta]\!]_\nu\}$
- $\rho(?H) = \{(\nu, \nu) \ : \ \nu \models H\}$
- $\rho(x' = \theta \,\&\, H) = \{(\varphi(0), \varphi(r)) \ : \ \varphi(t) \models x' = \theta$ and $\varphi(t) \models H$ for all $0 \leq t \leq r$ for a solution $\varphi : [0, r] \to \mathcal{S}$ of any duration $r\}$; i.e., with $\varphi(t)(x') \overset{\text{def}}{=} \frac{d\varphi(\zeta)(x)}{d\zeta}(t)$, φ solves the differential equation and satisfies H at all times.
- $\rho(\alpha \cup \beta) = \rho(\alpha) \cup \rho(\beta)$
- $\rho(\alpha;\beta) = \rho(\beta) \circ \rho(\alpha) = \{(\nu, \omega) : (\nu, \mu) \in \rho(\alpha), (\mu, \omega) \in \rho(\beta)\}$
- $\rho(\alpha^*) = \bigcup_{n \in \mathbb{N}} \rho(\alpha^n)$ with $\alpha^{n+1} \equiv \alpha^n; \alpha$ and $\alpha^0 \equiv ?\mathit{true}$

Differential dynamic logic combines first-order real arithmetic and dynamic logic generalized for hybrid programs [4]. Formulas in first-order arithmetic can be used to precisely specify complicated safety or control regions. In dL, a box modality can be used to express that a property ϕ holds in *all* states reachable by a transition of hybrid program α; the box modality is written $[\alpha]\phi$. Similarly, the diamond modality requires that ϕ hold in at least one state reachable by a transition of α; the diamond modality is written $\langle\alpha\rangle\phi$.

Differential refinement logic dRL introduces the refinement relation for hybrid programs to the semantics. This addition is indicated in bold in Definition 4. The formula $\alpha \leq \beta$ is true in state ν iff the set of all states reachable from ν by following the transitions of HP α is a subset of the states reachable from ν by following the transitions of HP β.

We also use $\alpha = \beta$ to denote equivalence of hybrid programs α and β, but this is defined syntactically as $\alpha \leq \beta \wedge \beta \leq \alpha$.

Definition 4 (dRL semantics)**.** The *satisfaction relation* $\nu \models \phi$ for a dRL formula ϕ in state ν is defined inductively and, aside from the refinement and equivalence relations, it is defined as usual in first-order modal logic for real arithmetic.

- $\nu \models (\theta_1 \leq \theta_2)$ iff $[\![\theta_1]\!]_\nu \leq [\![\theta_2]\!]_\nu$.
- $\nu \models \neg F$ iff $\nu \not\models F$, i.e. if it is not the case that $\nu \models F$.
- $\nu \models F \wedge G$ iff $\nu \models F$ and $\nu \models G$.
- $\nu \models \forall x\, F$ iff $\nu^d_x \models F$ for all $d \in \mathbb{R}$.
- $\nu \models [\alpha]\phi$ iff $\omega \models \phi$ for all ω with $(\nu, \omega) \in \rho(\alpha)$
- $\nu \models \langle\alpha\rangle\phi$ iff $\omega \models \phi$ for some ω with $(\nu, \omega) \in \rho(\alpha)$
- $\boldsymbol{\nu \models \alpha \leq \beta}$ iff $\boldsymbol{\{\omega : (\nu, \omega) \in \rho(\alpha)\} \subseteq \{\mu : (\nu, \mu) \in \rho(\beta)\}}$

If $\nu \models \phi$, we say that ϕ holds at state ν. A formula ϕ is *valid* iff ϕ holds at all states; we write $\models \phi$ to denote that ϕ is valid.

Notice that the state space is the collection of all mappings of variables to $\mathbb{R}$. Even when there is a variable x in hybrid program β which does not appear in hybrid program α, the variable x is still in the mapping of the state space of ν and the transition $\rho(\alpha)$ will require that the value of x is unchanged.

4.3 Decision procedure for *-free fragment

We can show that the refinement relation can actually be written equivalently in dL, so we could easily lift the axioms from dL and add the equivalence transformation to accomplish completeness for dRL:

$$\models_{\mathsf{dRL}} \alpha \leq \beta \iff \models_{\mathsf{dL}} \forall \bar{x}\, (\langle\alpha\rangle(x = \bar{x}) \rightarrow \langle\beta\rangle(x = \bar{x})), \tag{4.1}$$

where x is a vector of all variables in α and β, and $\bar{x}$ is a vector of fresh variables of equal length to x. as it is e completely the opposite of what we are trying to accomplish with dRL– taking advantage of the structure of a hybrid program to prove refinement relations. In practice, the formula on the right-hand side of (4.1) will be significantly more complicated (due to the added diamond modalities and quantifying over variables) than verifying properties about α and β directly, making this is a terrible idea.

Similarly, it is possible to encode the box modality of dL purely as a refinement relation:

$$\models_{\mathsf{dL}} [\alpha]\phi \iff \models_{\mathsf{dRL}} \alpha \leq x := *; ?\phi, \tag{4.2}$$

where x again is a vector of all variables in α, and $x := *$ is a nondeterministic assignment, so the HP $x := *; ?\phi$ transitions to all states where ϕ is satisfied.

These encodings illustrate that this thesis will not set out to prove that dRL has more expressivity than dL. Neither is our goal to show that the proof calculus of dRL improves the completeness results already established for dL ([1] proves dL is complete with respect to differential equations). However, in practice, we observe utility in combining these modes of reasoning.

The background and familiarity with dℒ that we have gained through previously completed work (discussed in Chapter 3 have shown a serious need for proper refinement.

We develop a proof calculus for dRℒ in order to directly leverage structural similarities between HPs with comparable behaviors. The improvements that dRℒ brings to dℒ will depend on the quality of refinement-specific proof rules. To evaluate dRℒ's usefulness, we will show that it aids verification of the examples with which we are most familiar (see Chapter 3). Additionally, we will endeavor to develop a decision procedure for at least the *-free fragment of dRℒ to give some theoretical support to this claim.

4.4 Proof Calculus

A proof calculus associated with a logical language such as dRℒ is a set of syntactic transformations that are each proved sound. By combining many of these transformations on a complicated formula, we may simplify and break apart the formula until we are left with formulas that we know are true, in which case we have a proof of our original complicated formula. Because this process is entirely syntactic, such a proof can be automatically checked by a computer or, more importantly, automatically generated (this process is called a proof search). In this section we present some preliminary rules for the dRℒ proof calculus. This collection of rules is nowhere near complete, but rather should be considered a starting point for the proof calculus for dRℒ.

The rules in Fig. 4.1 capture the fact that refinement relation over hybrid programs is a partial order. The rules in Fig. 4.2 are derived directly from KAT axioms [17]. For hybrid programs, nondeterministic choice $\cup$ is the additive operator, and sequential composition ; is the multiplicative operator.

The hybrid program $?\bot$, where $\bot$ is false, is a test that always fails. The transition semantics for this hybrid program is the empty set. It is therefore used as the additive identity (see rule $\cup_{id}$ in Fig. 4.2) and the multiplicative annihilator (see rules $;_{annih-r}$ and $;_{annih-l}$ in Fig. 4.2). The hybrid program $?\top$, where $\top$ is true, is a test that always succeeds. We use $?\top$ as the multiplicative identity (see rules $;_{id-r}$ and $;_{id-l}$ in Fig. 4.2), as it does not change the transition semantics of any hybrid program when sequentially composed.

Two proof rules that provide strong motivation for developing dRℒ can be seen in Fig. 4.3. Recall that a box modality $[\alpha]\phi$ holds only if ϕ holds in every state reachable by following transitions on α. So, the refinement rule for box modalities $[\leq]$ says that if formula ϕ holds in every state reachable on β, and α is a refinement of β, then ϕ must also hold in every state reachable on α. A diamond modality $\langle\alpha\rangle\phi$ holds if there is at least one transition on α to a state where ϕ is true. We have a similar rule for diamond modalities $\langle\leq\rangle$, which says that if α is a refinement of β, and ϕ holds in at least one transition on α, then that same state must also be reachable on β and therefore $\langle\beta\rangle\phi$ must bet true.

Proof rules presented here represent those for which we have already proved soundness. A good deal of work remains to identify which additional proof rules can be soundly added to the calculus to make it useful. For example, differential equations and differential algebraic constraints will need to be carefully handled, as will nondeterministic repetition.

$$\frac{}{\vdash \alpha \leq \alpha}(\leq_{refl}) \qquad \frac{}{\alpha \leq \beta \wedge \beta \leq \gamma \vdash \alpha \leq \gamma}(\leq_{trans}) \qquad \frac{\alpha \leq \beta \wedge \beta \leq \alpha}{\alpha = \beta}(\leq_{antisym})^{1}$$

Figure 4.1: dRL rules - refinement is a partial order

$$\frac{}{\vdash \alpha \cup (\beta \cup \gamma) = (\alpha \cup \beta) \cup \gamma}(\cup_{assoc}) \qquad \frac{}{\vdash \alpha \cup \beta = \beta \cup \alpha}(\cup_{comm})$$

$$\frac{}{\vdash \alpha \cup ?\bot = \alpha}(\cup_{id}) \qquad \frac{}{\vdash \alpha \cup \alpha = \alpha}(\cup_{idemp})$$

$$\frac{}{\vdash \alpha; (\beta; \gamma) = (\alpha; \beta); \gamma}(;_{assoc}) \qquad \frac{}{\vdash ?\top; \alpha = \alpha}(;_{id-l}) \qquad \frac{}{\vdash \alpha; ?\top = \alpha}(;_{id-r})$$

$$\frac{}{\vdash \alpha; (\beta \cup \gamma) = \big((\alpha; \beta) \cup (\alpha; \gamma)\big)}(dist - l) \qquad \frac{}{\vdash (\alpha \cup \beta); \gamma = \big((\alpha; \gamma) \cup (\beta; \gamma)\big)}(dist - r)$$

$$\frac{}{\vdash \alpha; ?\bot = ?\bot}(;_{annih-r}) \qquad \frac{}{\vdash ?\bot; \alpha = ?\bot}(;_{annih-l})$$

Figure 4.2: Idempotent semiring/KAT axioms for dRL

$$\frac{\vdash [\beta]\phi \qquad \vdash \alpha \leq \beta}{\vdash [\alpha]\phi}([\leq]) \qquad \frac{\vdash \langle\alpha\rangle\phi \qquad \vdash \alpha \leq \beta}{\vdash \langle\beta\rangle\phi}(\langle\leq\rangle)$$

Figure 4.3: dRL modality rules

[1]Equivalence of hybrid programs ($\alpha = \beta$) is syntactic sugar for $\alpha \leq \beta \wedge \beta \leq \alpha$, so this rule holds by definition.

4.5 Proposed Rules (Pending Proofs of Soundness)

Through experience and some preliminary experimentation, we expect these proof rules to be useful. However, for many of them we have not yet proved soundness (indicated in red), thus we set them apart in this section as proposed rules. We also believe some rules, such as $?_\vee$, can be derived from other proof rules. We present these rules in Fig. 4.4 and 4.5 to give the reader a more concrete understanding of dRL and a glimpse in the direction we are heading.

$$\frac{}{\vdash\ ?\top \cup (\alpha;\alpha^*) \leq \alpha^*}(unroll_l) \qquad \frac{}{\vdash\ ?\top \cup (\alpha^*;\alpha) \leq \alpha^*}(unroll_r)$$

$$\frac{\vdash (\alpha;\gamma) \cup \beta \leq \gamma}{\vdash \alpha^*;\beta \leq \gamma}(loop_l) \qquad \frac{\vdash (\gamma;\alpha) \cup \beta \leq \gamma}{\vdash \beta;\alpha^* \leq \gamma}(loop_r)$$

Figure 4.4: KAT axioms for * in dRL

$$\frac{\alpha \leq \gamma \wedge \beta \leq \gamma}{\alpha \cup \beta \leq \gamma}(\cup_l) \qquad \frac{\vdash \alpha \leq \beta \vee \alpha \leq \gamma}{\vdash \alpha \leq \beta \cup \gamma}(\cup_r) \qquad \frac{\vdash \alpha \leq \beta}{\vdash \alpha^* \leq \beta^*}(unloop)$$

$$\frac{}{\vdash (x := \theta) \leq (x := *)}(:= *) \qquad \frac{\vdash \alpha_1 \leq \alpha_2 \qquad \vdash \forall^{\alpha_2}\ \beta_1 \leq \beta_2}{\vdash (\alpha_1;\beta_1) \leq (\alpha_2;\beta_2)}(;)$$

$$\frac{}{\vdash (x' = \theta)^* = (x' = \theta)}('^*) \qquad \frac{\vdash \forall x, x^{(n)}\ (\mathscr{D} \to \mathscr{E})}{\vdash \mathscr{E} \leq \mathscr{D}}(DR)^3$$

$$\frac{\phi \to \psi}{?\phi \leq ?\psi}(?_{atomic}) \qquad \frac{}{\vdash ?(\phi \vee \psi) = (?\phi \cup ?\psi)}(?_\vee) \qquad \frac{}{\vdash ?(\phi \wedge \psi) = (?\phi;?\psi)}(?_\wedge)$$

$$\frac{}{\vdash (x := \theta_1; y := \theta_2) = (y := \theta_2; x := \theta_1)}(indep_{:=})^2$$

$$\frac{}{\vdash (x' = \theta_1; y' = \theta_2) = (y' = \theta_2; x' = \theta_1)}(indep')^2$$

$$\frac{}{\vdash (x := \theta_1; y' = \theta_2) = (y' = \theta_2; x := \theta_1)}(indep'_{:=})^2$$

Figure 4.5: dRL Misc Proposed Rules

[2]Where x and y are distinct variables, x does not occur free in θ_2, and y does not occur free in θ_1.

[3]$\mathscr{D}$ and $\mathscr{E}$ are differential algebraic constraints with the same changed variables. We treat x and its derivatives $x^{(n)}$ as variables and quantify over them so that they are independent.

4.6 Examples

Example 1 ($\alpha \cup \beta = \beta \leftrightarrow \alpha \leq \beta$)**.** In an idempotent semiring, we expect there to be a natural partial order induced by: $\alpha \cup \beta = \beta \leftrightarrow \alpha \leq \beta$, as described by Kozen in [53]. And, indeed, we find that this rule can also be derived for the refinement relation for hybrid programs from the above rules as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \alpha \leq \beta \leftrightarrow \alpha \leq \beta}ax}{\vdash (\alpha \leq \beta \wedge \beta \leq \beta) \leftrightarrow \alpha \leq \beta}\leq_{refl}}{\vdash \alpha \cup \beta \leq \beta \leftrightarrow \alpha \leq \beta}\cup_l}{\vdash (\alpha \cup \beta \leq \beta \wedge \beta \leq \alpha \cup \beta) \leftrightarrow \alpha \leq \beta}\cup_r, \leq_{refl}}{\vdash \alpha \cup \beta = \beta \leftrightarrow \alpha \leq \beta}\leq_{antisym}$$

Example 2 (Decomposing a system inside a loop)**.** dRL performs particularly well when determining refinement between HPs which differ only slightly, but within the context of a large and complicated system. In this example, the only difference between the two programs we are comparing is that the program on the left is setting variable x to a specific value θ, and the program on the right allows x to be assigned any value. This is an easy example for dRL:

$$\dfrac{\dfrac{\dfrac{}{\vdash \alpha \leq \alpha}\leq_{refl} \quad \dfrac{}{\vdash \forall^{(x:=\theta;\beta)}\,(x := \theta) \leq (x := *)}\forall_r, := * \quad \dfrac{}{\vdash \forall^{\beta}\,\beta \leq \beta}\forall_r, \leq_{refl}}{\vdash\ (\alpha; x := \theta; \beta) \leq (\alpha; x := *; \beta)};}{\vdash\ (\alpha; x := \theta; \beta)^* \leq (\alpha; x := *; \beta)^*}\ unloop$$

In order to take advantage of the similarities between these two HPs without refinement in dL, you would first have to deal with the outermost operator, in this case the Kleene star, by finding an appropriate loop invariant. By contrast, when using dRL, it is possible to be oblivious to the complexities of the two systems where they are the same and focus only on proving refinement in the places where they differ. This is a simple illustration of *breaking a system into parts*.

Example 3 (Guarded nondeterministic assignment)**.** This example illustrates the generalized safety envelopes use case. The nondeterministic assignment $x := *$ assigns an arbitrary real value to x. The test then restricts those values to be the ones satisfying the guard condition $\phi(x)$. Using guarded nondeterministic assignment can make a property easier to verify because it has stripped away all the details of the value of x except whatever is necessary for the proof of safety, in this case $\phi(x)$.

$$\phi(\theta) \vdash \ (x := \theta \ \leq \ x := *; ?\phi(x)) \tag{4.3}$$

Recall the discussion from Section 1.1 about designing an adaptive cruise control that is fuel efficient. We still want to verify that this controller doesn't allow the car to crash; however, the code that is specific for fuel efficiency may make it hard to verify that the car is still safe. In

this example, we can equate $x := \theta$ with the controller selecting the most fuel efficient choice. Additionally, we can think of $\phi(x)$ as being a condition on x that directly ensures safety. If that choice of θ satisfies property $\phi(\theta)$, in other words, if the fuel efficient controller satisfies the guard condition $\phi(\theta)$, then the fuel efficient controller is a refinement of the controller which is only concerned with safety. This is an example of *abstracting and implementation-specific design*.

Example 4 (Variable renaming)**.** Refinement ought to be able to handle two programs that are equivalent up to renaming of a variable. Suppose we modify HP $\alpha(x)$ by replacing variable x with a fresh variable X (so that X does not occur in α) as in (4.4).

$$\vdash \alpha(x) \leq (X := x; \alpha(X); X := *) \tag{4.4}$$

So long as the values of X are nondeterministically assigned at the end of the hybrid program, the refinement relation in (4.4) should hold.

Example 5 (Differential equations)**.** In the atomic case, it can often be easy to determine whether two programs are equivalent. For example, $(x := \theta_1) = (x := \theta_2)$ iff $\theta_1 = \theta_2$. However, the same rule does not apply to differential equations. Consider the following formulas:

$$\vdash (x' = 2) = (x' = 9) \tag{4.5}$$

$$\vdash (x' = 2, t' = 1) \neq (x' = 9, t' = 1) \tag{4.6}$$

In (4.5), because the duration of the evolution is nondeterministic, the effect of evolving for an arbitrary duration with a positive derivative is simply that the value of x after evolution is anything greater than or equal to the initial value of x. These two programs differ only in duration, but their reachability relation is the same so long as there is no variable that observes time.

However, in (4.6), time is now being recorded in the variable t. If the two evolutions differ in their duration, the discrepancy is recorded and therefore these programs are not equivalent.

Example 6 (Discrete assignment vs. continuous evolution)**.** Some hybrid programs look very different, but actually have the same set of transitions. This example extends the discussion of differential equations from Example 5 to checking refinement against discrete assignment.

$$\vdash (x := *; ?(x \geq 0)) = (x := 0; x' = 1) \tag{4.7}$$

$$\vdash x := 10 \leq (x := 1; x' = x) \tag{4.8}$$

The reachability set for the guarded nondeterministic assignment in (4.7) is for x to take any value greater than or equal to zero. And the reachability set if x is initially zero and then evolves with a positive derivative for an arbitrary amount of time is the same. Similar reasoning can be applied to (4.8).

Chapter 5

dR$\mathcal{L}$ and its Applications: Proposed and Future Work

5.1 MPC Design and Verification

A challenge we have seen using d$\mathcal{L}$ is when we have tried to verify a system that is very far removed in its design from the safety properties that we are trying to check. One such example, mentioned in Chapter 3, is the adaptive cruise controller designed to optimize fuel efficiency. The controller that optimizes fuel consumption (which would be included in the definition of function f) would obfuscate the properties of function f that ensure safe following.

$$[a_f := f(x_f, v_f, x_l, v_l); a_l := *; x_f' = v_f, v_f' = a_f, x_l' = v_l, v_l' = a_l](x_f < x_l) \tag{5.1}$$

On the other extreme, we could write a hybrid program with a controller that is nearly vacuously true. One that is, by perfect, implicit design, easy to verify satisfies the safety conditions.

$$\begin{aligned}&\Big[a_f := *;\\ &?([a_l := *; x_f' = v_f, v_f' = a_f, x_l' = v_l, v_l' = a_l](x_f < x_l));\\ &a_l := *;\\ &x_f' = v_f, v_f' = a_f, x_l' = v_l, v_l' = a_l\Big](x_f < x_l)\end{aligned} \tag{5.2}$$

This implicit controller is perfect by design; we call it MPC. MPC properties like (5.2) are easy to verify even without refinement due to their somewhat circular construction. However, implicit constructions are not exactly useful when it comes time to implement the model.

By using the refinement relation in dR$\mathcal{L}$, we hope to relate this easy-to-verify MPC property to our original property (5.1). If we can prove the following refinement, then property (5.1) must also hold:

$$(a_f := f(x_f, v_f, x_l, v_l)) \leq (a_f := *; ?([a_l := *; x_f' = v_f, v_f' = a_f, x_l' = v_l, v_l' = a_l](x_f < x_l))) \tag{5.3}$$

Previously, when proving properties about hybrid programs with implicit controllers in dℒ, we had to demonstrate as a side condition that there is some value which can in fact satisfy the implicit design, otherwise the proof would actually be vacuous. Now, the refinement relation in (5.3) cannot hold in the case where the implicit hybrid program satisfies the safety property vacuously, so we have a guarantee that the side condition was correctly and carefully handled.

5.1.1 Safety Envelope Verification

Related to MPC, but perhaps not taken to quite an extreme is the concept of verifying safety envelopes within which the controller is guaranteed to be safe, no matter what control choice it takes. This brings us a step closer to something that could be implementable, but is likely easier to prove than the explicit property.

This would allow us to put an intermediate step between the explicit controller, which is easy-to-implement, and the MPC controller, which is easy to verify.

$$(a_f := f(x_f, v_f, x_l, v_l)) \tag{5.4}$$

$$\leq a_f := *; ?(x_f + \frac{v_f^2}{-2a_f} < x_l \wedge a_f < 0) \tag{5.5}$$

$$\leq (a_f := *; ?([a_l := *; x_f' = v_f, v_f' = a_f, x_l' = v_l, v_l' = a_l](x_f < x_l))) \tag{5.6}$$

5.2 Refinement and Hierarchical Proofs

While exploring the use of refinement in the distributed car control and aircraft control case studies may reveal some specific examples where hierarchical proof structures can be retroactively injective using a refinement relation, ideally the structures would automatically arise during the proof search. Exploring new heuristics that leverage the refinement relation in a way that enforces hierarchical proof structures may lead to more efficient proof search algorithms. While designing and testing such proof search heuristics are likely out of scope for this thesis, they represent an interesting and possibly very fruitful extension of this research.

5.3 From Verification to Synthesis

Perhaps the most desirable way to close the gap between a verified model of a system and its implementation would be to automatically synthesize the implementation. This is only useful if: each step in the synthesis process maintains the verification guarantees; and the result of the synthesis adheres to the expectations and requirements of system designers. This challenge is amplified when the original model has many nondeterministic components.

Chapter 6

Conclusion

In conclusion, we are currently in the midst of designing dRℒ. We propose to continue developing dRℒ and its associated proof calculus. We hypothesize that dRℒ can improve the feasibility of theorem proving for hybrid systems by making it easier to break systems into smaller subsystems, abstract implementation-specific design details, leverage an iterative approach to system design, and maintain a modular proof structure. We will test our hypothesis on a collection of cutting-edge case studies that were previously verified using ad hoc and often tricky proof techniques. Finally, in addition to ensuring that dRℒ performs well on the case studies that we are most familiar with, we also plan to provide some theoretical reinforcement of our hypothesis by proving a decision procedure for at least the *-free fragment of dRℒ.

Stretch goals for this thesis include examining a wider domain of applications for dRℒ. Three such application areas (in order of priority) are: using implicit controllers and safety envelopes in hybrid programs, enforcing hierarchical proof structures through search heuristics that take advantage of the refinement operator, and synthesizing code from nondeterministic models.

Bibliography

[1] Platzer, A.: Differential dynamic logic for verifying parametric hybrid systems. In Olivetti, N., ed.: TABLEAUX. Volume 4548 of LNCS., Springer (2007) 216–232 1.1, 4.3

[2] Platzer, A., Quesel, J.D.: KeYmaera: A hybrid theorem prover for hybrid systems. In Armando, A., Baumgartner, P., Dowek, G., eds.: IJCAR. Volume 5195 of LNCS., Springer (2008) 171–178

[3] Platzer, A.: Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics. Springer, Heidelberg (2010) 2.2, 4.1, 4.2

[4] Platzer, A.: Logics of dynamical systems. In: LICS, IEEE (2012) 13–24 1.1, 2.2, 4.1, 4.2, 4.2

[5] Loos, S.M., Platzer, A., Nistor, L.: Adaptive cruise control: Hybrid, distributed, and now formally verified. In Butler, M., Schulte, W., eds.: FM. Volume 6664 of LNCS., Springer (2011) 42–56 1.1, 1.1

[6] Loos, S.M., Renshaw, D., Platzer, A.: Formal verification of distributed aircraft controllers. In: Proceedings of the 16th international conference on Hybrid systems: computation and control. HSCC '13, New York, NY, USA, ACM (2013) 125–130 1.1, 1.2, 1.1, 2.4

[7] Loos, S., Platzer, A.: Safe intersections: At the crossing of hybrid systems and verification. In: 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC). (2011) 1181–1186 1.1, 3.1

[8] Rajhans, A., Bhave, A., Loos, S., Krogh, B., Platzer, A., Garlan, D.: Using parameters in architectural views to support heterogeneous design and verification. In: 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC). (2011) 2705–2710 1.1, 3.3

[9] Arechiga, N., Loos, S., Platzer, A., Krogh, B.: Using theorem provers to guarantee closed-loop system properties. In: American Control Conference (ACC), 2012. (2012) 3573–3580 1.1, 1.1, 3.3

[10] Alur, R.: Formal verification of hybrid systems. In: 2011 Proceedings of the International Conference on Embedded Software (EMSOFT). (2011) 273–278 2.1

[11] Clark, M., Koutsoukos, X., Kumar, R., Lee, I., Pappas, G., Pike, L., Porter, J., Sokolsky, O.: A study on run time assurance for complex cyber physical systems. (2013) 2.1

[12] Mitsch, S., Quesel, J.D., Platzer, A.: Refactoring, refinement, and reasoning: A logical characterization for hybrid systems. In Jones, C.B., Pihlajasaari, P., Sun, J., eds.: FM.

(2014) 2.2

[13] Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement for symbolic model checking. Journal of the ACM (JACM) **50** (2003) 752–794 2.2

[14] Abrial, J.R., Butler, M., Hallerstede, S., Hoang, T.S., Mehta, F., Voisin, L.: Rodin: an open toolset for modelling and reasoning in event-b. International journal on software tools for technology transfer **12** (2010) 447–466 2.2

[15] Alur, R., Grosu, R., Lee, I., Sokolsky, O.: Compositional modeling and refinement for hierarchical hybrid systems. The Journal of Logic and Algebraic Programming **68** (2006) 105–128 2.2

[16] Hoare, C.A.R.: Communicating sequential processes. Volume 178. Prentice-hall Englewood Cliffs (1985) 2.2

[17] Kozen, D.: Kleene algebra with tests. ACM Transactions on Programming Languages and Systems (TOPLAS) **19** (1997) 427–443 2.2, 4.1, 4.4

[18] Kozen, D.: Netkata formal system for the verification of networks. In: Programming Languages and Systems. Springer (2014) 1–18 2.2

[19] Foster, N., Kozen, D., Milano, M., Silva, A., Thompson, L.: A coalgebraic decision procedure for NetKAT. Technical Report `http://hdl.handle.net/1813/36255`, Computing and Information Science, Cornell University (2014) POPL 2015, to appear. 2.2

[20] Chang, J., Cohen, D., Blincoe, L., Subramanian, R., Lombardo, L.: CICAS-V research on comprehensive costs of intersection crashes. Technical Report 07-0016, NHTSA (2007) 2.3

[21] Damm, W., Hungar, H., Olderog, E.R.: Verification of cooperating traffic agents. International Journal of Control **79** (2006) 395–421

[22] Dao, T.S., Clark, C.M., Huissoon, J.P.: Distributed platoon assignment and lane selection for traffic flow optimization. In: IEEE IV'08. (2008) 739–744 2.3

[23] Dao, T.S., Clark, C.M., Huissoon, J.P.: Optimized lane assignment using inter-vehicle communication. In: IEEE IV'07. (2007) 1217–1222 2.3

[24] Hall, R., Chin, C., Gadgil, N.: The automated highway system / street interface: Final report. PATH Research Report UCB-ITS-PRR-2003-06, UC Berkeley (2003) 2.3

[25] Hall, R., Chin, C.: Vehicle sorting for platoon formation: Impacts on highway entry and troughput. PATH Research Report UCB-ITS-PRR-2002-07, UC Berkeley (2002) 2.3

[26] Hsu, A., Eskafi, F., Sachs, S., Varaiya, P.: Design of platoon maneuver protocols for IVHS. PATH Research Report UCB-ITS-PRR-91-6, UC Berkeley (1991) 2.3

[27] Ioannou, P.A.: Automated Highway Systems. Springer (1997) 2.3

[28] Jula, H., Kosmatopoulos, E.B., Ioannou, P.A.: Collision avoidance analysis for lane changing and merging. PATH Research Report UCB-ITS-PRR-99-13, UC Berkeley (1999) 2.3

[29] Horowitz, R., Tan, C.W., Sun, X.: An efficient lane change maneuver for platoons of vehicles in an automated highway system. PATH Research Report UCB-ITS-PRR-2004-

16, UC Berkeley (2004) 2.3

[30] Shladover, S.E.: Effects of traffic density on communication requirements for Cooperative Intersection Collision Avoidance Systems (CICAS). PATH Working Paper UCB-ITS-PWP-2005-1, UC Berkeley (2004) 2.3

[31] Stursberg, O., Fehnker, A., Han, Z., Krogh, B.H.: Verification of a cruise control system using counterexample-guided search. Control Engineering Practice **38** (2004) 1269–1278 2.3

[32] Varaiya, P.: Smart cars on smart roads: problems of control. IEEE Trans. Automat. Control **38** (1993) 195–207 2.3

[33] Wongpiromsarn, T., Mitra, S., Murray, R.M., Lamperski, A.G.: Periodically controlled hybrid systems: Verifying a controller for an autonomous vehicle. In Majumdar, R., Tabuada, P., eds.: HSCC. Volume 5469 of LNCS., Springer (2009) 396–410

[34] Chee, W., Tomizuka, M.: Vehicle lane change maneuver in automated highway systems. PATH Research Report UCB-ITS-PRR-94-22, UC Berkeley (1994) 2.3

[35] Johansson, R., Rantzer, A., eds.: Nonlinear and Hybrid Systems in Automotive Control. Society of Automotive Engineers Inc. (2003)

[36] Althoff, M., Althoff, D., Wollherr, D., Buss, M.: Safety verification of autonomous vehicles for coordinated evasive maneuvers. In: IEEE IV'10. (2010) 1078 – 1083 2.3

[37] Berardi, L., Santis, E., Benedetto, M., Pola, G.: Approximations of maximal controlled safe sets for hybrid systems. In Johansson, R., Rantzer, A., eds.: Nonlinear and Hybrid Systems in Automotive Control, Springer (2003) 335–350 2.3

[38] Loos, S.M., Platzer, A., Nistor, L.: Adaptive cruise control: Hybrid, distributed, and now formally verified. In Butler, M., Schulte, W., eds.: FM 2011: Formal Methods, 17th International Symposium on Formal Methods, Limerick, Ireland. Volume 6664 of LNCS., Springer (2011) 42–56 2.3, 3.1, 3.1

[39] Tomlin, C., Pappas, G.J., Sastry, S.: Conflict resolution for air traffic management. IEEE T. Automat. Contr. **43** (1998) 509–521 2.4

[40] Hu, J., Prandini, M., Sastry, S.: Probabilistic safety analysis in three-dimensional aircraft flight. In: CDC. (2003) 2.4

[41] Dowek, G., Muñoz, C., Carreño, V.A.: Provably safe coordinated strategy for distributed conflict resolution. In: AIAA-2005-6047. (2005) 2.4

[42] Umeno, S., Lynch, N.A.: Proving safety properties of an aircraft landing protocol using I/O automata and the PVS theorem prover. In Misra, J., Nipkow, T., Sekerinski, E., eds.: FM. Volume 4085 of LNCS., Springer (2006) 64–80 2.4

[43] Galdino, A.L., Muñoz, C., Ayala-Rincón, M.: Formal verification of an optimal air traffic conflict resolution and recovery algorithm. In Leivant, D., de Queiroz, R., eds.: WoLLIC. Volume 4576 of LNCS., Springer (2007) 177–188 2.4

[44] Hwang, I., Kim, J., Tomlin, C.: Protocol-based conflict resolution for air traffic control. Air Traffic Control Quarterly **15** (2007) 1–34 2.4

[45] Umeno, S., Lynch, N.A.: Safety verification of an aircraft landing protocol: A refinement approach. In Bemporad, A., Bicchi, A., Buttazzo, G., eds.: HSCC. Volume 4416 of LNCS., Springer (2007) 557–572 2.4

[46] Johnson, T., Mitra, S.: Parameterized verification of distributed cyber-physical systems:an aircraft landing protocol case study. In: ACM/IEEE ICCPS. (2012) 2.4

[47] Duperret, J.M., Hafner, M.R., Del Vecchio, D.: Formal design of a provably safe roundabout system. (In: IEEE/RSJ IROS) 2006–2011 2.4

[48] Platzer, A., Clarke, E.M.: Formal verification of curved flight collision avoidance maneuvers: A case study. In: FM. Volume 5850 of LNCS., Springer (2009) 547–562 2.4

[49] Košecká, J., Tomlin, C., Pappas, G., Sastry, S.: 2-1/2D conflict resolution maneuvers for ATMS. In: CDC. Volume 3., Tampa, FL, USA (1998) 2650–2655 2.4

[50] Bicchi, A., Pallottino, L.: On optimal cooperative conflict resolution for air traffic management systems. IEEE Trans. ITS **1** (2000) 221–231 2.4

[51] Pallottino, L., Scordio, V., Frazzoli, E., Bicchi, A.: Decentralized cooperative policy for conflict resolution in multi-vehicle systems. IEEE Trans. on Robotics **23** (2007) 2.4

[52] Loos, S.M., Witmer, D., Steenkiste, P., Platzer, A.: Efficiency analysis of formally verified adaptive cruise controllers. In Hegyi, A., Schutter, B.D., eds.: Intelligent Transportation Systems (ITSC), 16th International IEEE Conference on, October 6-9, The Hague, Netherlands, Proceedings. (2013) 1565–1570 3.4

[53] Kozen, D.: The design and analysis of algorithms. Springer (1992) 1