Hybrid Dynamical Systems Logic and Its Refinements

André Platzer

[a] *Karlsruhe Institute of Technology, Karlsruhe, Germany*

Abstract

Hybrid dynamical systems describe the mixed discrete dynamics and continuous dynamics of cyber-physical systems such as aircraft, cars, trains, and robots. To justify correctness properties of the safety-critical control algorithms for their physical models, *differential dynamic logic* (dL) provides deductive specification and verification techniques implemented in the theorem prover KeYmaera X. The logic dL is useful for proving, e.g., that all runs of a hybrid dynamical system α satisfy safety property φ (i.e., $[\alpha]\varphi$), or that there is a run of the hybrid dynamical system α ultimately reaching the desired goal φ (i.e., $\langle\alpha\rangle\varphi$). Logical combinations of dL's operators naturally represent safety, liveness, stability and other properties. Variations of dL serve additional purposes. *Differential refinement logic* (dRL) adds an operator $\alpha \leq \beta$ expressing that hybrid system α refines hybrid system β, which is useful, e.g., for relating concrete system implementations α to their abstract verification models β. Just like dL, dRL is a logic closed under all operators, which opens up systematic ways of simultaneously relating systems and their properties, of reducing system properties to system relations or, vice versa, reducing system relations to system properties. A second variant of dL, *differential game logic* (dGL), adds the ability of referring to winning strategies of players in hybrid games, which is useful for establishing correctness properties where the actions of different agents may interfere either because they literally compete with one another or because they may interact accidentally. In the theorem prover KeYmaera X, dL and its variations have been used for verifying ground robot obstacle avoidance, the Federal Aviation Administration's Next-Generation Airborne Collision Avoidance System ACAS X, and the Federal Railroad Administration's train control model.

Keywords: Differential dynamic logic, Differential refinement logic, Differential game logic, Hybrid systems, Hybrid games, Theorem proving

1. Introduction

Hybrid dynamical systems, or *hybrid systems* for short, describe systems with a mixture of discrete dynamics and continuous dynamics and have many important applications [1–13]. The most canonical applications are those where the discrete dynamics of stepwise computation comes from computer controllers

Sci. Comput. Program. **239**, pp. 103179, 2025.
DOI: 10.1016/j.scico.2024.103179

while the continuous dynamics following continuous functions comes from physical motion, as, e.g., in cars, aircraft, trains, and robots. In those cases a discrete computer reaches decisions, e.g., to steer, accelerate or brake, while the vehicle moves continuously, affected by the braking and steering decisions. Other applications of hybrid systems include biological systems [14, 15] and chemical processes [16, 17]. Many of these applications are safety-critical [8, 13], which explains why a great deal of attention has been paid to the development of techniques that help to either find mistakes in controllers, or to verify that there are no mistakes by establishing that the controllers are guaranteed to satisfy desired correctness properties in the hybrid dynamical systems model [3, 6, 18, 19]. Dealing with the real world is always difficult, which explains why verification of hybrid dynamical systems is challenging, even undecidable [3]. However, the benefits of a more reliable system outweigh the verification cost whenever applications are important enough that mistakes could incur significant financial loss or even risk loss of life. This is the case in automotive, aerospace, robotics, and railway domains but also the energy domain and chemical process control.

This article reports on logics for hybrid dynamical systems. *Differential dynamic logic* (dL) [20–23] is a logic for specifying and verifying correctness properties of hybrid dynamical systems [18, 19, 24] that is also implemented in the hybrid systems theorem prover KeYmaera X [25] that is available on the web[1]. KeYmaera X has been used in interesting applications, including aircraft collision avoidance [26], ground robot obstacle avoidance [27], and train control [28]. The logic dL is the foundation for a whole family of logics with several useful variations. *Differential refinement logic* (dRL) [29] adds refinement relations between hybrid systems as a first-class citizen logical operator, which is useful for relating different models of a system or for relating those to properties of the systems. *Differential game logic* (dGL) [19, 30, 31] adds the ability to refer to the existence of winning strategies for games played on hybrid systems, which is useful for describing systems where different agents interact and may possibly interfere. The main purposes of all three of these logics will be sketched in this article. Other extensions of dL are useful but beyond the scope of this article, such as *quantified differential dynamic logic* (QdL) for distributed hybrid systems [32], *stochastic differential dynamic logic* (SdL) for stochastic hybrid systems [33], *dynamic logic for communicating hybrid programs* ($\mathsf{dL}_{\mathrm{CHP}}$) with parallelism [34], as well as *hybrid-nominal differential dynamic logic* (dHL) whose nominals support hyper properties such as hybrid information flow [35].

A technical survey of classical differential dynamic logic appeared at LICS'12 [21], a short high-level survey of its principles at IJCAR'16 [36]. Information on the theory of dL can be found in a book [18]. A very readable comprehensive account of dL and dGL is provided in a textbook [19]. This article is an extended version of an invited paper at ABZ'23 [37] extending it with detail on reasoning principles, explanations, intuition, examples, theorems, and justifications of the purpose, as well as more detail on games, and a short survey of applications.

[1] KeYmaera X is open-source software available at http://keymaeraX.org/

2. Differential Dynamic Logic Concepts

Differential dynamic logic is an extension of dynamic logic [38, 39], which was meant for program verification, and was generalized to Java programs [40].

2.1. Differential Dynamic Logic

Differential dynamic logic dL [18–24] provides a programming language for hybrid systems called *hybrid programs*, which works like an ordinary imperative programming language except that it supports nondeterminism to reflect the inherent uncertainty of the behavior of the real world and, crucially, supports differential equations $x' = f(x)\,\&\,Q$ to describe continuous dynamics within the evolution domain constraint Q (an overview can be found in the literature [18, 19, 24]). Besides the operators of first-order logic of real arithmetic, dL crucially provides modalities for hybrid programs α, where the dL formula $[\alpha]\varphi$ means that all final[2] states reachable by hybrid program α satisfy formula φ (*safety*), while the formula $\langle\alpha\rangle\varphi$ means that some final state reachable by hybrid program α satisfies formula φ (*liveness*).

A dL formula φ is *valid*, written $\vDash \varphi$, iff φ is true in all states. Valid dL formulas are the most valuable formulas because one can rely on them no matter what state the world is in. Typical patterns for safety properties are dL formulas of the form:

$$\psi \to [\alpha]\phi \tag{1}$$

dL formula (1) is valid iff in every state where the precondition formula ψ is true it is the case that, after all runs of hybrid program α, postcondition formula ϕ holds. Typical patterns for liveness properties are dL formulas of the form:

$$\psi \to \langle\alpha\rangle\phi \tag{2}$$

dL formula (2) is valid iff in every state where the precondition formula ψ is true it is the case that there is a run of hybrid program α that leads to a (final) state where the postcondition formula ϕ holds.

The dL shape (1) is akin to Hoare triples $\{\psi\}\alpha\{\phi\}$ except generalized to hybrid systems. Like a Hoare triple for partial correctness, it does not say whether α terminates, just that ϕ holds after it did. But since control applications are typically meant to run for an arbitrary amount of time and stop at any moment anyhow, termination is less of a concern than in conventional discrete programs. The dL formula (2) says that, if ψ holds initially, then α can run to a final state in which ϕ is true. In particular, α can terminate. Note, however, that the presence of nondeterminism is typically central in hybrid systems applications giving the universal and existential quantification over runs of $[\alpha]$ and $\langle\alpha\rangle$, respectively,

[2]Most hybrid programs are written such that all intermediate states are final states, because of their emphasis on nondeterminism. For instance, nondeterministic repetitions and differential equations in hybrid programs can stop nondeterministically after any number of rounds or any amount of time. The extension to differential temporal dynamic logic dTL, formalized in PVS [41], handles cases where other intermediate states play a role [42].

a genuinely interesting and useful meaning. The following dL formulas are more general and have no counterpart in Hoare triples.

In the same spirit, the dL formula $[\alpha]\phi$ already is "its own" weakest liberal precondition. The weakest liberal precondition of ϕ under α is the weakest of all the formulas that imply that ϕ holds after running α (if α terminates) [43]. But that is exactly what the dL formula $[\alpha]\phi$ already expresses, that always after running α, ϕ holds true. Yet, $[\alpha]\phi$ is not an extra-logical predicate transformer defined via an implicit characterization of a universal property among the set of all formulas, but rather simply a logical formula of dL itself. Similarly, for deterministic α, $\langle\alpha\rangle\phi$ is a syntactic formula that is "its own" weakest precondition, since $\langle\alpha\rangle\phi$ is true iff α can complete a (terminating) run to a state in which ϕ is true.

Even if (1) and (2) are fairly common shapes for dL formulas, dL's modalities can also be combined by propositional or first-order logic operators. For instance, the following formula expresses that, if formula ψ is true initially, then all runs of α satisfy ϕ_1 and also all runs of α satisfy ϕ_2 (which can easily be proved in dL to be equivalent to the dL formula $\psi \rightarrow [\alpha](\phi_1 \wedge \phi_2)$ that expresses both postconditions simultaneously):

$$\psi \rightarrow [\alpha]\phi_1 \wedge [\alpha]\phi_2 \tag{3}$$

Mixed safety and liveness formulas, are, of course, also possible in dL. The following formula expresses that if ψ holds initially, then all runs of α satisfy ϕ_1 and at least one run of α also reaches a state where ψ is true again and, if ψ_2 also holds initially, then there is even a run of β after which ϕ_2 holds:

$$\psi \rightarrow \big([\alpha]\phi_1 \wedge \langle\alpha\rangle\psi \wedge (\psi_2 \rightarrow \langle\beta\rangle\phi_2)\big) \tag{4}$$

The outer gray parentheses are superfluous, as they are implied by the notational convention that unary operators including $\forall, \exists, [\cdot], \langle\cdot\rangle$ bind stronger than binary operators and that $\wedge$ binds stronger than $\vee$, which binds stronger than $\rightarrow, \leftrightarrow$.

Just like its quantifiers, dL's operators can also be nested. For instance, $[\alpha]\langle\beta\rangle\varphi$ means that after all runs of hybrid system α there is still a run of hybrid system β to reach a state satisfying φ, which is useful, e.g., to describe that system β can always get all α behavior back into a safe spot. Stability properties nest even more operators of dL. For example, *stability* of the origin for the differential equation $x' = f(x)$ is characterized by the dL formula [44]:

$$\forall\varepsilon{>}0\,\exists\delta{>}0\,\forall x\,(\mathcal{U}_\delta(x=0) \rightarrow [x' = f(x)]\mathcal{U}_\varepsilon(x=0)) \tag{5}$$

The δ-neighborhood $\mathcal{U}_\delta(x=0)$ of the set of states where formula $x=0$ is true is definable by the formula $x^2 < \delta^2$. The dL formula (5) expresses stability by saying that for every desired ε-neighborhood of the origin there is a δ-neighborhood of the origin from which all solutions of the differential equation $x' = f(x)$ always stay within that ε-neighborhood of the origin. *Attractivity* of the origin for the differential equation $x' = f(x)$ is characterized by the dL formula [44]:

$$\exists\delta{>}0\,\forall x\,(\mathcal{U}_\delta(x=0) \rightarrow \forall\varepsilon{>}0\,\langle x' = f(x)\rangle[x' = f(x)]\mathcal{U}_\varepsilon(x=0)) \tag{6}$$

The dL formula (6) expresses that there is a δ-neighborhood of the origin from which the differential equation eventually stays within every ε-neighborhood of the origin forever. *Asymptotic stability* of the origin is characterized by the conjunction of dL formulas (5) and (6) [44]. The logic dL is a proper logic, so closed under all operators, which explains why the single logic dL can be used to characterize so many different properties of hybrid systems. Other properties such as controllability and reactivity can be expressed as well [45].

Example 1 (Two gear car). As a specific example consider this dL formula:

$$b > 0 \land A < 0 \land v^2 \leq 2b(m-x) \land v^2 \leq 2(-A)(m-x) \rightarrow \\ [(a := A \cup a := -b); \{x' = v, v' = a\}]\, x \leq m$$

It expresses that, under some assumptions on the parameters, a simple car that can, by a nondeterministic choice ($\cup$), either drive with acceleration A or with acceleration $-b$ following a differential equation system for any arbitrary amount of time, in which the time-derivative x' of position x is velocity v, whose time-derivative is a, will always satisfy $x \leq m$. More interesting car models repeatedly choose different gears corresponding to different accelerations. Since the above simplistic car model only has a single initial choice, and either choice is possible, nondeterministically, its safety needs positive b but negative A, as well as a safe distance for A and b compared to the velocity.

Axiomatics. While it is crucial that dL has a simple and elegant unambiguous mathematical semantics [18–24] such that all dL formulas have a clear meaning aligning with operational intuition, it is just as important that the logic dL comes with a proof calculus with which the validity of dL formulas can be verified rigorously [18–24, 46]. A dL formula φ is *provable*, written $\vdash \varphi$, iff it has a proof from the axioms of dL using the proof rules of dL. For example, the dL calculus includes the axiom of nondeterministic choice:

$$[\cup]\ [\alpha \cup \beta]P \leftrightarrow [\alpha]P \land [\beta]P$$

Axiom [$\cup$] states that all runs of a hybrid program $\alpha \cup \beta$ that has a nondeterministic choice between hybrid program α and hybrid program β satisfy the postcondition P iff all runs of hybrid program α satisfy P and, independently, all runs of hybrid program β satisfy P. This equivalence is true in every state and can be used in every context. Axiom [$\cup$] can be used to decompose its left-hand side $[\alpha \cup \beta]P$ to its corresponding right-hand side $[\alpha]P \land [\beta]P$. This has the benefit of making all hybrid programs in the remaining formulas simpler and smaller. With this simple axiom [$\cup$], dL captures the essence of all finite branching behavior in systems [38] and supports the nondeterministic uncertainty that is so crucial for describing the real world such as when we do not know whether another vehicle will accelerate (α) or brake (β) so that both behaviors need to be safe (P).

Additionally, the dL calculus includes the axiom of sequential composition

$$[;]\ [\alpha;\beta]P \leftrightarrow [\alpha][\beta]P$$

Axiom [;] states that all runs of a hybrid program $\alpha;\beta$ with a sequential composition between hybrid program α and β satisfy the postcondition P iff after all runs of α it is true that all runs of β satisfy the postcondition P. This axiom captures the essence of behavior happening one step at a time such as when a car first runs its discrete control actions and then follows its continuous plant dynamics. By using axiom [;] to decompose its left-hand side $[\alpha;\beta]P$ to its right-hand side $[\alpha][\beta]P$, all remaining hybrid programs also get simpler. The subformula $[\beta]P$ itself clearly is simpler but it also is a more difficult postcondition for the modality $[\alpha]$. These equivalence decompositions from left to right are well-founded, because hybrid programs can only get smaller finitely often.

Example 2. Using axiom [;] in the direction from left to right on Example 1 will equivalently decompose its sequential composition into two separate box modalities, one for the discrete program, and one for the differential equation:

$$b>0\land A<0\land v^2\leq 2b(m-x)\land v^2\leq 2(-A)(m-x)\rightarrow [a:=A\cup a:=-b][x'=v,v'=a]\,x\leq m$$

A subsequent use of axiom [∪] in the direction from left to right will equivalently split the nondeterministic choice into its two possible cases:

$$b>0\land A<0\land v^2\leq 2b(m-x)\land v^2\leq 2(-A)(m-x)\rightarrow [a:=A][x'=v,v'=a]\,x\leq m\land [a:=-b][x'=v,v'=a]\,x\leq m$$

An assignment axiom [:=] can now, equivalently, substitute in both assignments:

$$b>0\land A<0\land v^2\leq 2b(m-x)\land v^2\leq 2(-A)(m-x)\rightarrow [x'=v,v'=A]\,x\leq m\land [x'=v,v'=-b]\,x\leq m$$

Of course, dL's axioms for differential equations are fundamental to its success. The most exciting part of dL's differential equations axiomatization [23] is that it decides all equational properties of differential equations and decides all invariants of differential equations [46]. What is significantly less powerful but also easier to understand is dL's axiom that replaces differential equations $x'=f(x)$ with their solutions $y(t)$ and a universal quantifier for time $t\geq 0$:

$$[']\ [x'=f(x)]p(x)\leftrightarrow\forall t{\geq}0\,[x:=y(t)]p(x)\quad (y'(t)=f(y),y(0)=x)$$

Example 3 (Two gear car proof). Using axiom [′] for the last part of the last formula of Example 2 yields the following equivalence:

$$[x'=v,v'=-b]\,x\leq m\leftrightarrow\forall t{\geq}0\,[x:=-b/2t^2+vt+x]x\leq m$$

A subsequent use of the axiom [:=] substitutes in the assignment:

$$[x'=v,v'=-b]\,x\leq m\leftrightarrow\forall t{\geq}0\,(-b/2t^2+vt+x\leq m)$$

Plugging this equivalence (and a similar one for $x'=v,v'=A$ into the last formula of Example 2 yields an equivalent formula in first-order real arithmetic,

which is decidable by quantifier elimination [47–49] and will turn out to be valid. Consequently, following the chain of equivalences backwards, this implies that the formula in Example 1 is valid. Proofs in dL can work like in this example, except that they are typically conducted with all the additional proof structuring principles of sequent calculus [18, 19]. The resulting dL proof is shown in Fig. 1 where the sequent $\Gamma \vdash \Delta$ is short for $\bigwedge_{F\in\Gamma} F \rightarrow \bigvee_{G\in\Delta} G$ and $\mathcal{A}$ is short for the assumptions $b > 0 \land A < 0 \land v^2 \leq 2b(m-x) \land v^2 \leq 2(-A)(m-x)$ and $\mathbb{R}$ marks the use of real arithmetic decision procedures.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{*}{\mathcal{A} \vdash \forall t{\geq}0\,(A/2t^2 + vt + x \leq m) \land \forall t{\geq}0\,(-b/2t^2 + vt + x \leq m)}\,\mathbb{R}}{\mathcal{A} \vdash \forall t{\geq}0\,[x := A/2t^2 + vt + x]x \leq m \land \forall t{\geq}0\,[x := -b/2t^2 + vt + x]x \leq m}\,[:=]}{\mathcal{A} \vdash [x' = v, v' = A]\,x \leq m \land [x' = v, v' = -b]\,x \leq m}\,[']}{\mathcal{A} \vdash [a := A \cup a := -b][x' = v, v' = a]\,x \leq m}\,[:=]}{\mathcal{A} \vdash [a := A \cup a := -b][x' = v, v' = a]\,x \leq m}\,[\cup]}{\mathcal{A} \vdash [(a := A \cup a := -b); \{x' = v, v' = a\}]\,x \leq m}\,[;]}{\vdash \mathcal{A} \rightarrow [(a := A \cup a := -b); \{x' = v, v' = a\}]\,x \leq m}\,{\rightarrow}\mathrm{R}
$$

Figure 1: A proof of Example 1 in dL's sequent calculus

The advantage of the axiom $[']$ is that its use explicitly replaces a differential equation by its solution. The downside is that the resulting universal quantifier quickly leads to highly undecidable arithmetic [50] and that most differential equations do not even have closed-form solutions to begin with [51]. This is where dL's axioms for implicitly proving properties of differential equations without explicitly solving [23, 46] are significantly more general.

The easiest example is the differential invariant axiom for equality to 0 [19]:

$$\mathrm{DI}_{=0}\ \big([x' = f(x)]\,e = 0 \leftrightarrow e = 0\big) \leftarrow [x' = f(x)]\,(e)' = 0$$

The axiom is written using the backwards implication $\varphi \leftarrow \psi$, which is syntactic sugar for the forward implication $\psi \rightarrow \varphi$. The axiom gives a conditional equivalence: The equality $e = 0$ is always true in the future when following the differential equation $x' = f(x)$ iff the equality $e = 0$ is true right now, provided the differential $(e)' = 0$ is always true when following the differential equation. Since the differential $(e)'$ after the ODE intuitively refers to the rate of change of term e, axiom $\mathrm{DI}_{=0}$ expresses that e always stays zero if it starts zero and has a zero rate of change. Defining a rate of change in a sound way is a nontrivial challenge, requiring the use of differentials [23], but the intuitive reading suffices for this overview. Furthermore, the axiom $\mathrm{DI}_{=0}$ is a simple illustration of the phenomenon that dL proves equivalences between current and future truth. In fact, one can, for equalities, even derive the *differential equational equivalence*:

$$\mathrm{DI}'_{=0}\ [x' = f(x)]\,e = 0 \leftrightarrow e = 0 \land [x' = f(x)]\,(e)' = 0$$

It expresses that an equation is always true after a differential equation iff it is currently true and the differential $(e)' = 0$ is always true after the differential

equation. Besides proving an equivalence of a current and future truth, this gives rise to an induction principle [46].

The dL proof calculus is a sound and complete axiomatization of hybrid systems relative to either discrete dynamics [22] or to continuous dynamics [20, 22]. That is, every formula it proves is valid (true in all states), but also every valid formula is provable in the dL calculus from elementary properties of the respective subdynamics.

Theorem 1 (Relative completeness of dL [20, 22, 23])**.** *The* dL *calculus is a* sound and complete axiomatization *of hybrid systems relative to any differentially expressive*[3] *logic L (such as the first-order logic FOD of differential equations or the discrete dynamic logic DL), i.e., every provable* dL *formula is valid and every valid* dL *formula can be proved from L tautologies:*

$$\vDash \phi \quad \textit{iff} \quad \mathrm{Taut}_L \vdash \phi$$

For differential equation invariants, dL's axioms give a sound and complete axiomatization [46, 52] with which *all* true arithmetic invariants of (polynomial) differential equations can be proved in dL while *all* false ones can be disproved in dL, thereby giving a logical decision procedure. The soundness and completeness theorem for differential equations in dL's axiomatization is actually much stronger, but one easily understandable corollary is the following, giving a pure logic-based decision procedure of differential equation invariants:

Theorem 2 (Completeness of dL for differential equation invariants [46, 52])**.** *The* dL *calculus is a* sound and complete axiomatization *of differential equation invariants, both if true and if false, i.e., when Q, P are formulas of first-order logic of real arithmetic, then*

$$\vDash P \to [x' = f(x)\,\&\,Q]P \quad \textit{iff} \quad \vdash P \to [x' = f(x)\,\&\,Q]P$$
$$\vDash \neg(P \to [x' = f(x)\,\&\,Q]P) \quad \textit{iff} \quad \vdash \neg(P \to [x' = f(x)\,\&\,Q]P)$$

Theorem 3 (Analytic completeness [46, 52])**.** *The differential radical invariant axiom DRI derives in* dL *(where, e.g., Q is a real arithmetic formula formed from conjunctions and disjunctions of strict inequalities and $e^{\bullet(*)} = 0$ is a finite computable real arithmetic formula indicating that all Lie derivatives are 0):*

$$\mathrm{DRI}\ [x' = f(x)\,\&\,Q]e = 0 \leftrightarrow \big(Q \to e^{\bullet(*)} = 0\big)$$

Like its inductive consequence, derived axiom $\mathrm{DI}'_{=0}$, derived axiom DRI is an equivalence of future and current truth, yet its right-hand side is even purely arithmetical! For more general formulations with arbitrary domain constraints and generalizations to Noetherian functions as well as decidability of algebraic

[3] A logic L closed under first-order connectives is *differentially expressive* (for dL) if every dL formula ϕ has an equivalent $\phi^\flat$ in L and all equivalences of the form $\langle x' = f(x)\rangle G \leftrightarrow (\langle x' = f(x)\rangle G)^\flat$ for G in L are provable in its calculus.

properties of algebraic hybrid systems, see elsewhere [46, 52]. Similar soundness and completeness results hold for invariants of switched systems [53]. Liveness properties and existence properties of differential equations have corresponding proof principles derived in dL [54] and stability properties have proof principles derived in dL [44, 55] using Lyapunov functions. These syntactic derivations give the best of both worlds: fast-paced high-level reasoning yet with strong soundness guarantees coming from the parsimonious axiomatic foundation of dL. In fact, dL's syntactic derivations led to the discovery of several mistakes in prior liveness verification principles for differential equations [54] and in previously reported Lyapunov functions [44, 55].

2.2. Differential Refinement Logic

Specifying and verifying correctness properties of hybrid systems is important and useful, and dL is a versatile logic with a powerful proof calculus for the job. But some aspects of hybrid systems correctness go beyond what dL is naturally meant for. *Differential refinement logic* (dRL) [29, 56] adds a refinement operator between hybrid programs where the dRL formula $\alpha \leq \beta$ means that hybrid system α refines hybrid system β. That is, dRL formula $\alpha \leq \beta$ is true in a state whenever all states reachable from that state by following the transitions of α can also be reached by following the transitions of β. Note that $\alpha \leq \beta$ is a formula, however, not just a judgment, so it is true in some states and false in others. Whether all α behavior is subsumed by some β behavior may well depend on where the systems start, for which case formulas of the form $P \rightarrow \alpha \leq \beta$ are useful to condition on formula P being true initially. The refinement operator is useful, e.g., as $\gamma \leq \alpha$ to say that all runs of a concrete controller implementation γ are also runs of the abstract verification model α.

Example 4 (Two gear car refinements). Consider a time-triggered control variation of Example 1 with some test condition S_T when it is safe to accelerate for a reaction time of T, where the clock $t' = 1$ is reset via $t := 0$ before the differential equation and bounded by the evolution domain constraint $t \leq T$:

$$b > 0 \land A \geq 0 \land v^2 \leq 2b(m-x) \rightarrow \underbrace{[(?S_T; a := A \cup a := -b); t := 0; \{x' = v, v' = a, t' = 1 \,\&\, t \leq T\}]}_{car_T} x \leq m$$

Also consider an event-triggered control variation of Example 1 that uses some other test condition S_E and an event condition E to stop the differential equation before missing the critical event of having to brake:

$$b > 0 \land A \geq 0 \land v^2 \leq 2b(m-x) \rightarrow \underbrace{[(?S_E; a := A \cup a := -b); t := 0; \{x' = v, v' = a, t' = 1 \,\&\, E\}]}_{car_E} x \leq m$$

The time-triggered system is easy to implement yet the event-triggered system is easy to verify, because it merely reacts to the event in E as needed rather

than having to predict the impact of the acceleration control decision for the reaction time T. A dRL proof of safety of the time-triggered system derives from the simpler proof of safety of the event-triggered system together with a simple proof that the time-triggered system refines the event-triggered system [56]:

$$b > 0 \land A \geq 0 \land v^2 \leq 2b(m-x) \rightarrow car_T \leq car_E \land [car_E]\, x \leq m \tag{7}$$

Axiomatics. The box refinement rule shows that, if precondition P is true, then all runs of the concrete system γ satisfy postcondition Q (conclusion below rule bar), provided that the same implication holds for the abstract system α (left premise) and that the concrete system γ refines the abstract system α when starting in any state satisfying the precondition P (right premise).

$$[\leq]\ \frac{P \rightarrow [\alpha]Q \quad P \rightarrow \gamma \leq \alpha}{P \rightarrow [\gamma]Q}$$

The box refinement rule $[\leq]$ reduces one box property (conclusion) to another box property (left premise) and a refinement property (right premise), which is clever if the abstract system α is easier to verify than the concrete system γ. Even if the abstract system α has more behavior than the concrete γ from initial states satisfying P according to the second premise, its description and its proof of safety may still be easier. This happens, for example, when the abstract system α is more nondeterministic leaving out implementation detail that is important for performance of the actual implementation but quite irrelevant to safety. The logic behind rule $[\leq]$ can also be summarized more succinctly with the box refinement axiom that it derives from, saying that if α refines β then all α runs satisfy P provided that all β runs satisfy P:

$$[\leq]\ \alpha \leq \beta \rightarrow ([\alpha]P \leftarrow [\beta]P)$$

A similar diamond refinement rule handles refinements of $\langle\cdot\rangle$ properties (conclusion and left premise), but the converse refinement is required (right premise), because only if the hybrid system α refining the system γ from any state satisfying P can reach Q can the more flexible system γ reach Q, too:

$$\langle\leq\rangle\ \frac{P \rightarrow \langle\alpha\rangle Q \quad P \rightarrow \alpha \leq \gamma}{P \rightarrow \langle\gamma\rangle Q}$$

Example 5. By axiom $[\leq]$, the provable dRL formula (7) in Example 4 with its event-triggered safety and time-triggered refinement of the event-triggered system directly yields a simple dRL proof of safety of the time-triggered system:

$$b > 0 \land A \geq 0 \land v^2 \leq 2b(m-x) \rightarrow [car_T]\, x \leq m$$

Just as dRL's box and diamond refinement rules $[\leq]$,$\langle\leq\rangle$ reduce a system property to a refinement property (second premise), the converse reduction is possible in dRL as well. The sequential composition refinement rule (;) reduces a refinement of a sequential composition (conclusion) to a refinement of the first

program (left premise) and a property of the first concrete system α_1 (right premise) which in turn refers to a postcondition that is a refinement ($\beta_1 \leq \beta_2$):

$$(;)\ \frac{P \to \alpha_1 \leq \alpha_2 \quad P \to [\alpha_1](\beta_1 \leq \beta_2)}{P \to (\alpha_1;\beta_1) \leq (\alpha_2;\beta_2)}$$

The (;) rule of dRL is particularly clever, exploiting the fact that dRL is a proper logic closed under all operators. Unlike the following easier (derived) version

$$(;)_s\ \frac{P \to \alpha_1 \leq \alpha_2 \quad \beta_1 \leq \beta_2}{P \to (\alpha_1;\beta_1) \leq (\alpha_2;\beta_2)}$$

rule (;) maintains more knowledge (such as assumption P and the effects of hybrid system α_1) than the simple structural refinement rule $(;)_s$ which loses all information (even just assuming P would be unsound in its second premise, because P may have become false after α_i). Since the simple rule $(;)_s$ has to discard all assumptions, it rarely applies, because hybrid systems often only refine each other given the contextual information of what happened previously and what was assumed initially, which is explicitly available in the second premise of the composition refinement rule (;). Again, the logic behind the sequential composition refinement rule (;) can be summarized more succinctly with the axiom that it derives from, which expresses that two sequential compositions are in refinement if the respective heads of the sequential compositions are in refinement and, always after the refining head, the respective tails are in refinement:

$$(;)\ \alpha_1;\alpha_2 \leq \beta_1;\beta_2 \leftarrow \alpha_1 \leq \beta_1 \wedge [\alpha_1](\alpha_2 \leq \beta_2)$$

The same idea is behind the unlooping axiom that reduces the refinement of two loops to a refinement of the loop bodies always after the refining loop:

$$(\text{unloop})\ \alpha^* \leq \beta^* \leftarrow [\alpha^*](\alpha \leq \beta)$$

There are further dRL axioms for proving refinements of loops, though, by a fixpoint-like argument when the loop is left or right of ;, respectively:

$$(\text{loop}_l)\ \alpha^*;\beta \leq \beta \leftarrow [\alpha^*](\alpha;\beta \leq \beta) \qquad (\text{loop}_r)\ \alpha;\beta^* \leq \alpha \leftarrow \alpha;\beta \leq \alpha$$

Observe how the $[\alpha^*]$ modality is only needed when the fixpoint starts at the end of the loop as in (loop_l), not when it starts in the beginning as in (loop_r). The elegant compositional (unloop) axiom and dL's complete axiomatization for equational properties of differential equations [46] are exploited to obtain decidability for refinements of a nontrivial fragment of hybrid programs [56]. The key axiom about differential equation refinements is the following equivalence:

$$(\text{ODE})\ x' = e\,\&\,P \leq x' = k\,\&\,Q \leftrightarrow [x' = e\,\&\,P](x' = k \wedge Q)$$

It expresses that, for arbitrary terms e and k, differential equation $x' = e\,\&\,P$ refines differential equation $x' = k\,\&\,Q$ iff, always after the former differential equation, the latter differential equation and evolution domain constraint holds. Notice how the postcondition $x' = k \wedge Q$ is a conjunction of an *equation of differentials* and the formula constituting the *evolution domain constraint* [23], hence, for suitable Q, decidable by a logical equivalence transformation (Theorem 3).

2.3. Differential Game Logic

dGL generalizes dL to provide modalities referring to the existence of winning strategies for hybrid games [19, 30, 31]. Hybrid games describe games played on hybrid systems. The essential difference is that hybrid systems merely feature *nondeterministic* resolution of choices (such as at $\cup$) while hybrid games feature *adversarial* resolution of choices with competing intent or effect. Imagine two players, called Angel and Demon, are each in charge of some of the decisions during the run of a hybrid system. Then, rather than asking whether all or some runs of the hybrid system satisfy a property, it makes sense to, instead, ask which of the players has a winning strategy to ensure their desired property when playing the hybrid game, no matter what strategy the opponent follows. Games are useful to investigate whether one player has a winning strategy in a truly competitive situation such as when Angel and Demon each control soccer robots from competing RoboCup teams. It is also useful to model the game that a controlled system played by Angel plays against an unknown environment played by Demon. For the same reason, hybrid games can also be useful to model the game that two pilots play with each other who both want to avoid collisions but may, nevertheless, reach decisions that inadvertently interfere with one another, for example, if both pilots decide to try to prevent a collision by ascending above the respective other aircraft, which they cannot both achieve.

Besides properties of competitive hybrid games, dGL is useful for correctness properties of hybrid systems in which some but not all actions are under the system designer's control. This includes systems with uncertainty caused by actions of other agents or the environment that may interfere. Hybrid games reasoning in dGL is useful for systematically synthesizing safe hybrid system controllers [57] by solving how the control player resolves its choices to win against the environment. Hybrid systems synthesis is hybrid game solving [57, 58].

Hybrid games α of dGL have actions where each decision is resolved by one of the two players Angel and Demon, respectively. In dL and dRL, the modality $[\alpha]$ refers to all runs of hybrid system α and $\langle\alpha\rangle$ refers to some run of hybrid system α. Hybrid games α do not have runs like systems do, because the outcome of a game play depends on the *interplay* of decisions of both players during the game α, where Angel decides all of her choices while Demon decides all of his choices, both of which are resolved interactively during game play.

In dGL, the modality $[\alpha]$, thus, instead refers to the existence of winning strategies for Demon in hybrid game α. The dGL formula $[\alpha]\varphi$ expresses that there is a winning strategy for player Demon in the hybrid game α with which he can resolve Demon's decisions to reach any state in which formula φ is true, no matter what counterstrategy Angel plays to resolve her decisions during α. The dGL formula $\langle\alpha\rangle\varphi$ expresses that there is a winning strategy for player Angel in the hybrid game α with which she can resolve Angel's decisions to reach any state in which formula φ is true, no matter what counterstrategy Demon plays.

This conservatively extends dL since player Demon has no decisions in a hybrid *system* α where Angel resolves all nondeterminism, because the dGL

formula $[\alpha]\varphi$ then exactly means that Demon has a strategy to achieve φ in the game α where Demon has no say and only Angel gets to make any decisions, i.e., φ is true after all runs of α. Likewise, the dGL formula $\langle\alpha\rangle\varphi$ for a hybrid *system* α exactly means that Angel has a strategy to achieve φ in a game where Angel gets to make all decisions (so she always helps) and Demon can never interfere, i.e., φ is true after at least one run of α. Hybrid systems are the hybrid games where only a single player, Angel, gets to make any choices. The other extreme is dGL with differential hybrid games [31], which are hybrid games featuring differential games [59], i.e., differential equations whose behavior is subject to continuous-time decisions by the two players Angel and Demon. In other words, differential hybrid games even feature player interaction during their continuous dynamics, not just during their discrete, temporal, and adversarial dynamics.

Read as a dGL formula with hybrid game α, dGL formula

$$\psi \to [\alpha]\phi \tag{1*}$$

is valid iff from every state satisfying precondition ψ, Demon has a winning strategy in game α to achieve ϕ no matter what Angel plays. As a dGL formula,

$$\psi \to \langle\alpha\rangle\phi \tag{2*}$$

is valid iff from every state satisfying ψ, Angel has a winning strategy in game α to achieve ϕ. The interactive nature of game play in dGL gives (1) and (2) with hybrid games α in dGL more flexible interactions between the players than merely referring to all runs in dL formula (1), or to some run in dL formula (2).

Example 6 (Two gear car game). As a specific example consider this dGL formula, which is a variation of the dL car from Example 1:

$$b > 0 \land v^2 \leq 2b(m-x) \to [(a := A \cup a := -b)^{\mathsf{d}}; \{x' = v, v' = a\}]\, x \leq m$$

This dGL formula expresses that Demon has a winning strategy to ensure $x \leq m$ since, being in scope of a dual game $\cdot^{\mathsf{d}}$, Demon is in charge of deciding whether the left choice $a := A$ of $\cup$ is taken or the right choice $a := -b$. Strictly speaking, the assignments are also in a dual game context, but it does not matter who plays the assignment, because assignments leave nothing to be decided. Note how the above dGL formula needs less assumptions than the dL formula from Example 1 precisely because the player has more control over what happens. Demon controls the choice in $\cdot^{\mathsf{d}}$ but Angel controls the duration of the subsequent differential equation. If the dual operators are placed elsewhere, even less assumptions are needed for Demon to have a winning strategy:

$$x \leq m \to [(a := A \cup a := -b); \{x' = v, v' = a\}^{\mathsf{d}}]\, x \leq m$$

But in this case, the game trivializes, because, after Angel picks either acceleration, Demon is in control of the differential equation (since it is in the $\cdot^{\mathsf{d}}$ context), so he can simply choose to evolve for 0 time making the choice of a irrelevant. Since Demon already has a winning strategy, Angel cannot possibly also have a

winning strategy for the opposite goal $\neg x \leq m$, because otherwise playing both winning strategies against one another would simultaneously achieve $x \leq m$ and $\neg x \leq m$, which is impossible. That is, the following dGL formula is not valid:

$$x \leq m \rightarrow \langle (a := A \cup a := -b); \{x' = v, v' = a\}^{\mathsf{d}} \rangle \neg x \leq m$$

All these dGL formulas are provable in the dGL proof calculus with a proof similar to the one in Fig. 1 but suitably adapted to handle the presence of $\cdot^{\mathsf{d}}$ dualities. The last dGL formula, which is not valid, instead can be disproved in dGL.

Axiomatics. Somewhat ironically but unsurprisingly, many dGL axioms are the same as dL axioms. For example, axiom $\langle\cup\rangle$ expresses that Angel has a winning strategy in the game of choice $\alpha \cup \beta$ to achieve P iff she has a winning strategy in the subgame α to achieve P or if she has such a strategy in subgame β.

$$\langle\cup\rangle \quad \langle \alpha \cup \beta \rangle P \leftrightarrow \langle \alpha \rangle P \vee \langle \beta \rangle P$$

This axiom is the same as for hybrid systems even if it now refers to hybrid games α, β rather than just hybrid systems, because as long as the player making the choice is the one currently being asked for the existence of a winning strategy, there is no noticeable change in these simple cases even if the semantic justification of soundness is vastly different. With the $\langle\cdot\rangle$ modality, axiom $\langle\cup\rangle$ refers to the existence of Angel's winning strategy for the respective games, but Angel is the one deciding how to play her $\cup$ operator, explaining the disjunction.

The most important defining axiom of dGL, instead, is for the game duality operator α^{d} which swaps the roles of the two players Angel and Demon:

$$\langle^{d}\rangle \quad \langle \alpha^{\mathsf{d}} \rangle P \leftrightarrow \neg \langle \alpha \rangle \neg P$$

That is, Angel has a winning strategy to achieve P in the dual game α^{d} that flips the game around iff, when pretending to play for her opponent, it is not the case that she has a winning strategy in the opposite game α to achieve the opposite $\neg P$ of what she truly wants to achieve. Since the $[\cdot]$ axiom (which is the *determinacy axiom* in hybrid games saying that Demon has a winning strategy in α iff Angel has no winning strategy in α to achieve the opposite) still shows $[\alpha]P \leftrightarrow \neg\langle\alpha\rangle\neg P$ for dGL, the duality

$$\langle \alpha^{\mathsf{d}} \rangle P \leftrightarrow [\alpha] P \tag{8}$$

derives from axiom $\langle^{d}\rangle$, which implies that duality operators swap diamond modalities with box modalities and vice versa, giving rise to the dynamic interactivity of hybrid games. That is, dGL formula (8) makes explicit that the $\cdot^{\mathsf{d}}$ operator indicates where the game flips around from Angel's perspective to Demon's perspective. The easiest way to understand the added power of dGL compared to dL exactly uses the fact that dualities make modalities flip from box to diamond and back via (8). The dL modalities $[\alpha]$ and $\langle\alpha\rangle$ refer to all or some runs of α. Since dGL dualities α^{d} cause modalities to flip via (8), every part of a hybrid game may alternate between universal and existential resolution

of the remaining decisions in the subgame, leading to unbounded alternation, directly in line with the alternating roles of game play resolution by Angel and Demon, respectively [30]. Another way to understand the nesting of dualities in hybrid games is that they cause similar nestings of fixpoint alternations [60, 61].

dGL is a gentle, innocent generalization of dL, because the addition of the duality operator $\cdot^{\mathsf{d}}$ is the only syntactic change. However, games call for an entirely new reading of the modalities and a different style of semantics for the interactivity of game play that is absent from systems that either run or don't. This fundamental change causes new opportunities and new proving challenges.

In fact, some reasoning principles that are perfectly fine for hybrid systems are unacceptable for hybrid games, because game dualities may turn boxes into diamonds and vice versa. For example, the hybrid systems axiom $\langle\rangle\vee$ commuting diamonds with disjunctions is unsound for hybrid games

$$\langle\rangle\vee\ \langle\alpha\rangle(P\vee Q)\leftrightarrow\langle\alpha\rangle P\vee\langle\alpha\rangle Q$$

because using β^{d} for α would, by formula (8), flip its diamonds into boxes, but neither $[\beta]P$ nor $[\beta]Q$ nor their disjunction follows from $[\beta](P\vee Q)$. Indeed, the resulting formula $[\beta](P\vee Q)\leftrightarrow[\beta]P\vee[\beta]Q$ is not even valid for hybrid systems, because, even if all runs of β satisfy the disjunction $P\vee Q$, that still does not mean they would either all satisfy P or all satisfy Q.

But this duality principle from formula (8) does not explain why the following hybrid systems axiom for backwards iteration is unsound for hybrid games:

$$\overleftarrow{[^*]}\ [\alpha^*]P\leftrightarrow P\wedge[\alpha^*][\alpha]P$$

The reason why $\overleftarrow{[^*]}$ is unsound for hybrid games is that its right-hand side expects one round of advance notice when terminating a repetition α^*, which is a harder game challenge that has nothing to do with box versus diamond modalities.

Not just axioms but proof rules are affected by the presence of games as well. dL's Gödel generalization rule G concludes that any formula P with a proof also holds after all runs of hybrid program α, because if P is true everywhere then it's also true always after running hybrid program α.

$$\mathrm{G}\ \frac{P}{[\alpha]P}$$

But dL's rule G would be unsound for dGL. Even for trivially true postconditions such as $x^2\geq 0$, Demon may not have a winning strategy to achieve $x^2\geq 0$ in the hybrid game α in case Angel has a strategy to trick Demon into not having any more permitted moves in the hybrid game α before even ever making it to a final state. dGL still obeys the monotonicity rule saying that if Demon has a strategy in hybrid game α to achieve P, then if P implies Q (premise), Demon also has a strategy in the same game α to achieve Q:

$$\mathrm{M}[\cdot]\ \frac{P\to Q}{[\alpha]P\to[\alpha]Q}$$

dGL is sound and complete relative to dGL's differentially expressive logics.

Theorem 4 (Relative completeness of dGL [30])**.** *The* dGL *calculus is a* sound and complete axiomatization *of hybrid games relative to any differentially expressive logic L (for* dGL*), i.e., every provable* dGL *formula is valid and every valid* dGL *formula can be proved from L tautologies:*

$$\vDash \phi \quad \textit{iff} \quad \mathrm{Taut}_L \vdash \phi$$

The addition of differential games to differential game logic does not affect expressibility and yet still leads to natural differential game invariant proof principles [31] that prove properties of differential games without solving them. Analyzing differential games is notoriously challenging. Differential game invariants were the first verification technique for differential games with a correctness proof (discounting verification techniques whose correctness proofs were later shown incorrect).

3. KeYmaera X Theorem Prover for Hybrid Systems

The dL, dRL and dGL proof calculi are implemented in the KeYmaera X theorem prover[4] [25], enabling users to specify and verify their hybrid systems and hybrid games applications. KeYmaera X provides automatic, interactive, and semiautomatic proofs, proof search tactics and custom proofs [64], interfacing with real arithmetic decision procedures implemented in Mathematica or Z3.

Unlike its predecessor KeYmaera [62], KeYmaera X [25] is a microkernel prover with an exceedingly small trusted core of only a few thousand lines of code, which leads to several design advantages [66]. The biggest advantage of the microkernel design of KeYmaera X is that its uniform substitution proof calculus for dL [23] is simple and parsimonious to implement and also verified to be sound in both Isabelle/HOL and Coq [67]. The implementation of an axiom therefore consists solely of the single concrete dL formula that it represents, with all other instances generated by uniform substitution. This design isolates potential soundness mistakes in KeYmaera X to the specific source code implementation and the decision procedures it is calling for real arithmetic (which have provably sound implementations [68–70] even if they are not yet competitive with unverified implementations for high degree polynomials). The fact that dL's soundness proofs [23] have been proved formally [67] and that the remaining real arithmetic is decidable by an algorithm with a formal correctness proof [70] gives dL an exceedingly strong soundness foundation. With a chain of theorem provers, the resulting guarantees can also be carried forward to verified machine code [71]. Overall, dL provides increasingly strong chains of proofs to establish the correctness of dynamical systems.

[4]The name KeYmaera X comes from its predecessor KeYmaera [62] which was based on the KeY prover for Java [63]. KeYmaera sounds like the hybrid animal *Chimaera* from ancient Greek mythology. This naming tradition also explains the name of KeYmaera X's tactic language, Bellerophon [64], named after the hero who defeated the *Chimaera*, and why Pegasus [65], named after the winged horse that helped Bellerophon, is the name of the invariant generator for KeYmaera X.

4. Application Overview

Applications of dL include verified collision freedom in the Federal Aviation Administration's (FAA) Next-Generation Airborne Collision Avoidance System ACAS X [26], verified ground robot obstacle avoidance despite actuator disturbance and sensor uncertainty [27], and verified train separation of train controllers for the kinematic model of the Federal Railroad Administration (FRA) with roll and curvature resistance, track slope forces, and air pressure brake force propagation [28]. The technical challenges overcome in these case studies are different, beyond the high-level differences of flying [26], free ground motion [27], and railroad motion [28], respectively. The ACAS X verification dealt with the complexity of a complicated high-dimensional but spatially sparse MDP policy optimization with the uncertainty of human pilot reactions in vertical aircraft motion. The ACAS X verification reduced half a trillion cases to one page of dL and found 15 billion counterexamples in ACAS X. The ground robot verification dealt with static, passive and passive friendly safety with static and dynamic obstacles in unstructured environments of robot motion as well as the common challenge of sensor and actuator uncertainty. Identifying the safe distance that a ground robot needs to keep to avoid colliding with an obstacle now reduces to identifying the obstacle's parameter bounds. The train control verification handled the full physics of the FRA's kinematic model with the competing effect of gravity slowing down trains uphill and speeding them up downhill while faster motion generates more velocity from downhill motion yet also increases resistance, all while air pressure brakes slowly build up and propagate braking force along the train, needing dual speed bounds.

Other applications of dL include automotive [72–75], correct-by-construction driving [76], robotics [77–79] and robot manipulation [80], component-based [81], skill-based [82] and service-oriented decompositions [83], multi-choice reactive controllers [84], hybrid active objects [85, 86], PLC control transformations [87], Simulink model transformations [88–90], and safe AI for CPS [91–95]. Applications of dL beyond conventional mobile cyber-physical systems include verified controllers for plasma positioning [96], chemical reactions [17], SCUBA diving [97], and numerical software [98].

The logic dRL is useful for proving refinement relations of implementations to abstract verification models. Applications of dRL include general proofs establishing relations of easily verified event-triggered models to easily implemented time-triggered models [99]. Event-triggered models are easier to verify, because their correctness solely relies on immediately following the correct response for every relevant event, which are assumed to be detected perfectly and immediately. Time-triggered models instead are easy to implement, because all they do is read sensors and act within a certain period of time, but they are much harder to verify, because reaction delays may make the system miss a critical event, causing safety hazards and indicating poor event abstraction.

Applications of dGL include verified collision freedom despite intruder actions in the Next-Generation Airborne Collision Avoidance System ACAS X [100] as well as structured proof languages for hybrid systems and hybrid games

[101, 102], and systematic control envelope synthesis for hybrid systems [57]. Constructive versions of dGL [103] have applications in setting the foundation for correct monitors for cyber-physical system controllers [102], constructive crossovers of dGL and dRL provide refinements between hybrid games and systems proving that winning strategies reify as programs winning the games [58], and in the verification of intermittently powered embedded systems [104], where correct behavior needs to be maintained despite temporary power loss.

Yet other applications use the QdL extension of dL for distributed hybrid systems [32] to study distributed car controllers for an unbounded number of cars on a road [72], distributed aircraft controllers for an unbounded number of, e.g., unmanned aerial vehicles [105], surgical robot controls for an unbounded number of surgical operating boundaries [106], and mobile ground robot controllers for an unbounded number of static and moving dynamic obstacles [27].

5. Related Work

There are three primary approaches for analyzing hybrid systems properties [3, 7, 107–109] all of which are too wide to discuss here in full: reachability and model checking, abstract interpretation, and deductive proofs.

Reachability analysis and model checking [110] systematically compute the set of all reachable states from a sufficiently small initial region and focus on linear systems [111, 112], use support function representations [113], or Taylor models [114]. The advantage of reachability analysis and model checking is that their basis in numerical computation makes them fairly automatic, but challenges include the need to manually select parameters such as step sizes, time horizons, and support function templates to obtain results. That is why analyses typically have to focus on small compact initial regions and state spaces for fixed finite time horizons to obtain meaningful results. SpaceEx, for instance, also generates images illustrating the computed reachable sets [112], which can give users an intuition about the system.

Abstract interpretation has been proposed for hybrid systems focusing on fixed representations of overapproximations of reachable sets in which the actual control software is analyzed [115]. Abstract interpretation has been quite successful for discrete software programs and is often more automatic than model checking, but abstract interpretation tools are customized to one particular type of question that the abstract domain is tuned for. The core essential difference between model checking [116, 117] and abstract interpretation [118] is the presence of a widening operator in abstract interpreters that speeds up convergence at the expense of losing some precision.

Deduction for hybrid systems [18, 19, 24, 119, 120] is rooted in logic and systematically constructs a symbolic proof of correctness. The advantage is that symbolic constructs are easier to get sound than numerical computations and that the proof is a direct witness of truth. Proofs for hybrid systems can also work compositionally stemming from the compositional semantics of logics for hybrid systems [18–24, 121]. The downside is that proofs for applications that exceed the present proof automation capabilities need to be guided by

manual insights, but the resulting advantage is that they *can* be guided by human insights, because proofs are transparent and explain the reasoning. Solid progress in proof automation often subsequently automates more generally [46, 52, 54–57, 65, 81, 122, 123] what previously still needed human guidance.

Among the deductive hybrid systems proof approaches, and certainly within the dL family, differential dynamic logic may have received the most significant attention. This makes it harder to compare it on a fair basis to other approaches. dL features an exceedingly rich meta-theory of soundness and completeness [20, 22, 23, 46]. dL and its siblings have been implemented in a series of three very different theorem provers KeYmaera [62], KeYmaeraD [124] and KeYmaera X [25], which helped tease out the right way of doing sound hybrid systems deduction. A technical comparison between the provers is reported in the literature [66]. Completeness on the theoretical side and usable tools on the practical side may help explain why dL has been used in so many case studies.

dL has also been implemented both as a deep embedding in Isabelle/HOL as well as Coq [67] and as a shallow embedding in Isabelle/UTP [125, 126]. The deep embedding is needed to establish formal meta theorems about dL such as soundness and is needed to obtain a provably sound prover microkernel. The shallow embedding, instead, establishes a notational embedding of dL, thereby reusing Isabelle Isar proof styles and Sledgehammer without hybrid systems reasoning. A temporal logic extension of dL [42] is formalized in PVS [41].

Hybrid Hoare Logic (HHL) provers for Hybrid CSP with Duration Calculus have been implemented both in Isabelle/HOL [127] and in Python [128]. HHL proof rules are often more complicated, because the semantics is more deterministic, giving more intricate conditions for handovers between different parts of Hybrid CSP. Hybrid CSP supports generation of SystemC code [129], which is useful even if it is not formally verified like dL's VeriPhy pipeline to executable code [71]. The Chinese Train Control System has been verified in HHL [130], which is comparable to the European Train Control System verified in dL [45].

6. Conclusions and Research Outlook

Differential dynamic logic and its siblings provide a solid logical foundation for dynamical systems analysis and design. They come with an exceedingly rich theory, including completeness, practical theorem prover implementations with significant automatic proof search procedures, and have played an important role in applications, including leading to the discovery of 15 billion counterexamples in the Next-Generation Airborne Collision Avoidance System ACAS X.

Owing to the generality of dynamical systems, it is no surprise that the generality of dL and its siblings make them suitable for a wide range of applications. Since dL provides modalities for every hybrid dynamical system and a relatively complete axiomatization, dL can be used for proofs of any such dynamical system. At the same time, there are interesting practical challenges and insights waiting in every application. After all, every application has different control challenges and needs different control insights. Flying drones also requires dif-

ferent decisions than driving trains. But the same logic can be used to find out what these control decisions are, and then justify that they are chosen correctly.

Cyber-physical systems are everywhere as technical systems, and dynamical systems are everywhere as mathematical models. Both can be analyzed using dL and its siblings. Having said that, even if dL and its siblings are perfectly compositional [18, 19, 21, 36, 121] so that a proof of a big model reduces to a logical combination of proofs of its submodels, it is still a challenge to scale to large systems. Even if component techniques [81], systematic refinement refactorings [131] and design processes [132] help, applications, nevertheless, have a never-ending hunger for scale that is in need of additional structuring principles [101] and additional proof automation techniques [65]. While dL and its siblings identify all the right principles, it is still an interesting and impactful challenge to influence industrial practice by affecting and translating standard industrial notation. The trick is to do this in a way that minimizes overhead in both directions and maximizes the free preservation of information and invariants.

While dL shines particularly at establishing correctness of hybrid systems algorithms themselves, the correctness of lower-level implementations is no less important. Of course, low-level implementations are doomed to be wrong if the high-level control algorithms are incorrect. But low-level implementations may still have mistakes once the high-level control algorithms are correct. The dL line of work has three potential remedies, all of which deserve further study to increase practicality. One is the use of dRL with explicit proofs of refinement of verified abstract models to concrete controllers inheriting their safety guarantees [29, 99]. Another is the use of the dL-based shielding technique ModelPlex for provably correct monitor synthesis to carry safety guarantees about hybrid systems models over to cyber-physical system implementations [133], which also forms the basis of a verified pipeline from verified hybrid systems models to verified machine code [71]. ModelPlex is also the technical heart and soul of the dL-based shielding technique that enables the safe use of artificial intelligence techniques such as reinforcement learning [91, 134] and neural network control [95] in cyber-physical systems. Yet another are systematic relations in constructive dGL of verified models to monitors and controllers [58, 102].

Acknowledgment. This material is supported by the Alexander von Humboldt Foundation. I am much indebted to Katherine Kosaian, Jonathan Laurent, Noah Abou El Wafa, and Dominique Méry for their valuable feedback on a prior invited paper at ABZ 2023. I also thank the anonymous reviewers for their helpful feedback.

References

[1] A. Nerode, W. Kohn, Models for hybrid systems: Automata, topologies, controllability, observability, in: Grossman et al. [107], pp. 317–356.

[2] M. S. Branicky, Studies in hybrid systems: Modeling, analysis, and control, Ph.D. thesis, Dept. Elec. Eng. and Computer Sci., Massachusetts Inst. Technol., Cambridge, MA (1995).

[3] T. A. Henzinger, P. W. Kopke, A. Puri, P. Varaiya, What's decidable about hybrid automata?, J. Comput. Syst. Sci. 57 (1) (1998) 94–124. doi:10.1006/jcss.1998.1581.

[4] A. J. van der Schaft, H. Schumacher, An Introduction to Hybrid Dynamical Systems, Vol. 251 of Lecture Notes in Control and Information Sciences, Springer, 1999.

[5] D. Liberzon, Switching in Systems and Control, Systems and Control: Foundations and Applications, Birkhäuser, Boston, MA, 2003.

[6] E. Asarin, T. Dang, O. Maler, Verification and Synthesis of Hybrid Systems, Control Engineering, Birkhäuser, 2006.

[7] A. Nerode, Logic and control, in: S. B. Cooper, B. Löwe, A. Sorbi (Eds.), CiE, Vol. 4497 of LNCS, Springer, Berlin, 2007, pp. 585–597. doi:10.1007/978-3-540-73001-9_61.

[8] P. Tabuada, Verification and Control of Hybrid Systems: A Symbolic Approach, Springer, Berlin, 2009. doi:10.1007/978-1-4419-0224-5.

[9] J. Lunze, F. Lamnabhi-Lagarrigue (Eds.), Handbook of Hybrid Systems Control: Theory, Tools, Applications, Cambridge Univ. Press, Cambridge, 2009. doi:10.1017/CBO9780511807930.

[10] E. A. Lee, S. A. Seshia, Introduction to Embedded Systems — A Cyber-Physical Systems Approach, Lulu.com, 2013.

[11] R. Alur, Principles of Cyber-Physical Systems, MIT Press, Cambridge, 2015.

[12] P. Marwedel, Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things, 4th Edition, Springer, 2021. doi:10.1007/978-3-030-60910-8.

[13] S. Mitra, Verifying Cyber-Physical Systems: A Path to Safe Autonomy, MIT Press, 2021.

[14] A. Abate, A. Tiwari, S. Sastry, Box invariance in biologically-inspired dynamical systems, Automatica (2009).

[15] R. Grosu, G. Batt, F. H. Fenton, J. Glimm, C. L. Guernic, S. A. Smolka, E. Bartocci, From cardiac cells to genetic regulatory networks, in: Gopalakrishnan and Qadeer [135], pp. 396–411. doi:10.1007/978-3-642-22110-1_31.

[16] P. D. Christofides, N. H. El-Farra, Control of Nonlinear and Hybrid Process Systems, Lecture Notes in Control and Information Sciences: Designs for Uncertainty, Constraints and Time-Delays, Springer, 2005.

[17] R. Bohrer, Chemical case studies in KeYmaera X, in: J. F. Groote, M. Huisman (Eds.), Formal Methods for Industrial Critical Systems - 27th International Conference, FMICS 2022, Warsaw, Poland, September 14-15, 2022, Proceedings, Vol. 13487 of LNCS, Springer, 2022, pp. 103–120. doi:10.1007/978-3-031-15008-1_8.

[18] A. Platzer, Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics, Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14509-4.

[19] A. Platzer, Logical Foundations of Cyber-Physical Systems, Springer, Cham, 2018. doi:10.1007/978-3-319-63588-0.

[20] A. Platzer, Differential dynamic logic for hybrid systems., J. Autom. Reas. 41 (2) (2008) 143–189. doi:10.1007/s10817-008-9103-8.

[21] A. Platzer, Logics of dynamical systems, in: LICS [136], pp. 13–24. doi:10.1109/LICS.2012.13.

[22] A. Platzer, The complete proof theory of hybrid systems, in: LICS [136], pp. 541–550. doi:10.1109/LICS.2012.64.

[23] A. Platzer, A complete uniform substitution calculus for differential dynamic logic, J. Autom. Reas. 59 (2) (2017) 219–265. doi:10.1007/s10817-016-9385-1.

[24] A. Platzer, Differential dynamic logics: Automated theorem proving for hybrid systems, Ph.D. thesis, Department of Computing Science, University of Oldenburg (2008).
URL http://oops.uni-oldenburg.de/1403/

[25] N. Fulton, S. Mitsch, J.-D. Quesel, M. Völp, A. Platzer, KeYmaera X: An axiomatic tactical theorem prover for hybrid systems, in: A. Felty, A. Middeldorp (Eds.), CADE, Vol. 9195 of LNCS, Springer, Berlin, 2015, pp. 527–538. doi:10.1007/978-3-319-21401-6_36.

[26] J. Jeannin, K. Ghorbal, Y. Kouskoulas, A. Schmidt, R. Gardner, S. Mitsch, A. Platzer, A formally verified hybrid system for safe advisories in the next-generation airborne collision avoidance system, STTT 19 (6) (2017) 717–741. doi:10.1007/s10009-016-0434-1.

[27] S. Mitsch, K. Ghorbal, D. Vogelbacher, A. Platzer, Formal verification of obstacle avoidance and navigation of ground robots, I. J. Robotics Res. 36 (12) (2017) 1312–1340. doi:10.1177/0278364917733549.

[28] A. Kabra, S. Mitsch, A. Platzer, Verified train controllers for the Federal Railroad Administration train kinematics model: Balancing competing brake and track forces, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 41 (11) (2022) 4409–4420. doi:10.1109/TCAD.2022.3197690.

[29] S. M. Loos, A. Platzer, Differential refinement logic, in: M. Grohe, E. Koskinen, N. Shankar (Eds.), LICS, ACM, New York, 2016, pp. 505–514. doi:10.1145/2933575.2934555.

[30] A. Platzer, Differential game logic, ACM Trans. Comput. Log. 17 (1) (2015) 1:1–1:51. doi:10.1145/2817824.

[31] A. Platzer, Differential hybrid games, ACM Trans. Comput. Log. 18 (3) (2017) 19:1–19:44. doi:10.1145/3091123.

[32] A. Platzer, A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems, Log. Meth. Comput. Sci. 8 (4:17) (2012) 1–44, special issue for selected papers from CSL'10. doi:10.2168/LMCS-8(4:17)2012.

[33] A. Platzer, Stochastic differential dynamic logic for stochastic hybrid programs, in: N. Bjørner, V. Sofronie-Stokkermans (Eds.), CADE, Vol. 6803 of LNCS, Springer, Berlin, 2011, pp. 446–460. doi:10.1007/978-3-642-22438-6_34.

[34] M. Brieger, S. Mitsch, A. Platzer, Uniform substitution for dynamic logic with communicating hybrid programs, in: B. Pientka, C. Tinelli (Eds.), CADE, Vol. 14132 of LNCS, Springer, 2023, pp. 96–115. doi:10.1007/978-3-031-38499-8_6.

[35] B. Bohrer, A. Platzer, A hybrid, dynamic logic for hybrid-dynamic information flow, in: Dawar and Grädel [137], pp. 115–124. doi:10.1145/3209108.3209151.

[36] A. Platzer, Logic & proofs for cyber-physical systems, in: N. Olivetti, A. Tiwari (Eds.), IJCAR, Vol. 9706 of LNCS, Springer, Cham, 2016, pp. 15–21. doi:10.1007/978-3-319-40229-1_3.

[37] A. Platzer, Refinements of hybrid dynamical systems logic, in: U. Glässer, J. C. Campos, D. Méry, P. Palanque (Eds.), Rigorous State-Based Methods - 9th International Conference, ABZ 2023, Nancy, France, Proceedings, Vol. 14010 of LNCS, Springer, 2023, pp. 3–14. doi:10.1007/978-3-031-33163-3_1.

[38] V. R. Pratt, Semantical considerations on Floyd-Hoare logic, in: FOCS, IEEE, Los Alamitos, 1976, pp. 109–121. doi:10.1109/SFCS.1976.27.

[39] D. Harel, D. Kozen, J. Tiuryn, Dynamic Logic, MIT Press, Cambridge, 2000. doi:10.7551/mitpress/2516.001.0001.

[40] W. Ahrendt, B. Beckert, R. Bubel, R. Hähnle, P. H. Schmitt, M. Ulbrich (Eds.), Deductive Software Verification – The KeY Book, Vol. 10001 of LNCS, Springer, 2016. doi:10.1007/978-3-319-49812-6.

[41] L. White, L. Titolo, J. T. Slagel, C. A. Muñoz, A temporal differential dynamic logic formal embedding, in: A. Timany, D. Traytel, B. Pientka, S. Blazy (Eds.), Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2024, London, UK, January 15-16, 2024, ACM, 2024, pp. 162–176. doi:10.1145/3636501.3636943.

[42] J. Jeannin, A. Platzer, dTL^2: Differential temporal dynamic logic with nested temporalities for hybrid systems, in: S. Demri, D. Kapur, C. Weidenbach (Eds.), IJCAR, Vol. 8562 of LNCS, Springer, Berlin, 2014, pp. 292–306. doi:10.1007/978-3-319-08587-6_22.

[43] E. W. Dijkstra, Guarded commands, nondeterminacy and formal derivation of programs, Commun. ACM 18 (8) (1975) 453–457. doi:10.1145/360933.360975.

[44] Y. K. Tan, A. Platzer, Deductive stability proofs for ordinary differential equations, in: J. F. Groote, K. G. Larsen (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Proceedings, Part II, Vol. 12652 of LNCS, Springer, 2021, pp. 181–199. doi:10.1007/978-3-030-72013-1_10.

[45] A. Platzer, J.-D. Quesel, European Train Control System: A case study in formal verification, in: K. Breitman, A. Cavalcanti (Eds.), ICFEM, Vol. 5885 of LNCS, Springer, Berlin, 2009, pp. 246–265. doi:10.1007/978-3-642-10373-5_13.

[46] A. Platzer, Y. K. Tan, Differential equation invariance axiomatization, J. ACM 67 (1) (2020) 6:1–6:66. doi:10.1145/3380825.

[47] A. Tarski, A Decision Method for Elementary Algebra and Geometry, 2nd Edition, University of California Press, Berkeley, 1951. doi:10.1007/978-3-7091-9459-1_3.

[48] G. E. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decomposition, in: H. Barkhage (Ed.), Automata Theory and Formal Languages, Vol. 33 of LNCS, Springer, Berlin, 1975, pp. 134–183. doi:10.1007/3-540-07407-4_17.

[49] V. Weispfenning, Quantifier elimination for real algebra — the quadratic case and beyond, Appl. Algebra Eng. Commun. Comput. 8 (2) (1997) 85–101. doi:10.1007/s002000050055.

[50] D. Richardson, Some undecidable problems involving elementary functions of a real variable, J. Symb. Log. 33 (4) (1968) 514–520. doi:10.2307/2271358.

[51] W. Walter, Ordinary Differential Equations, Springer, Berlin, 1998. doi:10.1007/978-1-4612-0601-9.

[52] A. Platzer, Y. K. Tan, Differential equation axiomatization: The impressive power of differential ghosts, in: Dawar and Grädel [137], pp. 819–828. doi:10.1145/3209108.3209147.

[53] Y. K. Tan, A. Platzer, Switched systems as hybrid programs, in: R. M. Jungers, N. Ozay, A. Abate (Eds.), 7th IFAC Conference on Analysis and Design of Hybrid Systems, ADHS 2021, Brussels, Belgium, July 7-9, 2021, Vol. 54 of IFAC-PapersOnLine, Elsevier, 2021, pp. 247–252. doi:10.1016/j.ifacol.2021.08.506.

[54] Y. K. Tan, A. Platzer, An axiomatic approach to existence and liveness for differential equations, Formal Aspects Comput. 33 (4) (2021) 461–518. doi:10.1007/s00165-020-00525-0.

[55] Y. K. Tan, S. Mitsch, A. Platzer, Verifying switched system stability with logic, in: E. Bartocci, S. Putot (Eds.), Hybrid Systems: Computation and Control (part of CPS Week 2022), HSCC'22, ACM, 2022. doi:10.1145/3501710.3519541.

[56] E. Prebet, A. Platzer, Uniform substitution for differential refinement logic, in: C. Benzmüller, M. Heule, R. Schmidt (Eds.), IJCAR, Vol. 14740 of LNCS, Springer, 2024, pp. 196–215. doi:10.1007/978-3-031-63501-4_11.

[57] A. Kabra, J. Laurent, S. Mitsch, A. Platzer, CESAR: Control envelope synthesis via angelic refinements, in: B. Finkbeiner, L. Kovács (Eds.), TACAS, Vol. 14570 of LNCS, Springer, 2024, pp. 144–164. doi:10.1007/978-3-031-57246-3_9.

[58] B. Bohrer, A. Platzer, Refining constructive hybrid games, in: Z. M. Ariola (Ed.), 5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France, Vol. 167 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 14.1–14.19. doi:10.4230/LIPIcs.FSCD.2020.14.

[59] R. P. Isaacs, Differential Games, John Wiley, 1967.

[60] N. Abou El Wafa, A. Platzer, Complete game logic with sabotage, in: P. Sobocinski, U. D. Lago, J. Esparza (Eds.), LICS, ACM, New York, 2024, pp. 1:1–1:15. doi:10.1145/3661814.3662121.

[61] N. Abou El Wafa, A. Platzer, First-order game logic and modal mu-calculus, CoRR abs/2201.10012 (2022). arXiv:2201.10012.

[62] A. Platzer, J.-D. Quesel, KeYmaera: A hybrid theorem prover for hybrid systems., in: A. Armando, P. Baumgartner, G. Dowek (Eds.), IJCAR, Vol. 5195 of LNCS, Springer, Berlin, 2008, pp. 171–178. doi:10.1007/978-3-540-71070-7_15.

[63] W. Ahrendt, T. Baar, B. Beckert, R. Bubel, M. Giese, R. Hähnle, W. Menzel, W. Mostowski, A. Roth, S. Schlager, P. H. Schmitt, The KeY tool, Software and System Modeling 4 (1) (2005) 32–54. doi:10.1007/s10270-004-0058-x.

[64] N. Fulton, S. Mitsch, B. Bohrer, A. Platzer, Bellerophon: Tactical theorem proving for hybrid systems, in: M. Ayala-Rincón, C. A. Muñoz (Eds.), ITP, Vol. 10499 of LNCS, Springer, 2017, pp. 207–224. doi:10.1007/978-3-319-66107-0_14.

[65] A. Sogokon, S. Mitsch, Y. K. Tan, K. Cordwell, A. Platzer, Pegasus: Sound continuous invariant generation, Form. Methods Syst. Des. 58 (1) (2022) 5–41, special issue for selected papers from FM'19. doi:10.1007/s10703-020-00355-z.

[66] S. Mitsch, A. Platzer, A retrospective on developing hybrid systems provers in the KeYmaera family - A tale of three provers, in: W. Ahrendt, B. Beckert, R. Bubel, R. Hähnle, M. Ulbrich (Eds.), Deductive Software Verification: Future Perspectives - Reflections on the Occasion of 20 Years of KeY, Vol. 12345 of LNCS, Springer, 2020, pp. 21–64. doi:10.1007/978-3-030-64354-6_2.

[67] B. Bohrer, V. Rahli, I. Vukotic, M. Völp, A. Platzer, Formally verified differential dynamic logic, in: Y. Bertot, V. Vafeiadis (Eds.), Certified Programs and Proofs - 6th ACM SIGPLAN Conference, CPP 2017, Paris, France, January 16-17, 2017, ACM, New York, 2017, pp. 208–221. doi:10.1145/3018610.3018616.

[68] A. Platzer, J.-D. Quesel, P. Rümmer, Real world verification, in: R. A. Schmidt (Ed.), CADE, Vol. 5663 of LNCS, Springer, Berlin, 2009, pp. 485–501. doi:10.1007/978-3-642-02959-2_35.

[69] M. Scharager, K. Cordwell, S. Mitsch, A. Platzer, Verified quadratic virtual substitution for real arithmetic, in: M. Huisman, C. S. Pasareanu, N. Zhan (Eds.), FM, Vol. 13047 of LNCS, Springer, 2021, pp. 200–217. doi:10.1007/978-3-030-90870-6_11.

[70] K. Kosaian, Y. K. Tan, A. Platzer, A first complete algorithm for real quantifier elimination in Isabelle/HOL, in: B. Pientka, S. Zdancewic (Eds.), Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs, ACM, New York, 2023, p. 211–224. doi:10.1145/3573105.3575672.

[71] B. Bohrer, Y. K. Tan, S. Mitsch, M. O. Myreen, A. Platzer, VeriPhy: Verified controller executables from verified cyber-physical system models, in: D. Grossman (Ed.), Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018, ACM, 2018, pp. 617–630. doi:10.1145/3192366.3192406.

[72] S. M. Loos, A. Platzer, L. Nistor, Adaptive cruise control: Hybrid, distributed, and now formally verified, in: M. Butler, W. Schulte (Eds.), FM, Vol. 6664 of LNCS, Springer, Berlin, 2011, pp. 42–56. doi:10.1007/978-3-642-21437-0_6.

[73] A. Abhishek, H. Sood, J. Jeannin, Formal verification of braking while swerving in automobiles, in: A. D. Ames, S. A. Seshia, J. Deshmukh (Eds.), HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020, ACM, 2020, pp. 27:1–27:11. doi:10.1145/3365365.3382217.

[74] M. Strauss, S. Mitsch, Slow down, move over: A case study in formal verification, refinement, and testing of the responsibility-sensitive safety model for self-driving cars, in: V. Prevosto, C. Seceleanu (Eds.), Tests and Proofs - 17th International Conference, TAP 2023, Leicester, UK, July 18-19, 2023, Proceedings, Vol. 14066 of LNCS, Springer, 2023, pp. 149–167. doi:10.1007/978-3-031-38828-6_9.

[75] Y. Selvaraj, W. Ahrendt, M. Fabian, Formal development of safe automated driving using differential dynamic logic, IEEE Trans. Intell. Veh. 8 (1) (2023) 988–1000. doi:10.1109/TIV.2022.3204574.

[76] A. Kittelmann, T. Runge, T. Bordis, I. Schaefer, Runtime verification of correct-by-construction driving maneuvers, in: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification and Validation. Verification Principles - 11th International Symposium, ISoLA 2022, Rhodes, Greece, October 22-30, 2022, Proceedings, Part I, Vol. 13701 of LNCS, Springer, 2022, pp. 242–263. doi:10.1007/978-3-031-19849-6_15.

[77] R. R. da Silva, B. Wu, H. Lin, Formal design of robot integrated task and motion planning, in: 55th IEEE Conference on Decision and Control, CDC 2016, Las Vegas, NV, USA, December 12-14, 2016, IEEE, 2016, pp. 6589–6594. doi:10.1109/CDC.2016.7799283.

[78] A. Kopylov, S. Mitsch, A. Nogin, M. Warren, Formally verified safety net for waypoint navigation neural network controllers, in: M. Huisman, C. S. Pasareanu, N. Zhan (Eds.), Formal Methods - 24th International Symposium, FM 2021, Virtual Event, November 20-26, 2021, Proceedings, Vol. 13047 of LNCS, Springer, 2021, pp. 122–141. doi:10.1007/978-3-030-90870-6_7.

[79] Q. Lin, S. Mitsch, A. Platzer, J. M. Dolan, Safe and resilient practical waypoint-following for autonomous vehicles, IEEE Control. Syst. Lett. 6 (2022) 1574–1579. doi:10.1109/LCSYS.2021.3125717.

[80] A. Partovi, R. R. da Silva, H. Lin, Reactive integrated mission and motion planning for mobile robotic manipulators, in: 2018 Annual American

Control Conference, ACC 2018, Milwaukee, WI, USA, June 27-29, 2018, IEEE, 2018, pp. 3538–3543. doi:10.23919/ACC.2018.8431866.

[81] A. Müller, S. Mitsch, W. Retschitzegger, W. Schwinger, A. Platzer, Tactical contract composition for hybrid system component verification, STTT 20 (6) (2018) 615–643, special issue for selected papers from FASE'17. doi:10.1007/s10009-018-0502-9.

[82] A. Knüppel, I. Jatzkowski, M. Nolte, T. Thüm, T. Runge, I. Schaefer, Skill-based verification of cyber-physical systems, in: H. Wehrheim, J. Cabot (Eds.), Fundamental Approaches to Software Engineering - 23rd International Conference, FASE 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Vol. 12076 of LNCS, Springer, 2020, pp. 203–223. doi:10.1007/978-3-030-45234-6_10.

[83] T. Liebrenz, P. Herber, S. Glesner, Service-oriented decomposition and verification of hybrid system models using feature models and contracts, Sci. Comput. Program. 211 (2021) 102694. doi:10.1016/j.scico.2021.102694.

[84] S. Lunel, S. Mitsch, B. Boyer, J. Talpin, Parallel composition and modular verification of computer controlled systems in differential dynamic logic, in: M. H. ter Beek, A. McIver, J. N. Oliveira (Eds.), Formal Methods - The Next 30 Years - Third World Congress, FM 2019, Porto, Portugal, October 7-11, 2019, Proceedings, Vol. 11800 of LNCS, Springer, 2019, pp. 354–370. doi:10.1007/978-3-030-30942-8_22.

[85] E. Kamburjan, From post-conditions to post-region invariants: Deductive verification of hybrid objects, in: Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control, HSCC '21, Association for Computing Machinery, New York, NY, USA, 2021. doi:10.1145/3447928.3456633.

[86] E. Kamburjan, S. Mitsch, R. Hähnle, A hybrid programming language for formal modeling and verification of hybrid systems, Leibniz Trans. Embed. Syst. 8 (2) (2022) 04:1–04:34. doi:10.4230/LITES.8.2.4.

[87] L. Garcia, S. Mitsch, A. Platzer, HyPLC: Hybrid programmable logic controller program translation for verification, in: L. Bushnell, M. Pajic (Eds.), ICCPS, 2019, pp. 47–56. doi:10.1145/3302509.3311036.

[88] T. Liebrenz, P. Herber, S. Glesner, Deductive verification of hybrid control systems modeled in simulink with KeYmaera X, in: J. Sun, M. Sun (Eds.), Formal Methods and Software Engineering - 20th International Conference on Formal Engineering Methods, ICFEM 2018, Gold Coast, QLD, Australia, November 12-16, 2018, Proceedings, Vol. 11232 of LNCS, Springer, 2018, pp. 89–105. doi:10.1007/978-3-030-02450-5_6.

[89] P. Herber, T. Liebrenz, J. Adelt, Combining forces: How to formally verify informally defined embedded systems, in: M. Huisman, C. S. Pasareanu, N. Zhan (Eds.), Formal Methods - 24th International Symposium, FM 2021, Virtual Event, November 20-26, 2021, Proceedings, Vol. 13047 of LNCS, Springer, 2021, pp. 3–22. doi:10.1007/978-3-030-90870-6_1.

[90] P. Herber, J. Adelt, T. Liebrenz, Formal verification of intelligent cyber-physical systems with the interactive theorem prover KeYmaera X, in: S. Götz, L. Linsbauer, I. Schaefer, A. Wortmann (Eds.), Proceedings of the Software Engineering 2021 Satellite Events, Braunschweig/Virtual, Germany, February 22 - 26, 2021, Vol. 2814 of CEUR Workshop Proceedings, CEUR-WS.org, 2021.

[91] N. Fulton, A. Platzer, Safe reinforcement learning via formal methods: Toward safe control through proof and learning, in: S. A. McIlraith, K. Q. Weinberger (Eds.), AAAI, AAAI Press, 2018, pp. 6485–6492.

[92] N. Fulton, A. Platzer, Safe AI for CPS, in: IEEE International Test Conference, ITC 2018, Phoenix, AZ, USA, October 29 - Nov. 1, 2018, IEEE, 2018, pp. 1–7. doi:10.1109/TEST.2018.8624774.

[93] M. Qian, S. Mitsch, Reward shaping from hybrid systems models in reinforcement learning, in: K. Y. Rozier, S. Chaudhuri (Eds.), NASA Formal Methods - 15th International Symposium, NFM 2023, Houston, TX, USA, May 16-18, 2023, Proceedings, Vol. 13903 of LNCS, Springer, 2023, pp. 122–139. doi:10.1007/978-3-031-33170-1_8.

[94] J. Adelt, D. Brettschneider, P. Herber, Reusable contracts for safe integration of reinforcement learning in hybrid systems, in: A. Bouajjani, L. Holík, Z. Wu (Eds.), Automated Technology for Verification and Analysis - 20th International Symposium, ATVA 2022, Virtual Event, October 25-28, 2022, Proceedings, Vol. 13505 of LNCS, Springer, 2022, pp. 58–74. doi:10.1007/978-3-031-19992-9_4.

[95] S. Teuber, S. Mitsch, A. Platzer, Provably safe neural network controllers via differential dynamic logic, CoRR abs/2402.10998 (2024). arXiv:2402.10998.

[96] M. Wu, J. Rosenberg, N. Fulton, A formally verified plasma vertical position control algorithm, in: M. H. ter Beek, D. Nickovic (Eds.), Formal Methods for Industrial Critical Systems - 25th International Conference, FMICS 2020, Vienna, Austria, September 2-3, 2020, Proceedings, Vol. 12327 of LNCS, Springer, 2020, pp. 170–188. doi:10.1007/978-3-030-58298-2_7.

[97] V. Bajaj, K. Elmaaroufi, N. Fulton, A. Platzer, Verifiably safe SCUBA diving using commodity sensors: work-in-progress, in: Proceedings of the International Conference on Embedded Software Companion, New York,

NY, USA, October 13-18, 2019, ACM, 2019, p. 8. doi:10.1145/3349568.3351554.

[98] A. Altuntas, J. Baugh, Hybrid theorem proving as a lightweight method for verifying numerical software, in: Proceedings of the Second International Workshop on Software Correctness for HPC Applications, Correctness'18, IEEE, 2018, pp. 1–8.

[99] S. M. Loos, Differential refinement logic, Ph.D. thesis, Computer Science Department, School of Computer Science, Carnegie Mellon University (2016).

[100] R. Cleaveland, S. Mitsch, A. Platzer, Formally verified next-generation airborne collision avoidance games in ACAS X, ACM Trans. Embed. Comput. Syst. 22 (1) (2023) 1–30. doi:10.1145/3544970.

[101] B. Bohrer, A. Platzer, Structured proofs for adversarial cyber-physical systems, ACM Trans. Embed. Comput. Syst. 20 (5s) (2021) 93:1–93:26, special issue on EMSOFT 2021. doi:10.1145/3477024.

[102] R. Bohrer, Practical end-to-end verification of cyber-physical systems, Ph.D. thesis, Computer Science Department, School of Computer Science, Carnegie Mellon University (2021).

[103] B. Bohrer, A. Platzer, Constructive hybrid games, in: N. Peltier, V. Sofronie-Stokkermans (Eds.), IJCAR, Vol. 12166 of LNCS, Springer, 2020, pp. 454–473. doi:10.1007/978-3-030-51074-9_26.

[104] R. Bohrer, B. Islam, Cyber-physical verification of intermittently powered embedded systems, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 41 (11) (2022) 4361–4372. doi:10.1109/TCAD.2022.3197541.

[105] S. M. Loos, D. W. Renshaw, A. Platzer, Formal verification of distributed aircraft controllers, in: Belta and Ivancic [138], pp. 125–130. doi:10.1145/2461328.2461350.

[106] Y. Kouskoulas, D. W. Renshaw, A. Platzer, P. Kazanzides, Certifying the safe design of a virtual fixture control algorithm for a surgical robot, in: Belta and Ivancic [138], pp. 263–272. doi:10.1145/2461328.2461369.

[107] R. L. Grossman, A. Nerode, A. P. Ravn, H. Rischel (Eds.), Hybrid Systems, Vol. 736 of LNCS, Springer, Berlin, 1993.

[108] P. J. Antsaklis, W. Kohn, A. Nerode, S. Sastry (Eds.), Hybrid Systems II, Vol. 999 of LNCS, Springer, Berlin, 1995.

[109] R. Alur, Formal verification of hybrid systems, in: S. Chakraborty, A. Jerraya, S. K. Baruah, S. Fischmeister (Eds.), EMSOFT, ACM, New York, 2011, pp. 273–278. doi:10.1145/2038642.2038685.

[110] L. Doyen, G. Frehse, G. J. Pappas, A. Platzer, Verification of hybrid systems, in: E. M. Clarke, T. A. Henzinger, H. Veith, R. Bloem (Eds.), Handbook of Model Checking, Springer, 2018, pp. 1047–1110. doi:10.1007/978-3-319-10575-8_30.

[111] T. A. Henzinger, P.-H. Ho, H. Wong-Toi, HyTech: A model checker for hybrid systems., in: O. Grumberg (Ed.), CAV, Vol. 1254 of LNCS, Springer, 1997, pp. 460–463.

[112] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, O. Maler, SpaceEx: Scalable verification of hybrid systems, in: Gopalakrishnan and Qadeer [135], pp. 379–395. doi:10.1007/978-3-642-22110-1_30.

[113] C. L. Guernic, A. Girard, Reachability analysis of hybrid systems using support functions, in: Bouajjani and Maler [139], pp. 540–554.

[114] X. Chen, E. Ábrahám, S. Sankaranarayanan, Flow*: An analyzer for non-linear hybrid systems, in: N. Sharygina, H. Veith (Eds.), CAV, Vol. 8044 of LNCS, Springer, 2013, pp. 258–263. doi:10.1007/978-3-642-39799-8_18.

[115] O. Bouissou, E. Goubault, S. Putot, K. Tekkal, F. Védrine, HybridFluctuat: A static analyzer of numerical programs within a continuous environment, in: Bouajjani and Maler [139], pp. 620–626. doi:10.1007/978-3-642-02658-4_46.

[116] C. Baier, J.-P. Katoen, K. G. Larsen, Principles of Model Checking, MIT Press, 2008.

[117] E. M. Clarke, O. Grumberg, D. A. Peled, H. Veith, Model Checking, 2nd Edition, MIT Press, Cambridge, 2018.

[118] P. Cousot, Abstract interpretation, ACM Comput. Surv. 28 (2) (1996) 324–328.

[119] Z. Manna, H. Sipma, Deductive verification of hybrid systems using STeP, in: T. A. Henzinger, S. Sastry (Eds.), HSCC, Vol. 1386 of LNCS, Springer, 1998, pp. 305–318.

[120] J. M. Davoren, A. Nerode, Logics for hybrid systems, IEEE 88 (7) (2000) 985–1010. doi:10.1109/5.871305.

[121] A. Platzer, Logic and compositional verification of hybrid systems (invited tutorial), in: Gopalakrishnan and Qadeer [135], pp. 28–43. doi:10.1007/978-3-642-22110-1_4.

[122] A. Platzer, E. M. Clarke, Computing differential invariants of hybrid systems as fixedpoints, Form. Methods Syst. Des. 35 (1) (2009) 98–120, special issue for selected papers from CAV'08. doi:10.1007/s10703-009-0079-8.

[123] J. Gallicchio, Y. K. Tan, S. Mitsch, A. Platzer, Implicit definitions with differential equations for KeYmaera X - (system description), in: J. Blanchette, L. Kovács, D. Pattinson (Eds.), IJCAR, Vol. 13385 of LNCS, Springer, 2022, pp. 723–733. doi:10.1007/978-3-031-10769-6_42.

[124] D. W. Renshaw, S. M. Loos, A. Platzer, Distributed theorem proving for distributed hybrid systems, in: S. Qin, Z. Qiu (Eds.), ICFEM, Vol. 6991 of LNCS, Springer, 2011, pp. 356–371. doi:10.1007/978-3-642-24559-6_25.

[125] S. Foster, J. J. H. y Munive, G. Struth, Differential Hoare logics and refinement calculi for hybrid systems with Isabelle/HOL, in: U. Fahrenberg, P. Jipsen, M. Winter (Eds.), Relational and Algebraic Methods in Computer Science - 18th International Conference, RAMiCS 2020, Palaiseau, France, April 8-11, 2020, Proceedings, Vol. 12062 of LNCS, Springer, 2020, pp. 169–186. doi:10.1007/978-3-030-43520-2_11.

[126] S. Foster, J. J. H. y Munive, M. Gleirscher, G. Struth, Hybrid systems verification with Isabelle/HOL: Simpler syntax, better models, faster proofs, in: M. Huisman, C. S. Pasareanu, N. Zhan (Eds.), Formal Methods - 24th International Symposium, FM 2021, Virtual Event, November 20-26, 2021, Proceedings, Vol. 13047 of LNCS, Springer, 2021, pp. 367–386. doi:10.1007/978-3-030-90870-6_20.

[127] S. Wang, N. Zhan, L. Zou, An improved HHL prover: An interactive theorem prover for hybrid systems, in: M. J. Butler, S. Conchon, F. Zaïdi (Eds.), Formal Methods and Software Engineering - 17th International Conference on Formal Engineering Methods, ICFEM 2015, Paris, France, November 3-5, 2015, Proceedings, Vol. 9407 of LNCS, Springer, 2015, pp. 382–399. doi:10.1007/978-3-319-25423-4_25.

[128] H. Sheng, A. Bentkamp, B. Zhan, HHLPy: Practical verification of hybrid systems using Hoare logic, in: M. Chechik, J. Katoen, M. Leucker (Eds.), Formal Methods - 25th International Symposium, FM 2023, Lübeck, Germany, March 6-10, 2023, Proceedings, Vol. 14000 of LNCS, Springer, 2023, pp. 160–178. doi:10.1007/978-3-031-27481-7_11.

[129] G. Yan, L. Jiao, S. Wang, L. Wang, N. Zhan, Automatically generating SystemC code from HCSP formal models, ACM Trans. Softw. Eng. Methodol. 29 (1) (2020) 4:1–4:39. doi:10.1145/3360002.

[130] L. Zou, J. Lv, S. Wang, N. Zhan, T. Tang, L. Yuan, Y. Liu, Verifying chinese train control system under a combined scenario by theorem proving, in: E. Cohen, A. Rybalchenko (Eds.), VSTTE, LNCS, Springer, 2013.

[131] S. Mitsch, J.-D. Quesel, A. Platzer, Refactoring, refinement, and reasoning: A logical characterization for hybrid systems, in: C. B. Jones,

P. Pihlajasaari, J. Sun (Eds.), FM, Vol. 8442 of LNCS, Springer, 2014, pp. 481–496. doi:10.1007/978-3-319-06410-9_33.

[132] J.-D. Quesel, S. Mitsch, S. Loos, N. Aréchiga, A. Platzer, How to model and prove hybrid systems with KeYmaera: A tutorial on safety, STTT 18 (1) (2016) 67–91. doi:10.1007/s10009-015-0367-0.

[133] S. Mitsch, A. Platzer, ModelPlex: Verified runtime validation of verified cyber-physical system models, Form. Methods Syst. Des. 49 (1-2) (2016) 33–74, special issue of selected papers from RV'14. doi:10.1007/s10703-016-0241-z.

[134] N. Fulton, A. Platzer, Verifiably safe off-model reinforcement learning, in: T. Vojnar, L. Zhang (Eds.), TACAS, Part I, Vol. 11427 of LNCS, Springer, 2019, pp. 413–430. doi:10.1007/978-3-030-17462-0_28.

[135] G. Gopalakrishnan, S. Qadeer (Eds.), Computer Aided Verification, CAV 2011, Snowbird, UT, USA, Proceedings, Vol. 6806 of LNCS, Springer, 2011.

[136] Logic in Computer Science (LICS), 2012 27th Annual IEEE Symposium on, IEEE, Los Alamitos, 2012.

[137] A. Dawar, E. Grädel (Eds.), Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, ACM, New York, 2018.

[138] C. Belta, F. Ivancic (Eds.), Hybrid Systems: Computation and Control (part of CPS Week 2013), HSCC'13, Philadelphia, PA, USA, April 8-13, 2013, ACM, New York, 2013.

[139] A. Bouajjani, O. Maler (Eds.), Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings, Vol. 5643 of LNCS, Springer, 2009.