Refinements of Hybrid Dynamical Systems Logic*

André Platzer[1]

Karlsruhe Institute of Technology, Karlsruhe, Germany platzer@kit.edu

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 of their safety-critical controls 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 are safe ($[\alpha]\varphi$), or that there is a run of the hybrid dynamical system ultimately reaching the desired goal ($\langle\alpha\rangle\varphi$). 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 to their abstract verification models. 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. Differential game logic (dGL) adds the ability of referring to winning strategies of players in hybrid games, which is useful for establishing correctness properties of systems where the actions of different agents may interfere. dL and its variations have been used in KeYmaera X for verifying ground robot obstacle avoidance, the Next-Generation Airborne Collision Avoidance System ACAS X, and the kinematics of train control in the Federal Railroad Administration model with track terrain influence and air pressure brake propagation.

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 [3, 4, 13, 20, 24, 25, 29, 34, 35, 53, 55]. The most canonical applications are those where the discrete dynamics of stepwise computation comes from computer controllers while the continuous dynamics following continuous functions

* This material is supported by the Alexander von Humboldt Foundation.

U. Glässer et al. (Eds.): ABZ 2023, LNCS 14010, pp. 3–14, 2023.
DOI: 10.1007/978-3-031-33163-3_1

comes from physical motion, as, e.g., in cars, aircraft, trains, and robots. Other applications of hybrid systems include biological systems [1, 19] and chemical processes [12, 14]. Many of these applications are safety-critical, which explains why a great deal of attention has been paid to the development of techniques that help either find mistakes in controllers or verify that there are no mistakes by establishing that the controllers are guaranteed to satisfy the desired correctness properties in the hybrid dynamical systems model [4, 20, 30, 38, 47, 55]. The fact that dealing with the real world is always difficult explains why verification of hybrid dynamical systems is challenging. However, the benefits of a more reliable system outweigh the verification cost whenever applications are important enough because mistakes incur significant financial loss or even risk loss of life.

This paper reports on the use of logic for hybrid dynamical systems. *Differential dynamic logic* (dL) [36–38, 41, 42, 45, 47] is a logic for specifying and verifying correctness properties of hybrid dynamical systems that is also implemented in the hybrid systems theorem prover KeYmaera X [18] that is available on the web[1] and has been used in interesting applications, including aircraft collision avoidance [21], ground robot obstacle avoidance [31], and railway control [22]. In fact, dL started its whole family of logics with several useful refinements and variations. *Differential refinement logic* (dRL) [28] adds refinement relations between hybrid systems as a first-class citizen logical operator. *Differential game logic* (dGL) [43, 46, 47]. The main purposes of all three of these logics will be sketched in this paper. Other extensions of dL are useful but beyond the scope of this paper, such as *hybrid-nominal differential dynamic logic* (dHL) whose nominals support hyper properties such as hybrid information flow [5], *quantified differential dynamic logic* (QdL) for distributed hybrid systems [40], and *stochastic differential dynamic logic* (SdL) for stochastic hybrid systems [39].

A technical survey of classical differential dynamic logic appeared at LICS'12 [42], a high-level survey of its principles at IJCAR'16 [44]. Information on the theory of dL can be found in a book [38]. A very readable comprehensive account of dL and dGL is provided in a textbook [47].

2 Differential Dynamic Logic Ideas

Differential Dynamic Logic. dL [36–38, 41, 42, 45, 47] provides a programming language for hybrid systems called *hybrid programs*, which functions 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 to describe continuous dynamics. Besides the operators of first-order logic of real arithmetic, dL provides modalities for hybrid programs α, where the dL formula $[\alpha]\varphi$ means that all final 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 is *valid* iff it is true in all states. Typical patterns for safety properties

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

are dL formulas of the form:

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

which are akin to Hoare triples except generalized to hybrid systems. 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 state where the postcondition formula ϕ holds. Stability properties nest more operators of dL. For example, *stability* of the origin for the differential equation $x' = f(x)$ is characterized by the dL formula [58]:

$$\forall\varepsilon{>}0\,\exists\delta{>}0\,\forall x\,(\mathcal{U}_\delta(x=0) \to [x' = f(x)]\mathcal{U}_\varepsilon(x=0)) \tag{3}$$

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 (3) 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 the ε-neighborhood of the origin. *Attractivity* of the origin for the differential equation $x' = f(x)$ is characterized by the dL formula [58]:

$$\exists\delta{>}0\,\forall x\,(\mathcal{U}_\delta(x=0) \to \forall\varepsilon{>}0\,\langle x' = f(x)\rangle[x' = f(x)]\mathcal{U}_\varepsilon(x=0)) \tag{4}$$

The dL formula (4) 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 (3) and (4) [58]. This illustrates how the fact that dL is a proper logic closed under all operators can be used to characterize many different properties of hybrid systems in a single logic. Other properties such as controllability and reactivity can be stated as well [49].

While it is crucial that dL has a simple and elegant unambiguous mathematical semantics [36–38,41,42,45,47] such that all dL formulas have a clear meaning, 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 [36–38, 41, 42, 45, 47]. For example, the dL calculus includes the axiom of nondeterministic choice:

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

Axiom [∪] 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 if and only 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. By using axiom [∪] to decompose its left-hand side $[\alpha \cup \beta]P$ to its corresponding right-hand side $[\alpha]P \wedge [\beta]P$, all

hybrid programs in the remaining verification question get simpler and smaller. Of course, dL's axioms for differential equations are fundamental to its success.

The dL proof calculus is a sound and complete axiomatization of hybrid systems relative to either discrete dynamics [41] or to continuous dynamics [36, 41]. For differential equation invariants, dL's axioms give a sound and complete axiomatization [51, 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. Similar soundness and completeness results hold for invariants of switched systems [59]. Liveness properties and existence properties of differential equations have corresponding proof principles derived in dL [57] and stability properties have proof principles derived in dL [56, 58] using Lyapunov functions.

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) [28] adds a refinement operator 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. 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 control model α. This view also gives rise to the box refinement rule, which proves that if precondition P is true, then all runs of the concrete system γ satisfy postcondition Q (conclusion below rule bar) by proving that the same implication for the abstract system α (left premise) and proving that the concrete system γ refines the abstract system α from all states satisfying the precondition P.

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

The box refinement rule $[\leq]$ reduces one box property (conclusion) to another $[\cdot]$ 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, e.g., when the abstract system α is more nondeterministic leaving out implementation detail that is important for performance of the actual implementation but irrelevant to safety. 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 can reach Q can the system γ reach Q, too:

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

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 (right premise) which in turn refers to a postcondition that is a refinement:

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

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 P and the effects of the actions of hybrid system α_1) than the simple structural refinement rule $(;)_s$ which loses all information (even just assuming P would be unsound in the second premise). Because 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 assumed initially, which is explicitly available in the second premise of the composition refinement rule (;).

Differential Game Logic. dGL generalizes dL to provide modalities referring to the existence of winning strategies for hybrid games [43,46,47]. Hybrid games α of dGL have actions where each decision is resolved by one of the two players called Angel and Demon, respectively. In dL and dRL, the modality $[\alpha]$ refers to all runs of hybrid system α. Hybrid games α do not have runs like systems do, because the outcome of a game play depends on the decisions of the 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 $[\alpha]$ refers to the existence of winning strategies for Demon in hybrid game α. More precisely, 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. 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 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. The most important defining axiom of dGL is for the duality operator α^{d} which swaps the roles of the two players Angel and Demon:

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

Since the $[\cdot]$ axiom (which is called the determinacy axiom in hybrid games) still derives $[\alpha]P \leftrightarrow \neg\langle\alpha\rangle\neg P$ for dGL, the duality

$$\langle\alpha^{\mathsf{d}}\rangle P \leftrightarrow [\alpha]P \tag{5}$$

derives, which implies that duality operators swap diamond modalities with box modalities and vice versa, giving rise to the dynamic interactivity of hybrid games. The easiest way to understand the added power of dGL uses the fact that dualities make modalities flip from box to diamond and back via (5). 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, every part of a hybrid game may alternate between universal and existential resolution of the remaining decisions in the subgame leading to unbounded alternation [43].

Read as a dGL formula with hybrid game α, dGL formula (1) is valid iff from every state where precondition ψ is true, Demon has a winning strategy in game α to achieve ϕ. As a dGL formula, (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 both (1) and (2) as dGL formulas with hybrid games α a significantly refined pattern of interaction between the players than merely referring to all runs as in dL formula (1), or to some run as in dL formula (2).

In some ways, dGL is a gentle and 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 logical modalities and a different style of semantics for the interactivity of game play that is absent from systems that either have a run or don't. This change causes new proving challenges. dL's Gödel generalization rule, G, for instance

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

concludes that any formula P with a proof also holds after all runs of hybrid program α. But this would be unsound for dGL, because even for trivial post-conditions such as $x^2 \geq 0$, is it not clear whether Demon has a winning strategy to achieve the obvious $x^2 \geq 0$ in the hybrid game α in case Angel has a winning strategy to trick Demon into violating the rules of the hybrid game α, so Demon never even successfully reaches a final state in which $x^2 \geq 0$ would then hold. 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:

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

Besides properties of competitive hybrid games, dGL is particularly useful to prove 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.

3 KeYmaera X Theorem Prover for Hybrid Systems

The dL and dGL proof calculi are implemented in the KeYmaera X theorem prover[2] [18], enabling users to specify and verify their hybrid systems and hybrid games applications. KeYmaera X provides automatic, interactive, and semiautomatic proofs, as well as proof search tactics and custom proofs [17], interfacing with real arithmetic decision procedures implemented in Mathematica or Z3.

Unlike its predecessor KeYmaera [48], KeYmaera X [18] is a microkernel prover with an exceedingly small trusted core, which leads to several design advantages [33]. The biggest advantage of the microkernel design of KeYmaera X is that its uniform substitution proof calculus for dL [45] is simple and parsimonious to implement and also verified to be sound in both Isabelle/HOL and Coq [9]. This design isolates potential soundness mistakes in KeYmaera X to the specific source code implementation or the decision procedures it is calling for real arithmetic (which have sound implementations [23, 50, 54] even if they are not yet always competitive with unverified implementations).

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 [21], verified ground robot obstacle avoidance in the presence of actuator disturbance and sensor uncertainty [31], 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 [22]. Applications of dL beyond conventional mobile cyber-physical systems include verified controllers for chemical reactions [12]. 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 [27]. Applications of dGL include verified collision freedom despite intruder actions in the Next-Generation Airborne Collision Avoidance System [15] as well as structured proof languages for hybrid systems and hybrid games [8, 11]. Constructive versions of dGL [6] also have important applications

[2] The KeYmaera X prover inherits its name from its predecessor KeYmaera [48] which was based on the KeY prover [2] and explains the spelling. KeYmaera is a homophone to *Chimaera*, the hybrid animal from ancient Greek mythology, which is a hybrid mixture of multiple animals just like KeYmaera is a prover mixing discrete and continuous mathematics and multiple theorem proving techniques.

in setting the foundation for monitors for cyber-physical system controllers [11], and constructive crossovers of dGL and dRL provide refinements between hybrid games and hybrid systems proving that winning strategies reify as programs winning the games [7].

5 Conclusions and Future Work

Differential dynamic logic and its siblings provide a solid logical foundation for cyber-physical systems analysis and design. They have also 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.

While differential dynamic logic itself 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 even 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 refinements to increase practicality. One is the the use of dRL with explicit proofs of refinement of verified abstract models to concrete controllers inheriting the safety guarantees [27, 28]. Another is the use of the dL-based ModelPlex technique for provably correct monitor synthesis to carry safety guarantees about hybrid systems models over to cyber-physical system implementations [32], which also forms the basis of a verified pipeline from verified hybrid systems models to verified machine code [10]. Yet another are systematic relations in constructive dGL of verified models to monitors and controllers [7,11].

Acknowledgment. I am much indebted to Katherine Kosaian, Jonathan Laurent, Noah Abou El Wafa, and Dominique Méry for their valuable feedback.

References

1. Abate, A., Tiwari, A., Sastry, S.: Box invariance in biologically-inspired dynamical systems. Automatica (2009)
2. Ahrendt, W., Baar, T., Beckert, B., Bubel, R., Giese, M., Hähnle, R., Menzel, W., Mostowski, W., Roth, A., Schlager, S., Schmitt, P.H.: The KeY tool. Software and System Modeling **4**(1), 32–54 (2005). doi: 10.1007/s10270-004-0058-x
3. Alur, R.: Principles of Cyber-Physical Systems. MIT Press, Cambridge (2015)
4. Asarin, E., Dang, T., Maler, O.: Verification and Synthesis of Hybrid Systems. Control Engineering, Birkhäuser (2006)
5. Bohrer, B., Platzer, A.: A hybrid, dynamic logic for hybrid-dynamic information flow. In: Dawar and Grädel [16], pp. 115–124. doi: 10.1145/3209108.3209151
6. Bohrer, B., Platzer, A.: Constructive hybrid games. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR. LNCS, vol. 12166, pp. 454–473. Springer (2020). doi: 10.1007/978-3-030-51074-9_26

7. Bohrer, B., Platzer, A.: Refining constructive hybrid games. In: Ariola, Z.M. (ed.) 5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France. LIPIcs, vol. 167, pp. 14.1–14.19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). doi: 10.4230/LIPIcs.FSCD.2020.14
8. Bohrer, B., Platzer, A.: Structured proofs for adversarial cyber-physical systems. ACM Trans. Embed. Comput. Syst. **20**(5s), 93:1–93:26 (2021). doi: 10.1145/3477024, special issue on EMSOFT 2021
9. Bohrer, B., Rahli, V., Vukotic, I., Völp, M., Platzer, A.: Formally verified differential dynamic logic. In: Bertot, Y., Vafeiadis, V. (eds.) Certified Programs and Proofs - 6th ACM SIGPLAN Conference, CPP 2017, Paris, France, January 16-17, 2017. pp. 208–221. ACM, New York (2017). doi: 10.1145/3018610.3018616
10. Bohrer, B., Tan, Y.K., Mitsch, S., Myreen, M.O., Platzer, A.: VeriPhy: Verified controller executables from verified cyber-physical system models. In: Grossman, D. (ed.) Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018. pp. 617–630. ACM (2018). doi: 10.1145/3192366.3192406
11. Bohrer, R.: Practical End-to-End Verification of Cyber-Physical Systems. Ph.D. thesis, Computer Science Department, School of Computer Science, Carnegie Mellon University (2021)
12. Bohrer, R.: Chemical case studies in KeYmaera X. In: Groote, J.F., Huisman, M. (eds.) Formal Methods for Industrial Critical Systems - 27th International Conference, FMICS 2022, Warsaw, Poland, September 14-15, 2022, Proceedings. LNCS, vol. 13487, pp. 103–120. Springer (2022). doi: 10.1007/978-3-031-15008-1_8
13. Branicky, M.S.: Studies in Hybrid Systems: Modeling, Analysis, and Control. Ph.D. thesis, Dept. Elec. Eng. and Computer Sci., Massachusetts Inst. Technol., Cambridge, MA (1995)
14. Christofides, P.D., El-Farra, N.H.: Control of Nonlinear and Hybrid Process Systems. Lecture Notes in Control and Information Sciences: Designs for Uncertainty, Constraints and Time-Delays, Springer (2005)
15. Cleaveland, R., Mitsch, S., Platzer, A.: Formally verified next-generation airborne collision avoidance games in ACAS X. ACM Trans. Embed. Comput. Syst. **22**(1), 1–30 (2023). doi: 10.1145/3544970
16. Dawar, A., Grädel, E. (eds.): Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science. ACM, New York (2018)
17. Fulton, N., Mitsch, S., Bohrer, B., Platzer, A.: Bellerophon: Tactical theorem proving for hybrid systems. In: Ayala-Rincón, M., Muñoz, C.A. (eds.) ITP. LNCS, vol. 10499, pp. 207–224. Springer (2017). doi: 10.1007/978-3-319-66107-0_14
18. Fulton, N., Mitsch, S., Quesel, J.D., Völp, M., Platzer, A.: KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In: Felty, A., Middeldorp, A. (eds.) CADE. LNCS, vol. 9195, pp. 527–538. Springer, Berlin (2015). doi: 10.1007/978-3-319-21401-6_36
19. Grosu, R., Batt, G., Fenton, F.H., Glimm, J., Guernic, C.L., Smolka, S.A., Bartocci, E.: From cardiac cells to genetic regulatory networks. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV. LNCS, vol. 6806, pp. 396–411. Springer, Berlin (2011). doi: 10.1007/978-3-642-22110-1_31
20. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? J. Comput. Syst. Sci. **57**(1), 94–124 (1998). doi: http://doi.org/10.1006/jcss.1998.1581

21. Jeannin, J., Ghorbal, K., Kouskoulas, Y., Schmidt, A., Gardner, R., Mitsch, S., Platzer, A.: A formally verified hybrid system for safe advisories in the next-generation airborne collision avoidance system. STTT **19**(6), 717–741 (2017). doi: 10.1007/s10009-016-0434-1
22. Kabra, A., Mitsch, S., Platzer, A.: Verified train controllers for the Federal Railroad Administration train kinematics model: Balancing competing brake and track forces. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **41**(11), 4409–4420 (2022). doi: 10.1109/TCAD.2022.3197690
23. Kosaian, K., Tan, Y.K., Platzer, A.: A first complete algorithm for real quantifier elimination in Isabelle/HOL. In: Pientka, B., Zdancewic, S. (eds.) Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs. p. 211224. ACM, New York (2023). doi: 10.1145/3573105.3575672
24. Lee, E.A., Seshia, S.A.: Introduction to Embedded Systems — A Cyber-Physical Systems Approach. Lulu.com (2013)
25. Liberzon, D.: Switching in Systems and Control. Systems and Control: Foundations and Applications, Birkhäuser, Boston, MA (2003)
26. Logic in Computer Science (LICS), 2012 27th Annual IEEE Symposium on. IEEE, Los Alamitos (2012)
27. Loos, S.M.: Differential Refinement Logic. Ph.D. thesis, Computer Science Department, School of Computer Science, Carnegie Mellon University (2016)
28. Loos, S.M., Platzer, A.: Differential refinement logic. In: Grohe, M., Koskinen, E., Shankar, N. (eds.) LICS. pp. 505–514. ACM, New York (2016). doi: 10.1145/2933575.2934555
29. Lunze, J., Lamnabhi-Lagarrigue, F. (eds.): Handbook of Hybrid Systems Control: Theory, Tools, Applications. Cambridge Univ. Press, Cambridge (2009). doi: 10.1017/CBO9780511807930
30. Mitra, S.: Verifying Cyber-Physical Systems: A Path to Safe Autonomy. MIT Press (2021)
31. Mitsch, S., Ghorbal, K., Vogelbacher, D., Platzer, A.: Formal verification of obstacle avoidance and navigation of ground robots. I. J. Robotics Res. **36**(12), 1312–1340 (2017). doi: 10.1177/0278364917733549
32. Mitsch, S., Platzer, A.: ModelPlex: Verified runtime validation of verified cyber-physical system models. Form. Methods Syst. Des. **49**(1-2), 33–74 (2016). doi: 10.1007/s10703-016-0241-z, special issue of selected papers from RV'14
33. Mitsch, S., Platzer, A.: A retrospective on developing hybrid systems provers in the KeYmaera family - A tale of three provers. In: Ahrendt, W., Beckert, B., Bubel, R., Hähnle, R., Ulbrich, M. (eds.) Deductive Software Verification: Future Perspectives - Reflections on the Occasion of 20 Years of KeY, LNCS, vol. 12345, pp. 21–64. Springer (2020). doi: 10.1007/978-3-030-64354-6_2
34. Nerode, A.: Logic and control. In: Cooper, S.B., Löwe, B., Sorbi, A. (eds.) CiE. LNCS, vol. 4497, pp. 585–597. Springer, Berlin (2007). doi: 10.1007/978-3-540-73001-9_61
35. Nerode, A., Kohn, W.: Models for hybrid systems: Automata, topologies, controllability, observability. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) Hybrid Systems. LNCS, vol. 736, pp. 317–356. Springer, Berlin (1992)
36. Platzer, A.: Differential dynamic logic for hybrid systems. J. Autom. Reas. **41**(2), 143–189 (2008). doi: 10.1007/s10817-008-9103-8
37. Platzer, A.: Differential Dynamic Logics: Automated Theorem Proving for Hybrid Systems. Ph.D. thesis, Department of Computing Science, University of Oldenburg (2008)

38. Platzer, A.: Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics. Springer, Heidelberg (2010). doi: 10.1007/978-3-642-14509-4
39. Platzer, A.: Stochastic differential dynamic logic for stochastic hybrid programs. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE. LNCS, vol. 6803, pp. 446–460. Springer, Berlin (2011). doi: 10.1007/978-3-642-22438-6_34
40. Platzer, A.: A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems. Log. Meth. Comput. Sci. **8**(4:17), 1–44 (2012). doi: 10.2168/LMCS-8(4:17)2012, special issue for selected papers from CSL'10
41. Platzer, A.: The complete proof theory of hybrid systems. In: LICS [26], pp. 541–550. doi: 10.1109/LICS.2012.64
42. Platzer, A.: Logics of dynamical systems. In: LICS [26], pp. 13–24. doi: 10.1109/LICS.2012.13
43. Platzer, A.: Differential game logic. ACM Trans. Comput. Log. **17**(1), 1:1–1:51 (2015). doi: 10.1145/2817824
44. Platzer, A.: Logic & proofs for cyber-physical systems. In: Olivetti, N., Tiwari, A. (eds.) IJCAR. LNCS, vol. 9706, pp. 15–21. Springer, Cham (2016). doi: 10.1007/978-3-319-40229-1_3
45. Platzer, A.: A complete uniform substitution calculus for differential dynamic logic. J. Autom. Reas. **59**(2), 219–265 (2017). doi: 10.1007/s10817-016-9385-1
46. Platzer, A.: Differential hybrid games. ACM Trans. Comput. Log. **18**(3), 19:1–19:44 (2017). doi: 10.1145/3091123
47. Platzer, A.: Logical Foundations of Cyber-Physical Systems. Springer, Cham (2018). doi: 10.1007/978-3-319-63588-0
48. Platzer, A., Quesel, J.D.: KeYmaera: A hybrid theorem prover for hybrid systems. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR. LNCS, vol. 5195, pp. 171–178. Springer, Berlin (2008). doi: 10.1007/978-3-540-71070-7_15
49. Platzer, A., Quesel, J.D.: European Train Control System: A case study in formal verification. In: Breitman, K., Cavalcanti, A. (eds.) ICFEM. LNCS, vol. 5885, pp. 246–265. Springer, Berlin (2009). doi: 10.1007/978-3-642-10373-5_13
50. Platzer, A., Quesel, J.D., Rümmer, P.: Real world verification. In: Schmidt, R.A. (ed.) CADE. LNCS, vol. 5663, pp. 485–501. Springer, Berlin (2009). doi: 10.1007/978-3-642-02959-2_35
51. Platzer, A., Tan, Y.K.: Differential equation axiomatization: The impressive power of differential ghosts. In: Dawar and Grädel [16], pp. 819–828. doi: 10.1145/3209108.3209147
52. Platzer, A., Tan, Y.K.: Differential equation invariance axiomatization. J. ACM **67**(1), 6:1–6:66 (2020). doi: 10.1145/3380825
53. van der Schaft, A.J., Schumacher, H.: An Introduction to Hybrid Dynamical Systems, Lecture Notes in Control and Information Sciences, vol. 251. Springer (1999)
54. Scharager, M., Cordwell, K., Mitsch, S., Platzer, A.: Verified quadratic virtual substitution for real arithmetic. In: Huisman, M., Pasareanu, C.S., Zhan, N. (eds.) FM. LNCS, vol. 13047, pp. 200–217. Springer (2021). doi: 10.1007/978-3-030-90870-6_11
55. Tabuada, P.: Verification and Control of Hybrid Systems: A Symbolic Approach. Springer, Berlin (2009). doi: 10.1007/978-1-4419-0224-5
56. Tan, Y.K., Mitsch, S., Platzer, A.: Verifying switched system stability with logic. In: Bartocci, E., Putot, S. (eds.) Hybrid Systems: Computation and Control (part of CPS Week 2022), HSCC'22. ACM (2022). doi: 10.1145/3501710.3519541
57. Tan, Y.K., Platzer, A.: An axiomatic approach to existence and liveness for differential equations. Formal Aspects Comput. **33**(4), 461–518 (2021). doi: 10.1007/s00165-020-00525-0

58. Tan, Y.K., Platzer, A.: Deductive stability proofs for ordinary differential equations. In: Groote, J.F., Larsen, K.G. (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. LNCS, vol. 12652, pp. 181–199. Springer (2021). doi: 10.1007/978-3-030-72013-1_10
59. Tan, Y.K., Platzer, A.: Switched systems as hybrid programs. In: Jungers, R.M., Ozay, N., Abate, A. (eds.) 7th IFAC Conference on Analysis and Design of Hybrid Systems, ADHS 2021, Brussels, Belgium, July 7-9, 2021. IFAC-PapersOnLine, vol. 54, pp. 247–252. Elsevier (2021). doi: 10.1016/j.ifacol.2021.08.506