Embedding Differential Temporal Dynamic Logic in PVS

Lauren White[1], Laura Titolo[2], and J. Tanner Slagel[1]

[1] NASA Langley Research Center, USA
{lauren.m.white,j.tanner.slagel}@nasa.gov
[2] National Institute of Aerospace
laura.titolo@nianet.org

1 Introduction

Differential dynamic logic (dL) [11, 12, 13, 14] is a formal framework to specify and reason about hybrid programs (HPs). The core of dL is a proof calculus that contains a collection of axioms and rules for the rigorous verification of properties of HPs. This calculus is implemented in the KeYmaera X[1] theorem prover which has been used for formal verification of several cyber-physical systems [4, 8, 6, 2, 1, 3, 7, 9]. Recently, dL has been embedded within the theorem prover Prototype Verification System (PVS) [10] resulting in the tool Plaidypvs[2]. The integration of dL into PVS expands its expressive power; user defined functions, such as trigonometric and other transcendental functions, can be used inside the dL framework, and meta-reasoning about HPs can be performed, including reasoning about entire classes of HPs, specified using dependent types in PVS.

One limitation of dL, KeYmaera X, and Plaidypvs is that they can only reason on the input/output semantics of an HP. Nevertheless, it is often the case that the correctness of an HP depends on the intermediate states that it can reach during its executions. For example, guaranteeing the position of an aircraft stays within a geofenced region. The differential temporal dynamic logic (dTL^2) has been introduced in [5] to extend dL with temporal logic operators and reason about all the states reachable during the execution of an HP.

This paper presents a work in progress focusing on embedding dTL^2 in PVS as an extension of Plaidypvs. Plaidypvs is expanded with the formalization of a trace semantics for HPs, the definition of the LTL temporal operators *eventually* and *globally*, and the implementation of the proof calculus for dTL^2. This new embedding has the same capabilities as Plaidypvs, which allows user defined functions and meta-reasoning of properties of HPs.

2 Embedding dTL^2 in PVS

HPs combine discrete programs and continuous evolutions and are defined by the syntax

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

where x and x' are variables, θ is a real expression, and P is a Boolean expression. The statement $x := \theta$ denotes a discrete assignment, $x' := \theta \,\&\, P$ models the continuous first order differential equation defined by θ that satisfies P, $?P$ tests if P is satisfied, and $x := * \,\&\, P$ assigns to x an arbitrary real value r such that $P(r)$ holds. HPs can be combined through sequential execution $(\alpha_1; \alpha_2)$, nondeterministic choice $(\alpha_1 \cup \alpha_2)$, and nondeterministic finite repetition (α_1^*).

[1] https://keymaerax.org
[2] Plaidypvs is part of the NASA PVS library available at https://github.com/nasa/pvslib/tree/master/dL.

Real and Boolean expressions are shallowly embedded in PVS. Given a state (or environment) in $\mathbb{S}$ which maps variables into $\mathbb{R}$, a real expression has type $[\mathbb{S} \to \mathbb{R}]$, while a Boolean expression has type $[\mathbb{S} \to \mathbb{B}]$ where $\mathbb{B}$ is the Boolean domain. This type of embedding is more general and easily extendable than the deep embedding where each operator must be defined with a dedicated datatype. Additionally, it allows interpretation of the operators directly in the logic of PVS, facilitating the task of writing the proofs.

The semantics of an HP is defined as a set of traces. In PVS, a trace is defined as a function $\sigma : \mathbb{N} \times \mathbb{R} \to \mathbb{S}$. A state $\sigma(i, r) \in \mathbb{S}$ denotes the state in trace σ occurring at the discrete step i and at time r. In the following, let σ_i denote $\sigma(i, 0)$ which is defined on the interval $[0, 0]$ and is used to model discrete steps. The trace semantics is defined as a relation $\tau(\alpha, t)$ that holds when a trace t belongs to the semantics of an HP α. For instance, it holds that $\tau(x := \theta, \sigma_0\sigma_1)$ when $\sigma_1 = \sigma_0[x/\theta]$ and $\tau(x' := \theta \,\&\, P, \sigma')$ where σ' is a state flow of order one solution of θ defined on $[0, r]$ such that for all $t \in [0, r]$, $\sigma(t)$ satisfies P.

There are two kinds of formulas in dTL^2: *state formulas* (ϕ) that are interpreted over a single state and trace formulas (π) that are interpreted over a trace:

$$\phi ::= \theta_1 \geq \theta_2 \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \forall x\ \phi \mid \exists x\ \phi \mid [\alpha]\pi \mid \langle\alpha\rangle\pi$$
$$\pi ::= \phi \mid \neg\pi \mid \Box\pi \mid \Diamond\pi$$

where θ_1 and θ_2 are real expressions. The run quantification statement $[\alpha]P$ asserts that every run of α ends by satisfying P. Similarly, $\langle\alpha\rangle P$ asserts that there exists a run of α where the final state satisfies P. The operators $\Box$, globally, and $\Diamond$, eventually, are defined in the typical way according to LTL with the restriction that they can be nested at most twice, i.e., the only combinations allowed are $\Box\Diamond$ and $\Diamond\Box$. While this can look like a stringent limitation, the combination of the LTL temporal operators with the dL run quantification allows reasoning on the reachable states of different computational paths.

Each proof rule of dTL^2 presented in [5] is specified as a lemma and proven correct in PVS. The non-temporal rules that reason on state formulas are inherited from Plaidypvs. For instance, the rules for proving a globally statement on all the runs of an assignment and a sequential composition are the following.

$$\frac{[x := \theta](\phi)}{[x := \theta](\Box\phi)} \qquad \frac{[\alpha_1][\alpha_2](\Box\phi)}{[\alpha_1; \alpha_2](\Box\phi)}$$

A lemma that encodes a desired rule can be instantiated in the PVS proof environment to prove a given property for an HP. In the future, proof strategies automating this process will be developed.

3 Conclusions

This extended abstract presents a work in progress for the implementation of the dTL^2 logic in PVS. Upon the completion of this work, the formalization of dTL^2 will be added to the Plaidypvs tool and made available in the NASA PVS library. The combination of LTL operators with dL allows for reasoning about the intermediate states on different computational paths of an HP, enabling an interesting fragment of CTL*. The dTL^2 embedding in PVS is implemented as a mixture of deep and shallow embeddings enabling user defined functions, and meta-reasoning about HPs using dependent types in PVS. To the best of the authors' knowledge this is the first implementation of dTL^2.

References

[1] B. Bohrer, Y.K. Tan, S. Mitsch, A. Sogokon, and A. Platzer. A formal safety net for waypoint following in ground robots. *IEEE Robotics and Automation Letters*, 4(3):2910–2917, 2019.

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

[3] N. Fulton and A. Platzer. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 6485–6492. AAAI Press, 2018.

[4] J. Jeannin, K. Ghorbal, Y. Kouskoulas, A. Schmidt, R. Gardner, S. Mitsch, and A. Platzer. A formally verified hybrid system for safe advisories in the next-generation airborne collision avoidance system. *International Journal on Software Tools for Technology Transfer*, 19(6):717–741, 2017.

[5] J. Jeannin and A. Platzer. dtl2: Differential temporal dynamic logic with nested temporalities for hybrid systems. In *Proceedings of the 7th International Joint Conference on Automated Reasoning (IJCAR 2014)*, volume 8562 of *LNCS*, pages 292–306. Springer, 2014.

[6] A. Kabra, S. Mitsch, and A. Platzer. Verified train controllers for the federal railroad administration train kinematics model: Balancing competing brake and track forces. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(11):4409–4420, 2022.

[7] S. Mitsch, M. Gario, C. J. Budnik, M. Golm, and A. Platzer. Formal verification of train control with air pressure brakes. In *Proceedings of the 2nd International Conference Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification (RSSRail 2017)*, volume 10598 of *LNCS*, pages 173–191. Springer, 2017.

[8] S. Mitsch, K. Ghorbal, D. Vogelbacher, and A. Platzer. Formal verification of obstacle avoidance and navigation of ground robots. *International Journal of Robotics Research*, 36(12):1312–1340, 2017.

[9] A. Müller, S. Mitsch, W. Retschitzegger, W. Schwinger, and A. Platzer. Change and delay contracts for hybrid system component verification. In *Proceedings of the 20th International Conference on Fundamental Approaches to Software Engineering (FASE 2017)*, volume 10202 of *LNCS*, pages 134–151. Springer, 2017.

[10] S. Owre, J. Rushby, and N. Shankar. PVS: A prototype verification system. In *Proceedings of the 11th International Conference on Automated Deduction (CADE 1992)*, pages 748–752. Springer, 1992.

[11] A. Platzer. Differential dynamic logic for verifying parametric hybrid systems. In *Proceedings of 16th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2007)*, volume 4548 of *LNCS*, pages 216–232. Springer, 2007.

[12] A. Platzer. Differential dynamic logic for hybrid systems. *Journal of Automated Reasoning*, 41(2):143–189, 2008.

[13] A. Platzer. A complete uniform substitution calculus for differential dynamic logic. *Journal of Automated Reasoning*, 59(2):219–265, 2017.

[14] A. Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, 2018.