Monitor Specification and Verified Runtime Monitoring of Component Models in Differential Dynamic Logic*

Stefan Mitsch

Computer Science Department
Carnegie Mellon University, Pittsburgh PA 15213, USA,
smitsch@cs.cmu.edu

Formal verification plays a crucial role in cyber-physical system (CPS) safety with strong guarantees about control choices under a certain model of physical behavior and interaction with the environment. Component-based modeling and verification techniques [1] address the challenges of modern CPSs—increasing system complexity, autonomy, and interacting heterogeneous agents—by splitting a traditionally monolithic verification efforts into more manageable pieces. Safety of the entire model is concluded from isolated safety arguments about its subsystems and their interaction. These guarantees translate to true runtime safety *if the models used for formal verification accurately reflect the true system behavior*, which can only be determined through runtime moitoring [2]. In CPS, runtime safety is challenging because it requires predictions about physical dynamics in order to act ahead of time, before physics makes violating the desired safety properties inevitable. Runtime monitoring provides system execution guarantees *if the monitors correctly predict and cover all conditions that need monitoring ahead of time to conclude safety*.

ModelPlex [2] *combines offline verification with verified runtime monitoring to simultaneously and verifiably resolve the complementary assumptions of formal verification and runtime monitoring*: ModelPlex checks that the models used for offline verification reflect reality with monitors that are derived by proof from the very models that we seek to validate. That way, the monitors inherit the verified predictive power and coverage from the model, and the model simultaneously inherits the validation power from the monitors. ModelPlex is implemented as a proof tactic in KeYmaera X [3]. This gives us the tools to create component models of the safety-relevant conditions of controllers and their physical effect, and manipulate them with provably correct techniques to synthesize monitoring conditions that are both executable and flexible enough to bridge the gap between abstract models and data obtained from sensors.

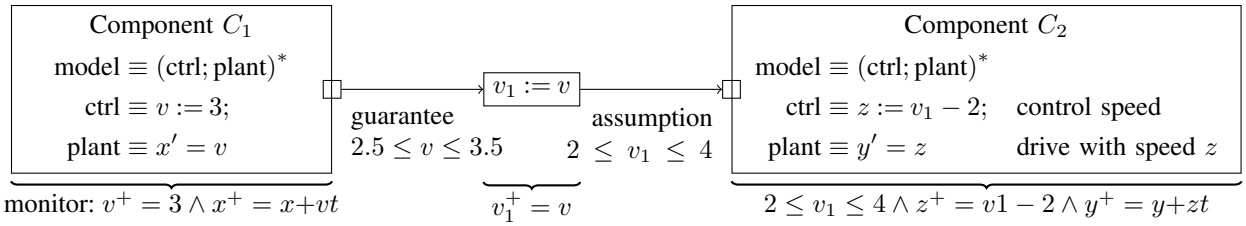
Monitor Specifications in Differential Dynamic Logic. The underlying idea of ModelPlex [2] is to simplify verification of the true system to verification of the current system run: the offline proof shows *safety of all runs of a model* and so the current true system run is safe *if it is one of the runs allowed in the model*. This means that the verified model itself is a specification for the predictions and conditions we need to runtime monitor. This idea can be expressed in differential dynamic logic (dL [4]) with relational properties about the input-output behavior observable on states ν_{i-1} and ν_i of a model $\alpha(x)$ as follows [2, Lemma 4]:

$$\underbrace{(\nu_{i-1},\nu_i) \in [\![\alpha(x)]\!]}_{\text{semantical condition}} \Leftrightarrow \underbrace{(\nu_{i-1},\nu_i) \models \langle\alpha(x)\rangle(x = x^+)}_{\text{Logical condition in dL}} \quad .$$

Semantical monitor condition $(\nu_{i-1},\nu_i) \in [\![\alpha]\!]$ is satisfied when the states ν_{i-1} and ν_i are connected through a run of the model $\alpha(x)$, which is equivalently expressed in dL as existence of a run ($\langle\cdot\rangle$) where the model $\alpha(x)$ is a hybrid systems model with differential equations that manipulates variables x of state ν_{i-1} such that their values become equal to the values x^+ of the followup state ν_i. In order to spare the computation overhead of proving the logical condition with concrete sensor measurements plugged into x and x^+ at runtime, ModelPlex performs an offline proof to retrieve arithmetic monitor conditions that can be checked based on sensor measurements. These arithmetic monitor conditions are left open in the offline proof, but when shown to be true at system runtime they provably witness compliance between the model and reality. In other words, evaluating the conditions with concrete sensor measurements closes the proof at runtime. The resulting monitor is, thus, accompanied by a correctness proof. Through ModelPlex, dL is a specification language for runtime monitor conditions with the full expressive power of discrete computations and differential equations without incurring the performance penalty of analyzing such specifications at runtime. Recent advances address partial model observability rooted in actuator disturbance and sensor uncertainty [5].

Monitoring of Verified Components. Component-based verification [1] describes an otherwise monolithic system with isolated components and additional communication programs describing the interaction between the components.

* This material is based upon work supported by AFOSR grant FA9550-16-1-0288 and by the United States Air Force and DARPA under Contract No. FA8750-18-C-0092. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.



The system safety proof follows from component contract compliance proofs with local contracts and additional communication and compatibility proofs for the interaction between the components [1]. In dL, contract compliance proofs and compatibility proofs are both phrased as safety properties over all runs of a component or all possible ways of interaction between components described in hybrid programs (e.g., contract compliance $A \rightarrow [\alpha]G$ is valid when under assumptions A all runs of the component α result in guarantees G). As a result, ModelPlex is applicable as a monitoring approach both for the components themselves and also for the interaction between the components.

Quantitative Monitoring Signals. CPSs become increasingly autonomous and so are likely to operate under conditions and in environments that are not fully anticipated at design time. Data-driven learning techniques promise to provide control components that generalize and are adaptable to situations not encountered at design time. Such an approach raises challenges related to detecting when a system operates outside its design regime, safely exploring unknown states, assessing how adaptations affect safety margins, and guiding a system from unsafe states back to safe operating conditions. These challenges ask for a quantitative monitor signal that not only separates the system states into safe and unsafe ones, but also measures the remaining safety margin and provides progress vectors leading from unsafe to safe states. The relational properties of ModelPlex monitor conditions compare system states on the basis of sensor measurements, which can be turned into quantitative safety margins similar to robustness measures for STL and MTL monitoring [7,8]. Such quantitative ModelPlex safety margins were recently used as a reward signal for safe reinforcement learning [9]. To unfold the full potential of quantitative monitoring signals we yet have to address several challenges: (i) How to create a uniform safety margin across different quantities (e.g., position vs. speed safety margin?) (ii) How to compare safety margins (e.g., sign-preserving multiplicative scaling retains safety semantics but destroys comparability)? (iii) How to detect performance deterioration (e.g., detect when online learning and adaptation reduces the safety margin over time)?

Summary. ModelPlex uses proofs to turn verified dL models (which enable monitor specifications with predictive models over discrete and continuous dynamics) into monitor conditions that, when satisfied from sensor measurements, provably imply true system safety at runtime. Compilation to verified machine code is available through the VeriPhy compilation pipeline [6]. ModelPlex has been used to produce runtime monitors of various case studies [2,5] and tested with experiments on real systems and in simulation. With increasing autonomy in CPSs, scaling to component models and support for learning with quantitative monitoring signals becomes a crucial aspect of monitoring.

References

1. Müller, A., Mitsch, S., Retschitzegger, W., Schwinger, W., Platzer, A.: Tactical contract composition for hybrid system component verification. STTT **20**(6) (2018) 615–643 Special issue for selected papers from FASE'17.
2. Mitsch, S., Platzer, A.: ModelPlex: Verified runtime validation of verified cyber-physical system models. Form. Methods Syst. Des. **49**(1) (2016) 33–74 Special issue of selected papers from RV'14.
3. Fulton, N., Mitsch, S., Quesel, J.D., Völp, M., Platzer, A.: KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In Felty, A.P., Middeldorp, A., eds.: CADE. Volume 9195 of LNCS., Springer (2015) 527–538
4. Platzer, A.: A complete uniform substitution calculus for differential dynamic logic. J. Autom. Reas. **59**(2) (2017) 219–265
5. Mitsch, S., Platzer, A.: Verified runtime validation for partially observable hybrid systems. CoRR **abs/1811.06502** (2018)
6. 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.: PLDI, ACM (2018) 617–630
7. Dokhanchi, A., Hoxha, B., Fainekos, G.E.: On-line monitoring for temporal logic robustness. In: Runtime Verification - 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings. (2014) 231–246
8. Deshmukh, J.V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., Seshia, S.A.: Robust online monitoring of signal temporal logic. Formal Methods in System Design **51**(1) (2017) 5–30
9. Fulton, N., Platzer, A.: Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In McIlraith, S., Weinberger, K., eds.: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, February 2-7, 2018, New Orleans, Louisiana, USA., AAAI Press (2018) 6485–6492