Towards Proving Runtime Properties of Data-Driven Systems Using Safety Envelopes

SAMUEL BREESE, FOTIS KOPSAFTOPOULOS
and CARLOS VARELA

ABSTRACT

Dynamic data-driven application systems [1, 2] (DDDAS) allow for unprecedented self-healing and self-diagnostic behavior across a broad swathe of domains. The usefulness of these systems is offset against their inherent complexity, and therefore fragility to specification or implementation error. Further, DDDAS techniques are often applied in safety-critical domains, where correctness is paramount. Formal methods facilitate the development of correctness proofs about software systems, which provide stronger behavioral guarantees than non-exhaustive unit tests. While unit testing can validate that a system behaves correctly in some finite number of configurations, formal methods enable us to prove correctness in an infinite subset of the configuration space, which is often needed in cyber-physical systems involving continuous mechanics. Although the efficacy of formal methods is traditionally offset by significantly greater development cost, we propose new development techniques that can mitigate this concern.

In this paper, we explore novel techniques for assuring the correctness of data-driven systems based on certified programming and software verification. In particular, we focus on the use of interactive theorem-proving systems to prove foundational properties about data-driven systems, possibly reliant upon physics-based assumptions and models. We introduce the concept of the *formal safety envelope*, analogous to the concept of an aircraft's performance envelope, which organizes system properties in a way that makes it clear which properties hold under which assumptions. Beyond maintaining modularity in proof development, this technique furthermore enables the derivation of runtime monitors to detect potentially unsafe system state changes, allowing the user to know precisely which properties have been verified to hold for the current system state. Using this method, we demonstrate the partial verification of an archetypal data-driven system from avionics, where wing sensor data is used to determine whether or not an airplane is likely to be in a stall state.

Samuel Breese, Carlos Varela, Department of Computer Science, Rensselaer Polytechnic Institute, 110 Eighth Street, Troy, NY 12180, USA

Fotis Kopsaftopoulos, Department of Mechanical, Aerospace, and Nuclear Engineering, Rensselaer Polytechnic Institute, 110 Eight Street, Troy, NY 12180, USA

in the Proceedings of the 12th International Workshop on Structural Health Monitoring
September 2017, Stanford, CA, USA

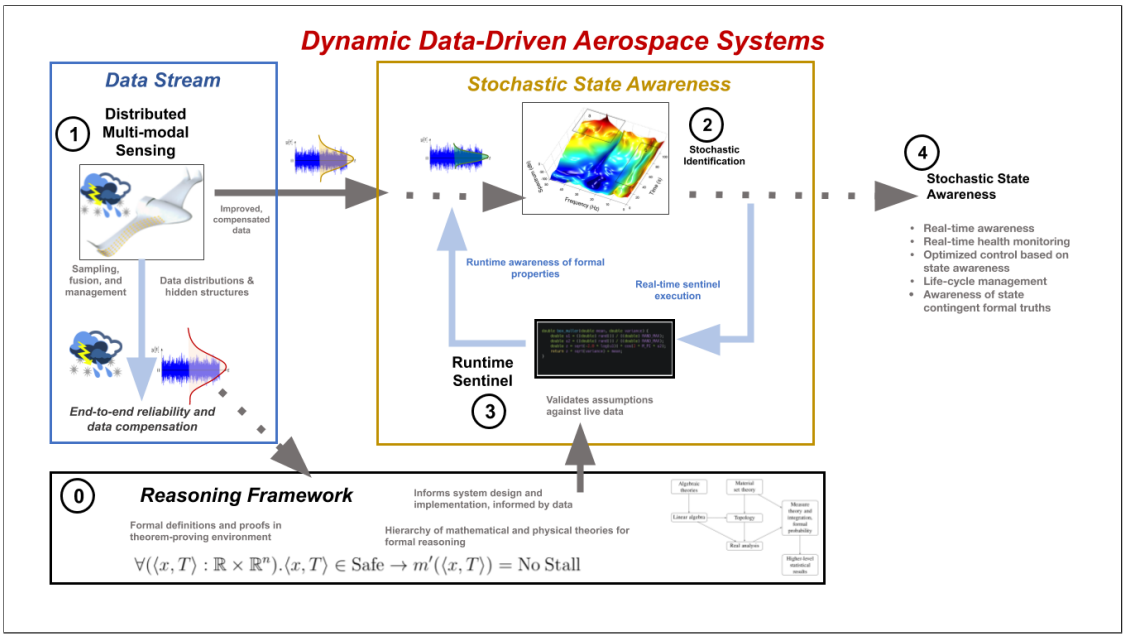

Figure 1. Integration of verification process into DDDAS development workflow

INTRODUCTION

Dynamic data-driven aerospace systems [3–6] involve the dynamic integration and unification of sensor data, models, and computation. These systems, which fall under the multidisciplinary umbrella of dynamic data-driven application systems (DDDAS), are characterized by their capability both to incorporate runtime data in an executing process and to allow that same executing process to influence the measurement and collection of runtime data. However, these dynamic data-driven behaviors introduce significant complexity when analyzing system correctness, often to the point that exhaustive system testing becomes infeasible due to the dynamic nature of the data reintegration process and the continuous nature of the runtime data itself. Techniques from formal methods, and in particular formal software verification techniques, can be used to alleviate the burden this complexity places upon system development and analysis. We have adapted techniques traditionally used in the analysis of discrete systems (*e.g.*, programming languages) and adapted them to continuous stochastic systems, enabling us to analyze the reliability of DDDAS in a precise and principled manner (Figure 1).

We propose verification of correctness properties for a stochastic system modulo precise *safety envelopes*. We formally verify some high-level correctness properties of an archetypal data-driven model for stall detection on an aircraft. In the process, we address challenges related to the correct treatment of unpredictable runtime data in a formal setting. We develop a proof framework that highlights and enables compositional reasoning, stressing the reusability of lower-level formal theories in the specification and proof of higher-level stochastic properties. This framework emphasizes the dynamic nature of the systems at play through the generation of *runtime sentinels* from envelope specifications, allowing the runtime system to respond intelligently to changes in the formal environment inhabited by the current system configuration and runtime state. This cul-

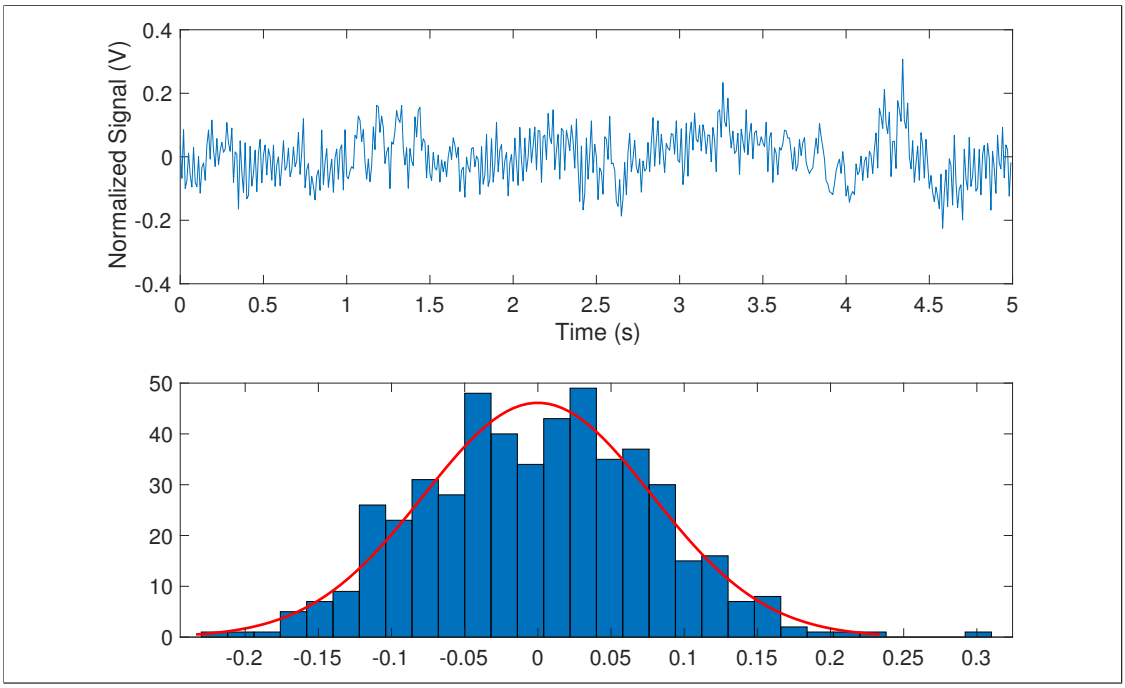

Figure 2. Visualization of experimental signal energy used to train model

minates in the continuing development of a hierarchy of algebraic and analytic theories enabling high-level reasoning about stochastic properties of data-driven systems.

Formal Methods

Typically, verification efforts focused on cyber-physical systems are primarily concerned with applying automated methods, for example, model checking. More recently, however, there has been interest in applying interactive / human-guided techniques. For example, see the VeriDrone project [7], which builds upon existing work using differential dynamic logic [8] to verify properties of hybrid systems [9] foundationally using an interactive theorem-proving system.

Interactive theorem-proving is the process of formally specifying properties, and then formally proving that these properties are correct within a *proof assistant*. One popular proof assistant is Coq [10], developed at INRIA, which we use in this work. Using a proof assistant is much like developing a proof of correctness on paper, except that the proof assistant ensures that proofs are sound, *i.e.*, that theorems are logical consequences of axioms. There are some barriers to using such tools: in particular, it requires the user to work at a level of specificity and detail that is not typical of even the most formal mathematical reasoning on paper. Further, the user must have access to a formalization of every definition and theorem they require, either by using a library (for example, Coquelicot [11] for real analysis) or by developing the required formalization "in-house".

Interactive methods are interesting because they can be used without sacrificing the undoubtedly-useful automated methods. Tools like SMT solvers can be used to automatically discharge simple propositions, while more complex propositions that would be intractable in a purely-automated approach can be tackled "manually". There is ex-

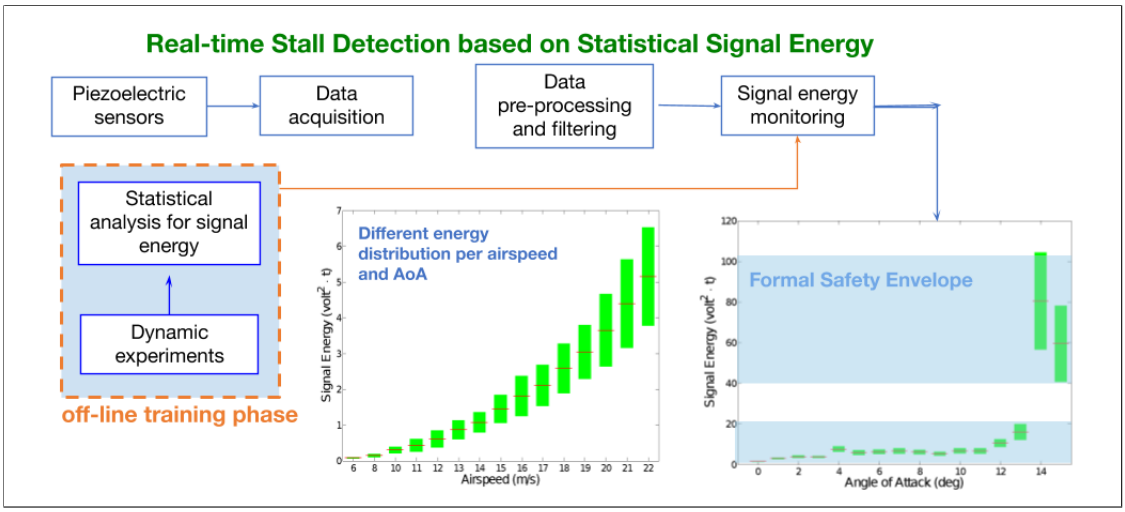

Figure 3. Safety envelope for data-driven model

isting work on integrating such tools with interactive theorem-provers [12–14], allowing us to use the theorem-prover for high-level organization and proofs of difficult lemmas, while leaving the mundane and well-understood work to automated tools. As such, we believe that a high-level interactive theorem-proving system like Coq is an essential tool in nearly any complex verification task by merit of forcing propositions to be organized in a principled way, even if most of the actual "proving" is done automatically.

A NONPARAMETRIC DATA-DRIVEN MODEL

We consider a model presented by Kopsaftopoulos [15, 16] that associates the signal energy from a piezoelectric sensor on a wing with the likelihood that an aircraft is in a stall state. The model is based upon experimental wind tunnel data, obtained by controlling the angle of attack and airspeed configuration and recording the mean and variance of signal energy. Figure 2 illustrates a visualization of the signal energy collected from one sensor on the wing (for a fixed airspeed of 15 m/s and for a fixed angle of attack of 7 degrees, with a sampling rate of 1000 Hz). Specific angle of attack / airspeed configurations correlate with aeroelastic properties, allowing certain signal energy distributions to be associated with stall/no stall conditions.

There are two distinct scenarios related to this model where we would like to verify correctness properties. First, given the assumption that the experimental signal energy data is normally distributed, we want to be sure that the model behaves "correctly". Additionally, there should be some interval (or union of intervals) of signal energies that the model classifies as unlikely to correspond to the stall state (assuming some reasonable significance level). It is important to distinguish these cases and to treat them appropriately. In the end, we can view the model as a function

$$m : \mathbb{R}^n \to (\mathbb{R} \to \{\text{Stall}, \text{No Stall}\})$$

where n is the size of the training data (in our model $n = 90000$). This function can be

uncurried to a function

$$m' : \mathbb{R}^n \times \mathbb{R} \to \{\text{Stall}, \text{No Stall}\},$$

which will allow us to treat pairs of training data and runtime signal energy as system states. Thus, we can formally define a safety envelope as a constraint on these training data and signal energy pairs, an important simplification that enhances the modularity of the verification process.

Formal Safety Envelope

A formal safety envelope, analogous to a flight safety envelope describing the safe operating conditions of an aircraft, is a computable subset of the DDDAS state space that describes the (ideally weakest possible) constraints on the operating conditions under which a correctness guarantee is valid. This subset can involve both continuous constraints (*e.g.* membership in some interval of the state space) and non-continuous constraints (*e.g.* Gaussian distribution of the sample history of a sensor). Figure 3 shows a depiction of the continuous constraint placed upon runtime signal energy. Alongside this continuous constraint is a statistical constraint on the distribution of signal energy sensor data at the time of training.

Correctness Results

Assuming both of these constraints hold for a given training data / runtime signal energy pair, we wish to prove that the model always correctly classifies that runtime signal energy. This proof relies on some extensional properties of the model function m'. Here, we treat m' as follows: There is a set of signal energy means and variances $D(T)$ taken from the experimental data T at various airspeeds and angles of attack. Certain such airspeed / angle of attack configurations correspond to stall (using physical properties, or even observationally), and therefore a certain subset of signal energy means and variances $S(T) \subseteq D(T)$ correspond to stall states. The model function m' tests the runtime signal energy against every mean and variance in that subset $S(T)$ assuming normality, and we are sure that it classifies that runtime signal energy as likely to correspond to stall if it is suitably likely (*e.g.*, 99%) to occur in any such distribution, while also being suitably unlikely (*e.g.* 1%) for all distributions in $D(T) \setminus S(T)$. Analogously, we are sure that it classifies that runtime signal energy as unlikely to correspond to stall if it is likely in any $D(T) \setminus S(T)$, while also unlikely in all $S(T)$.

From here, it is relatively easy to verify model correctness. Since we know that the experimental data is normally distributed, we know that each distribution in $D(T)$ is normal. We wish to prove that for all signal energies in a given interval, the model behaves in a predictable way. Formally, we might express this proposition for the "No Stall" classification as

$$\forall(\langle x, T\rangle : \mathbb{R} \times \mathbb{R}^n).\langle x, T\rangle \in \text{Safe} \to m'(\langle x, T\rangle) = \text{No Stall}$$

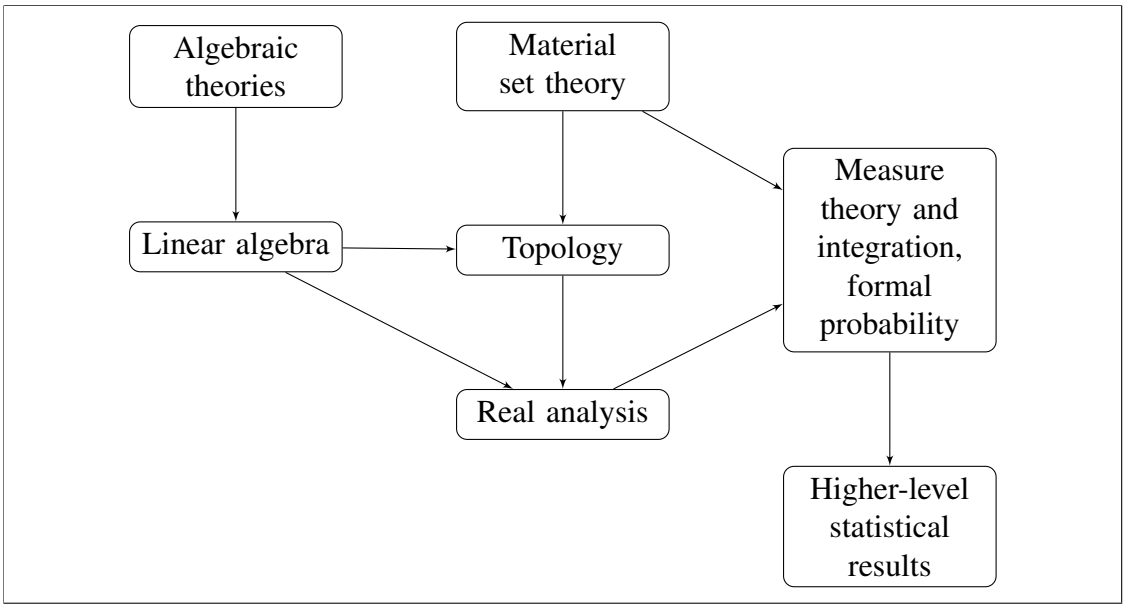

Figure 4. Hierarchical nature of theories for reasoning about high-level stochastic properties

where Safe, the safe subset, is all $\langle x, T\rangle$ satisfying

$$\begin{aligned}&(\forall(d \in D(T)).\mathrm{Gaussian}(d))\\ &\wedge (\forall(d \in S(T))\, |x - \mu(d)| > 3\sigma(d))\\ &\wedge (\exists(d \in D(T) \setminus S(T))\, |x - \mu(d)| < 3\sigma(d)),\end{aligned}$$

where $\mu(d)$ and $\sigma(d)$ are the mean and standard deviation of the distribution d, respectively. Analogously, the corresponding proposition for the "Stall" classification is similar except for an inversion of the roles of $S(T)$ and $D(T) \setminus S(T)$:

$$\forall(\langle x, T\rangle : \mathbb{R} \times \mathbb{R}^n).\langle x, T\rangle \in \mathrm{Safe}' \rightarrow m'(\langle x, T\rangle) = \mathrm{Stall}$$

where the new safe subset Safe' is all $\langle x, T\rangle$ satisfying

$$\begin{aligned}&(\forall(d \in D(T)).\mathrm{Gaussian}(d))\\ &\wedge (\forall(d \in D(T) \setminus S(T))\, |x - \mu(d)| > 3\sigma(d))\\ &\wedge (\exists(d \in S(T))\, |x - \mu(d)| < 3\sigma(d)).\end{aligned}$$

To prove these, we rely on the previously expressed properties of m': we can simply compute the CDF of each Gaussian distribution (known from the first assumption in the safe subsets) given minimum / maximum values for x ($\mu \pm 3\sigma$, from the second and third assumptions in the safe subsets). While this is simple on paper, involving just simple algebra and some mechanical computation, the nature of foundational verification makes it a somewhat daunting task in the formal setting of a proof assistant. In Figure 4, we see the hierarchy of mathematical theories that must be (in full or in part) formalized in order to allow high-level reasoning about statistics and properties of stochastic systems. This complexity hides beneath even the simplest statements related to *e.g.*, probability,

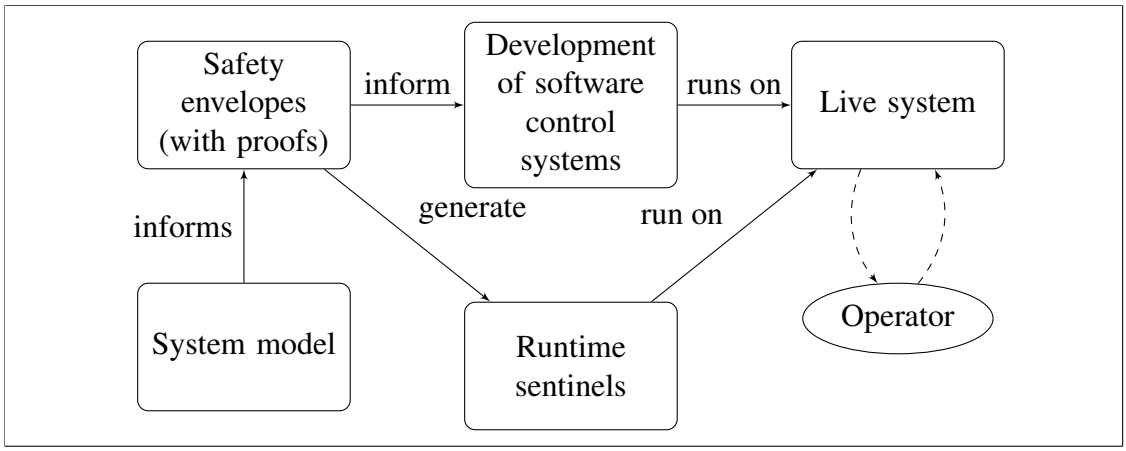

Figure 5. Workflow for verification with runtime sentinels

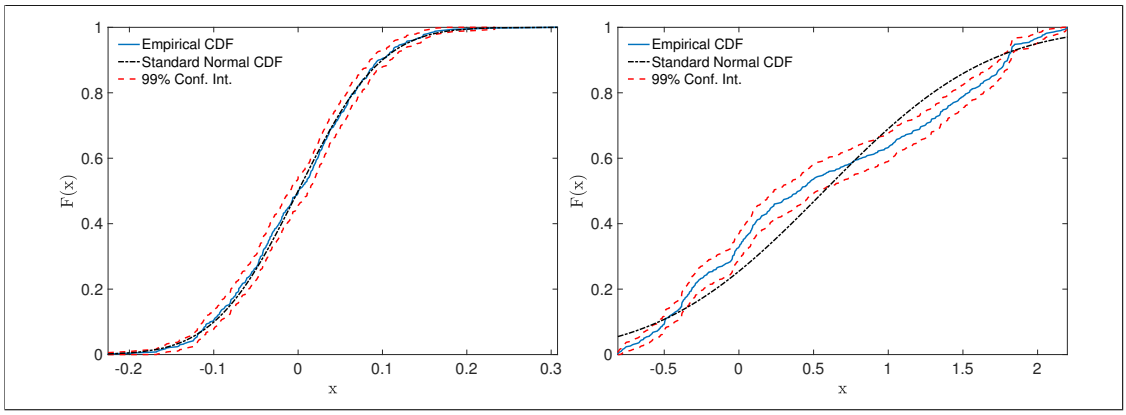

Figure 6. Normality visualization for preprocessed and non-preprocessed data

introducing challenges that are not present in domains involving only deterministic computation. For example, to even express the notion that a variable follows a continuous normal distribution, we need a large number of definitions from across mathematics: the real numbers (either as Cauchy sequences or Dedekind cuts or otherwise) and various algebraic properties thereof, the Lebesgue integral, some amount of measure theory and topology to express the prior, etc.

To manage this underlying complexity, we adopt a divide-and-conquer approach that synthesizes both "bottom-up" and "top-down" proof development. In order to demonstrate the applicability of the methods involved, we take as axioms certain high-level statistical theorems, allowing us to prove interesting properties about DDDAS. At the same time, we develop the lower-level theory modules for algebra, analysis, measure theory, etc., with the goal of eventually replacing the axiomatization of high-level propositions with actual proofs.

SAFETY ENVELOPE SENTINEL

The nature of the safety envelope coupled with the dynamic aspects of data-driven

systems has encouraged us to explore runtime-accessible envelope representations. The *runtime sentinel* represents a high-level system configuration / state constraint as an executable program in a runtime-accessible programming language (*e.g.* C or even architecture-specific machine code). In general, it is desirable to automatically generate these sentinels from a proof-accessible *embedded domain-specific language*, allowing sentinel correctness to be verified in the same manner as the code-generation module of a verified compiler [17], but at present we manually implement C-language sentinels to showcase the techniques involved. Integrating sentinels into the larger system (Figure 5) presents unique opportunities for runtime awareness of formal properties.

We have implemented a sentinel that checks both of the previously-discussed correctness conditions. First, it detects the normality of the sensor signal energy in the training data. This sentinel relies upon a standard statistical test for normality and some appropriate significance level. Next, the sentinel detects the continuous condition that the runtime signal energy falls within some safe interval. (By safe interval, we refer only to an interval where the model we are sure of the model's classification.) This is done using C floating-point arithmetic.

It is important to note that in both cases presented above, the sentinel does not correspond exactly with the formal assumption. In the first case, it is obvious that a test for normality up to some significance level does not imply that the data is actually normal. Further complicating things is the often-necessary introduction of a nontrivial preprocessing step: filtering, downsampling, etc. In Figure 6, we demonstrate the effects that pre-processing can have on the normality assumption. Perhaps less obviously, there is a similar concern in the second case, although it is far simpler: floating point arithmetic is not the same as arithmetic over the real numbers (*e.g.* floating point addition is not guaranteed to be commutative or associative in C). In future work, we would like to establish a formal connection between proof assumptions and runtime sentinels, but we expect this to be extremely complex even in the simplest cases. For now, sentinels provide a useful estimate of assumption validity at runtime.

Existing developments using runtime monitors in tandem with a verified system (*e.g.* ModelPlex [18]) focus on using runtime monitors to validate the system model against real behavior at runtime. Notably, the Copilot system [19] generates C runtime monitors from specifications written in a Haskell embedded domain-specific language, and can verify the equivalence of different code-generation backends using CompCert [20] and CBMC [21], a model-checking tool. Our use of runtime monitoring techniques is essentially similar, but different in character: rather than focusing on *whether* all of the properties proven about a model hold, we use runtime monitoring to alert users about *which* properties hold given the current system state.

CONCLUDING REMARKS

In this paper, we described an organizational scheme for proving stochastic properties about data-driven systems. This scheme is centered around safety envelopes, propositions of a particular structure that facilitate the generation of runtime sentinels. These sentinels inform a runtime system about the validity of static guarantees in a given instantaneous-time system state. We analyzed and discussed the assertions one might

make contingent upon such safe subsets when in the context of a broader data-driven system. In the future, we plan to use this architecture as the base for a complete "toolbox" for formalizing and proving correctness properties about data-driven systems. We also intend to explore more thoroughly some areas that were not touched upon explicitly here, in particular formalizing dynamic reintegration of data into the models.

ACKNOWLEDGMENTS

This research is partially supported by Air Force Office of Scientific Research Grant No. FA9550-19-1-0054, "Formal Verification of Stochastic State Awareness for Dynamic Data-Driven Intelligent Aerospace Systems" with Program Officer Dr. Erik Blasch.

This material is based upon work supported by the National Science Foundation under Grant No. 1816307. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

1. Darema, F. 2004. "Dynamic data-driven application systems: A new paradigm for application simulations and measurements," in *Computational Science-ICCS 2004*, Springer, pp. 662–669.
2. Blasch, E., É. Bossé, and D. A. Lambert. 2012. *High-level information fusion management and systems design*, Artech House.
3. Varela, C. A. 2013. *Programming Distributed Computing Systems: A Foundational Approach*, MIT Press.
4. Imai, S., S. Chen, W. Zhu, and C. A. Varela. 2017. "Dynamic Data-Driven Learning for Self-Healing Avionics," *Cluster Computing*, ISSN 1573-7543, doi:10.1007/s10586-017-1291-8.
5. Imai, S., E. Blasch, A. Galli, W. Zhu, F. Lee, and C. A. Varela. 2017. "Airplane Flight Safety Using Error-Tolerant Data Stream Processing," *IEEE Aerospace and Electronics Systems Magazine*, 32(4):4–17.
6. Chen, S., S. Imai, W. Zhu, and C. A. Varela. 2018. "Towards Learning Spatio-Temporal Data Stream Relationships for Failure Detection in Avionics," *Handbook of Dynamic Data-Driven Application Systems*:97–121.
7. Ricketts, D., G. Malecha, M. M. Alvarez, V. Gowda, and S. Lerner. 2015. "Towards verification of hybrid systems in a foundational proof assistant," in *Formal Methods and Models for Codesign (MEMOCODE), 2015 ACM/IEEE International Conference on*, IEEE, pp. 248–257.
8. Platzer, A. 2008. "Differential dynamic logic for hybrid systems," *Journal of Automated Reasoning*, 41(2):143–189.
9. Ghorbal, K., J.-B. Jeannin, E. Zawadzki, A. Platzer, G. J. Gordon, and P. Capell. 2014. "Hybrid theorem proving of aerospace systems: Applications and challenges," *Journal of Aerospace Information Systems*, 11(10):702–713.
10. Barras, B., S. Boutin, C. Cornes, J. Courant, J.-C. Filliatre, E. Gimenez, H. Herbelin, G. Huet, C. Munoz, C. Murthy, et al. 1997. "The Coq proof assistant reference manual: Version 6.1," .
11. Boldo, S., C. Lelay, and G. Melquiond. 2015. "Coquelicot: A user-friendly library of real analysis for Coq," *Mathematics in Computer Science*, 9(1):41–62.

12. Fontaine, P., J.-Y. Marion, S. Merz, L. P. Nieto, and A. Tiu. 2006. "Expressiveness+ automation+ soundness: Towards combining SMT solvers and interactive proof assistants," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, pp. 167–181.
13. Blanchette, J. C., S. Böhme, and L. C. Paulson. 2011. "Extending Sledgehammer with SMT solvers," in *International Conference on Automated Deduction*, Springer, pp. 116–130.
14. Armand, M., G. Faure, B. Grégoire, C. Keller, L. Théry, and B. Werner. 2011. "A modular integration of SAT/SMT solvers to Coq through proof witnesses," in *International Conference on Certified Programs and Proofs*, Springer, pp. 135–150.
15. Kopsaftopoulos, F. and F.-K. Chang. 2018. "A Dynamic Data-Driven Stochastic State-Awareness Framework for the Next Generation of Bio-inspired Fly-by-Feel Aerospace Vehicles," in *Handbook of Dynamic Data Driven Applications Systems*, Springer, pp. 697–721.
16. Kopsaftopoulos, F. 2019. "Data-driven stochastic identification for fly-by-feel aerospace structures: Critical assessment of non-parametric and parametric approaches," in *AIAA Scitech 2019 Forum*, p. 1534.
17. Leroy, X. 2009. "Formal verification of a realistic compiler," *Communications of the ACM*, 52(7):107–115.
18. Mitsch, S. and A. Platzer. 2016. "ModelPlex: Verified runtime validation of verified cyber-physical system models," *Formal Methods in System Design*, 49(1-2):33–74.
19. Pike, L., A. Goodloe, R. Morisset, and S. Niller. 2010. "Copilot: a hard real-time runtime monitor," in *International Conference on Runtime Verification*, Springer, pp. 345–359.
20. Leroy, X. et al. 2012. "The CompCert verified compiler," *Documentation and users manual. INRIA Paris-Rocquencourt*.
21. Kroening, D. and M. Tautschnig. 2014. "CBMC–C bounded model checker," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, pp. 389–391.