Dynamic Data-Driven Formal Progress Envelopes for Distributed Algorithms

Saswata Paul, Fotis Kopsaftopoulos, Stacy Patterson, and Carlos A. Varela

Rensselaer Polytechnic Institute, Troy, New York, 12180, USA
{pauls4,kopsaf}@rpi.edu, {sep,cvarela}@cs.rpi.edu

Abstract. This work presents *formal progress envelopes* applied to flight systems for distinctly classifying a system's state space into regions where a formal proof of progress for a distributed algorithm holds or does not hold. It also presents an approach for runtime integration of formal methods in the dynamic data-driven applications systems (DDDAS) architecture using *parameterized proofs*. Finally, it showcases the development of reusable parameterized proof libraries for high-level statistical and stochastic reasoning in the Athena proof assistant and demonstrates their use with a progress proof for the Paxos distributed consensus protocol.

1 Introduction

Intelligent aerospace systems of the future will be "smarter" and more self-sufficient in terms of self-diagnosis [5, 18], self-healing [16], safe navigation [24], and overall situational awareness. Such enhanced capabilities will stem from access to an unprecedented amount of real-time data collected from onboard sensors and a network of ground-stations, satellites, and aircraft, which we call the *Internet-of-Planes* (IoP). *Dynamic data-driven applications systems* (DDDAS) [6] can use this data for creating low-fidelity models in real-time that can reflect the operating conditions of an aerospace system almost as effectively as a high-fidelity model [23].

Data from the IoP can be used for various *mission-critical* and *safety-critical* applications such as *conflict-aware* [24] and *weather-aware* [7] navigation. Many *distributed consensus algorithms* [19, 22] allow participating aircraft to *eventually* reach agreement on data in a decentralized manner. However, the useful lifetimes of navigation and weather data are usually limited, rendering them obsolete after relatively short durations. Guarantees of eventual agreement are, therefore, ill-suited for most data-driven applications in the IoP. Moreover, the asynchronous and stochastic nature of real-life networks makes it impossible to provide deterministic guarantees about the progress of consensus algorithms, as message delays are indistinguishable from failures [8]. Under such circumstances, guarantees of probabilistic progress properties can be provided using statistical techniques. The failure of safety-critical aerospace systems can be catastrophic to life, property, or the environment [26], making it necessary to verify their correctness guarantees. Formal methods [3] can be used for the mechanical verification of such systems by writing machine-checked correctness proofs.

In this paper, we present the concept of *formal progress envelopes* applied to flight systems for distributed algorithms (Section 2) and propose an approach for integrating formal methods in the runtime architecture of DDDAS by using *parameterized proofs* (Section 3). We also showcase the development of a proof library in the Athena proof assistant [1] for reasoning about statistical properties of dynamic data-driven systems (Section 4) and provide a simple example to demonstrate its application (Section 5). We compare related work on runtime and stochastic verification (Section 6) and conclude the paper with a discussion about possible future directions of work (Section 7).

2 Formal Progress Envelopes

A *formal progress envelope* for a distributed algorithm is a computable subset of the system state space where the formal proof of a progress property holds. It is defined by a set of logical constraints parameterized by the system's operational conditions. Progress, depending on the algorithm, may refer to either termination or the successful completion of a given sequence of message rounds. To illustrate formal progress envelopes, let us consider asynchronous distributed consensus algorithms for use in safety-critical applications. The asynchronous and stochastic nature of real-life distributed systems makes it difficult to deterministically predict the message delay between any two nodes. However, we can statistically analyze the message delays – *e.g.*, if the message delays from a node A to another node B and vice-versa are represented by the continuous random variables X and Y respectively, over a period of time, it is possible to statistically observe the behavior of X and Y as their *probability density functions* (pdf) $f_X(t)$ and $f_Y(t)$. A distributed consensus algorithm may require multiple rounds of messages between two or more nodes for making progress. If in the worst-case scenario, a deterministic number of message rounds are required for reaching consensus, then the total worst-case message delay can be represented by a random variable which is the sum of the random variables representing the delays of each message involved. Since there exist statistical theorems about the nature of random variables (*e.g.*, *Cramér's Decomposition Theorem* [17]) it is possible to provide probabilistic guarantees about the worst-case time of progress – *e.g.*, "*The probability that the worst-case time for consensus will be at most 0.8 seconds is 98%*". A naive progress envelope for this particular property would be: $\forall D \in \mathbb{D} : D \sim \mathcal{N}(\mu_D, \sigma_D^2)$, where $\mathbb{D}$ is the set of all message delays and $D \sim \mathcal{N}(\mu_D, \sigma_D^2)$ implies that the random variable D is normally distributed.

Since the parameters that define formal progress envelopes can be quantified and measured, the envelopes can be analyzed against real-time data. Special runtime-accessible programs called *sentinels* [4] can analyze real-time data to check if the system state satisfies the progress envelope constraints. To ensure their correctness and effectiveness, sentinels may be generated directly from the formal specifications of the envelopes and some underlying models of uncertainty.

3 Augmenting DDDAS with Formal Methods

Unpredictable operating conditions of dynamic data-driven systems restrict the practicality of pre-developed formal proofs to the pre-deployment stages. It is, therefore, desirable to develop *parameterized proofs* that can be augmented in real-time, making them versatile over dynamic parameters – *e.g.*, instead of stating a static property such as: "*The probability that the round-trip message delay will be at most 0.9 seconds is 99%*", a parameterized proof would state: "*If the one-way message delays follow normal distributions* $\mathcal{N}(\mu_X, \sigma_X^2)$ *and* $\mathcal{N}(\mu_Y, \sigma_Y^2)$*, then the probability that the round-trip message delay will be at most* t *seconds is* $\boldsymbol{F}(\mu_X, \sigma_X, \mu_Y, \sigma_Y, t)$", where $\boldsymbol{F}$ is the cumulative distribution function (cdf).

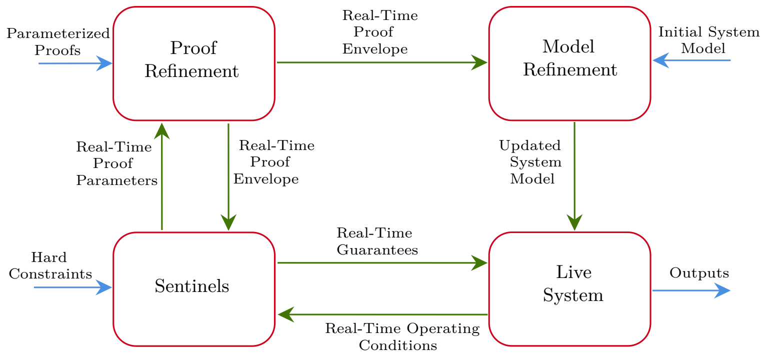

Fig. 1: A feedback loop for integrating formal methods in the DDDAS architecture.

Formal envelopes, in conjunction with parameterized proofs and runtime sentinels, can be directly incorporated in the DDDAS architecture using a feedback loop (Fig. 1). This involves four logically separate components:

- The *proof refinement* component receives a parameterized proof with a set of initial parameters and real-time inputs from the sentinels. If possible, it refines the proofs with the real-time parameters and provides new envelopes.
- The *model refinement* component receives real-time envelopes from the proof refinement component and an initial system model. It updates the system model according to the latest envelopes.
- The *sentinels* analyze the real-time operating conditions of the system against the latest envelopes. If the conditions do not conform to the envelopes, they send real-time parameters to the proof refinement component. They also inform the live system about guarantees that hold in real-time. Hard constraints dictate when possible guarantees cease to be useful.
- The *live system* runs using the updated system model from the model refinement component.

To illustrate, let us consider an aircraft that is participating in distributed consensus for collaborative flight-planning by implementing the feedback loop in

Fig. 1. It may propose a conflict-aware flight-plan p [24] that is due to start after 1 second, as there is an initial proof that traffic aircraft will reach consensus under 1 second with 99.8% probability. During runtime, the sentinels observe that the actual message delays do not conform to the envelope of the initial proof, so they send the actual parameters to the proof refinement component. The proof refinement component generates new proofs which state that consensus under 1 second has a probability of only 60% and consensus under 2 seconds has a probability of 99.5%. The model refinement component may then update the proposal with another flight-plan p' that is due to start after 2 seconds.

The above example shows that our approach can be used to dynamically update formal proofs with parameters that reflect the runtime operating conditions of a system. This will allow the development of highly-adaptive dynamic data-driven applications systems that can adapt to formally-verified properties that hold during runtime, thus extending the practicality of formal verification techniques beyond the pre-deployment stages.

4 Proof Library for High-Level Statistical Inference

The nature of foundational verification makes the process of developing mechanically verified formal proofs an arduous task. This is because when developing the proofs of high-level properties in mechanically-verifiable languages, a significant amount of effort needs to be put in for formalizing the lower-level theory. This calls for the development of proof libraries for formal verification languages, which can be reused to prove higher-level properties, similar to code libraries developed for general-purpose programming languages. Mechanically verifying the higher-level statistical properties of stochastic systems requires reasoning about the lower-level mathematical theory of random variables, distributions of random variables, algebra, and probability. We adopt a top-down approach of proof development which allows us to first formalize the high-level properties and then develop the lower-level theory required for fully verifying them. We present two mechanically-verified results that can be used for reasoning about the statistical properties of stochastic systems.

Lemma 1 *Given two normal probability density functions $f_X(x) = \mathcal{N}(\mu_X, \sigma_X^2)$ and $f_Y(y) = \mathcal{N}(\mu_Y, \sigma_Y^2)$, the probability that a random variable following their convolution will take a value of at most r is given by: $\int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$ where $z = \frac{r-(\mu_X+\mu_Y)}{\sqrt{2(\sigma_X^2+\sigma_Y^2)}}$.*

Theorem 1 *If two independent random variables X and Y are normally distributed with probability density functions $f_X(x) = \mathcal{N}(\mu_X, \sigma_X^2)$ and $f_Y(y) = \mathcal{N}(\mu_Y, \sigma_Y^2)$, then the probability that $X + Y$ will take a value of at most r is given by: $\int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$ where $z = \frac{r-(\mu_X+\mu_Y)}{\sqrt{2(\sigma_X^2+\sigma_Y^2)}}$.*

The formal proof of Theorem 1 uses Cramèr's Decomposition Theorem and the following postulates:

Postulate 1 *The standard score of a value v with respect to a normal distribution $\mathcal{N}(\mu_X, \sigma_X^2)$ is given by: $z = \frac{v-\mu_X}{\sigma_X}$*

Postulate 2 *Given the standard score z of a value v with respect to a normal distribution f_X, the probability that a random variable X following f_X will take values of at most v is given by: $\int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$*

Postulate 3 *Given two independent random variables X and Y, the pdf of their sum $X + Y$ is the convolution of their individual pdfs.*

```
conclude # Lemma 1
(forall X Y T .
  ((and (is_normal X) (is_normal Y))
    ==>
    (= (probability (convolution X Y) T)
      (integral_SND (z_score T (convolution X Y))))))
pick-any x:Dist
pick-any y:Dist
pick-any t:Real
assume (and (is_normal x) (is_normal y))
let{xyNormal := (and (is_normal x) (is_normal y));
  xyCramers := (!uspec (!uspec Cramers_Decomposition_Theorem x) y);
  xPlusyNormal := (!mp xyCramers xyNormal);
  txPlusyProbability := (!uspec (!uspec probability_result t) (convolution x y))}
(!mp txPlusyProbability xPlusyNormal)
```

```
conclude # Theorem 1
(forall x y T .
  (and (is_normal (pdf x)) (is_normal (pdf y)))
    ==>
        (= (probability_randVar (sum x y) T)
           (integral_SND (z_score T (convolution (pdf x) (pdf y))))))
pick-any x:RandVar
pick-any y:RandVar
pick-any T:Real
assume  (and (is_normal (pdf x)) (is_normal (pdf y)))
let{xPDF := (pdf x);
    yPDF := (pdf y);
    xyPDFnormal := (and  (is_normal xPDF) (is_normal yPDF) );
    convolutionxyPDF := (convolution xPDF yPDF);
    z := (sum x y);
    xySumRandVars := (!uspec (!uspec sum_randVars x) y);
    zPDF := (pdf z);
    zPDFConvolution := (!chain [(pdf z)
                              = (pdf (sum x y))   [z]
                              = convolutionxyPDF [xySumRandVars]]);
    probthm1 := (!uspec (!uspec (!uspec Probability_Theorem1 xPDF) yPDF) T);
    probthm1result := (!mp probthm1 xyPDFnormal)}
(!chain [(probability_randVar (sum x y) T)
     = (probability (pdf (sum x y) ) T) [probability_randVar_axiom]
     = (probability (pdf z) T)          [z]
     = (probability convolutionxyPDF T) [zPDFConvolution]
     = (probability (convolution xPDF yPDF) T) [convolutionxyPDF]
     = (integral_SND (z_score T (convolution (pdf x) (pdf y)))) [probthm1result]])
```

We have used the Athena proof assistant to formalize and mechanically verify our proofs (shown above). Our work extends the Athena proof library [2] with theory about statistical inference that can be reused for higher-level proofs[1].

[1] Available at http://wcl.cs.rpi.edu/pilots/fvdddas

5 A Sample Application of our Proof Library

We demonstrate how our extensions to the Athena proof library can be used to provide a mechanically-verified proof of probabilistic progress for a simple implementation of Paxos [20], a consensus algorithm which involves a set of agents called *proposers* that propose values to be chosen and a set of agents called *acceptors* that vote on those values. For our example, we consider a system in which there is one proposer and two acceptors. Paxos assumes an asynchronous, non-Byzantine system model where agents operate at arbitrary speed, may fail and restart, and have stable storage. Messages can be duplicated, lost, and have arbitrary transmission times, but are not corrupted. For the sake of simplicity, we make some additional assumptions – all agents are always available; there is no message loss; it is possible to observe the pdf of the message delay between any pair of agents as a normal distribution; and it is possible to observe the pdf of the processing time of every agent as a normal distribution. Our implementation of Paxos involves two *prepare* messages from the proposer to the acceptors, followed by two *promise* messages from the acceptors to the proposer, and finally, two *accept* messages from the proposer to the acceptors. The agents take some time to process each message. The algorithm makes progress when all messages have been transmitted, received, and processed.

We can define the worst-case scenario as the sequential operation of the protocol where no pair of actions (processing or message transmission) have any overlap in time. The total time for progress will, therefore, be the sum of the total processing time and the total message delay. If the message delays and processing times are represented by the sets of random variables $\mathbb{X} = \{X_1, X_2, X_3, X_4, X_5, X_6\}$ and $\mathbb{Y} = \{Y_1, Y_2, Y_3, Y_4, Y_5, Y_6\}$ respectively, then under our assumptions, Cramer's Decomposition Theorem can be recursively used to prove that $Z = \sum_{i=1}^{6} X_i + \sum_{i=1}^{5} Y_i$ follows a normal distribution $\mathcal{N}(\mu_Z, \sigma_Z)$. Theorem 1 can then be used to prove the following stochastic progress property:

Theorem 2 *The probability that $Z + Y_6$ will take a value of at most c_2 is given by: $\int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$ where $z = \frac{c_2 - (\mu_{Y_6} + \mu_Z)}{\sqrt{2(\sigma_{Y_6}^2 + \sigma_Z^2)}}$.*

A straightforward progress envelope for Theorem 2 would be the constraint $\forall X \in \mathbb{X} : X \sim \mathcal{N}(\mu_X, \sigma_X^2) \wedge \forall Y \in \mathbb{Y} : Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$.

6 Related Work

Runtime monitoring of formal properties has been previously investigated in [21] and [25]. Formal safety envelopes for stochastic state identification have been proposed in [4]. Formal verification of expectation and variance of discrete random variables and tail distribution bounds have been studied in [12] and [13]. Formalizations of the uniform random variable and continuous probability distributions in the HOL theorem prover [15] have been presented in [11] and [10]. Probabilistic analysis of wireless systems has been studied in [14]. [9] addresses probabilistic theorem proving as the problem of computing the probability of a logical formula given the probabilities of a set of formulas.

The existing work does not focus on augmenting formal proofs with runtime data or creating a dedicated proof library that can be directly reused for higher-level statistical properties of stochastic aerospace systems. We improve upon it by introducing runtime-modifiable formal proof techniques in the DDDAS architecture to allow the development of safety-critical aerospace systems that can dynamically adapt to the formal proofs that hold during runtime.

7 Conclusion

We have presented an approach for integrating formal methods directly in the dynamic data-driven applications systems (DDDAS) architecture that will allow the development of highly-adaptive formally-verified aerospace systems. We have also showcased the development of formal proof libraries in the Athena proof assistant that can be used as reusable building blocks to develop proofs of higher-level probabilistic properties of stochastic systems.

Real-life data is seldom perfect – *e.g.*, normality of sensor data may only be tested up to some significant level and may also be affected by pre-processing. It is also computationally expensive to effectively estimate the tail bounds of distributions in real-time. Future directions of work, therefore, include investigating the development of formally-verified runtime sentinels that can find accurate and meaningful estimates from data and further expansion of our stochastic proof library in Athena by creating parameterized proofs for lower-level theory.

Acknowledgment: This research was partially supported by the National Science Foundation (NSF), Grant No. – CNS-1816307 and the Air Force Office of Scientific Research (AFOSR), DDDAS Grant No. – FA9550-19-1-0054.

References

1. Arkoudas, K.: Athena. http://proofcentral.org/athena, http://proofcentral.org/athena
2. Arkoudas, K., Musser, D.: Athena Libraries. http://proofcentral.org/athena/lib, http://proofcentral.org/athena/lib
3. Arkoudas, K., Musser, D.: Fundamental Proof Methods in Computer Science: A Computer-Based Approach. MIT Press (2017)
4. Breese, S., Kopsaftopoulos, F., Varela, C.: Towards proving runtime properties of data-driven systems using safety envelopes. In: The 12th International Workshop on Structural Health Monitoring. Stanford, CA (Sep 2019)
5. Chen, S., Imai, S., Zhu, W., Varela, C.A.: Towards learning spatio-temporal data stream relationships for failure detection in avionics. Handbook of Dynamic Data-Driven Application Systems pp. 97–121 (2018)
6. Darema, F.: Dynamic data-driven application systems: A new paradigm for application simulations and measurements. In: Computational Science-ICCS 2004. pp. 662–669. Springer (2004)

7. DeLaura, R., Robinson, M., Pawlak, M., Evans, J.: Modeling convective weather avoidance in enroute airspace. In: 13th Conference on Aviation, Range, and Aerospace Meteorology, AMS, New Orleans, LA (2008)
8. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. Journal of the ACM (JACM) **32**(2), 374–382 (1985)
9. Gogate, V., Domingos, P.: Probabilistic theorem proving. Communications of the ACM **59**(7), 107–115 (2016)
10. Hasan, O., Tahar, S.: Formalization of continuous probability distributions. In: International Conference on Automated Deduction. pp. 3–18. Springer (2007)
11. Hasan, O., Tahar, S.: Formalization of the standard uniform random variable. Theoretical Computer Science **382**, 71–83 (2007)
12. Hasan, O., Tahar, S.: Using theorem proving to verify expectation and variance for discrete random variables. Journal of Automated Reasoning **41**(3-4), 295–323 (2008)
13. Hasan, O., Tahar, S.: Formal verification of tail distribution bounds in the HOL theorem prover. Mathematical Methods in the Applied Sciences **32**(4), 480–504 (2009)
14. Hasan, O., Tahar, S.: Probabilistic analysis of wireless systems using theorem proving. Electronic Notes in Theoretical Computer Science **242**(2), 43–58 (2009)
15. Hurd, J.: Formal verification of probabilistic algorithms. Ph.D. thesis, University of Cambridge (2002)
16. Imai, S., Chen, S., Zhu, W., Varela, C.A.: Dynamic data-driven learning for self-healing avionics. Cluster Computing **20**, 1–24 (Nov 2017). https://doi.org/10.1007/s10586-017-1291-8
17. Jaynes, E.T.: Probability theory: The logic of science. Cambridge University Press (2003)
18. Kopsaftopoulos, F.: Data-driven stochastic identification for fly-by-feel aerospace structures: Critical assessment of non-parametric and parametric approaches. In: AIAA Scitech 2019 Forum. pp. 15–34 (2019)
19. Lamport, L.: The Part-Time Parliament. ACM Transactions on Computer Systems (TOCS) **16**(2), 133–169 (1998)
20. Lamport, L.: Paxos Made Simple. ACM SIGACT News **32**(4), 18–25 (2001)
21. Mitsch, S., Platzer, A.: Modelplex: Verified runtime validation of verified cyber-physical system models. Formal Methods in System Design **49**(1-2), 33–74 (2016)
22. Ongaro, D., Ousterhout, J.: In Search of an Understandable Consensus Algorithm. In: 2014 USENIX Annual Technical Conference. pp. 305–319 (2014)
23. Paul, S., Hole, F., Zytek, A., Varela, C.A.: Wind-aware trajectory planning for fixed-wing aircraft in loss of thrust emergencies. In: The 37th AIAA/IEEE Digital Avionics Systems Conference. pp. 558–567. London, England (Sep 2018)
24. Paul, S., Patterson, S., Varela, C.A.: Conflict-Aware Flight Planning for Avoiding Near Mid-Air Collisions. In: The 38th AIAA/IEEE Digital Avionics Systems Conference. San Diego, CA (Sep 2019)
25. Pike, L., Goodloe, A., Morisset, R., Niller, S.: Copilot: a hard real-time runtime monitor. In: International Conference on Runtime Verification. pp. 345–359. Springer (2010)
26. Sommerville, I.: Software engineering. Addison-Wesley/Pearson (2011)