Towards Formal Correctness Envelopes for Dynamic Data-Driven Aerospace Systems

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

Rensselaer Polytechnic Institute, Troy, NY 12180, USA
{pauls4,pattes3,kopsaf,varelc}@rpi.edu

Abstract. Intelligent aerospace systems of the future are expected to be "smarter" and more self-sufficient in terms of self-diagnosis, self-healing, self-navigation, and overall situational awareness. Enhanced situational awareness will be facilitated by access to a vast amount of real-time data from on-board sensors, other aircraft, ground stations, and satellites, as well as contextual models from environment analysis of weather, terrain, and structures. The Dynamic data-driven application systems (DDDAS) paradigm incorporates real-time data for creating high-fidelity models to aid in flight-diagnosis and decision-making. DDDAS techniques accommodate the fusion of dynamic-data, algorithms, computation, and interpretation, making them apposite for use in safety-critical intelligent sensing systems. In safety-critical systems, it is important to have irrefutable system assurance over hardware and software guarantees. Formal methods allow the development of machine-checked correctness proofs for such guarantees, facilitating the verification of such systems on infinite states. This chapter presents formal correctness envelopes (FCE), analogous to performance envelopes of an aircraft, which represent the operating conditions under which the guarantees regarding a system's properties hold. FCEs are data-driven, allowing them to be computed, quantified, and monitored in real-time using correctness sentinels. Correctness sentinels are executable programs that use the notion of correctness envelopes to monitor real-time data-streams and detect the status of a system's state with respect to relevant envelopes during runtime. At any given point of time, correctness sentinels can provide useful information about which system properties can be guaranteed and with how much confidence. FCEs, in tandem with correctness sentinels, allow the development of aerospace systems that can monitor formal guarantees in real-time and dynamically adapt to remain within the bounds of those guarantees.

Keywords: data-driven; formal methods; fly-by-feel; Dynamic Data-Driven Applications Systems; Dynami Data-Driven Aerospace Systems; correctness envelopes; correctness sentinels

1 Introduction

The future of aviation is going to be largely autonomous, giving way to more intelligent and situationally-aware flight control systems. Smart aerospace systems of the future will be able to analyze their own health and environmental

conditions and make decisions to optimize flight-performance while ensuring safe navigation through the national airspace system (NAS). Intelligence in the NAS will stem from an influx of valuable real-time data from both local sources such as on-board physical sensors and remote sources such as other aircraft, ground-stations, and satellites [24, 48, 50]. Dynamic data-driven application systems (DDDAS) is a paradigm that allows models of a system under consideration to dynamically incorporate new data [7, 14]. In addition, DDDAS enables an application to influence the data collection and real-time measurement processes towards more effective collection and measurement of data, thus leading to better quality of data specifically suited for the application and more broadly, model-cognizant control of the system's instrumentation.

The DDDAS paradigm involves the use of a feedback loop which updates an existing application model with real-time data to create a more accurate model, reflecting system behaviors. When DDDAS techniques are integrated into aerospace systems, they are known as *dynamic data-driven aerospace systems*. Dynamic data-driven aerospace systems employ DDDAS concepts for dynamic integration and unification of real-time data for a wide range of aerospace-related applications and can be extremely effective in designing high fidelity diagnostic and decision-support systems for both manned and unmanned aerial vehicles (UAVs). DDDAS-aerospace techniques have been used for applications ranging from error detection and recovery from sensor failures [20–23, 28] to generation of high-fidelity emergency trajectories by considering aircraft flight-performance [45–47].

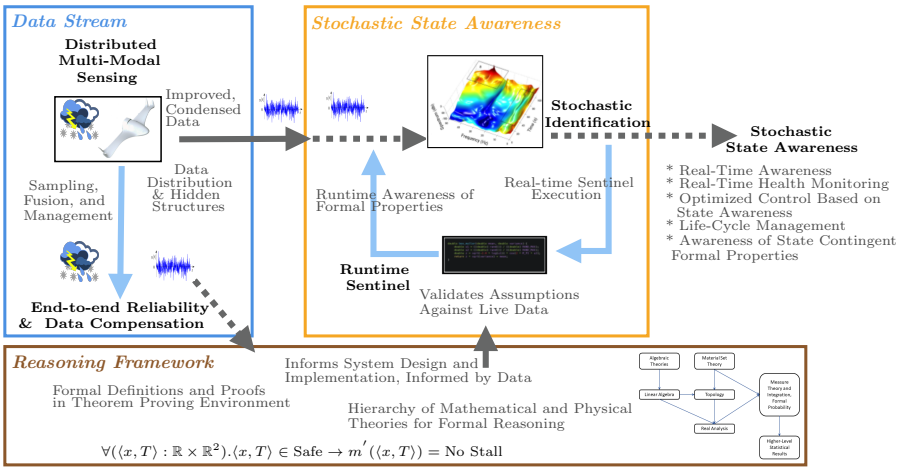

Fig. 1: Formal methods in dynamic data-driven aerospace system workflow.

Aerospace systems are *safety-critical* which means that their failures can be catastrophic to life, environment, and property [58]. Therefore, it is imperative to have high confidence in the correctness of the software components that are

used in engineering these systems. However, dynamic data-driven aerospace systems are, more often than not, inherently complex, making them vulnerable to specification and/or implementation errors. Under such dynamic conditions, formal methods can be used to mechanically verify the guarantees provided by a system. Fig. 1 depicts a sample framework for the integration of formal methods in dynamic data-driven aerospace systems workflow, such as software verification techniques to reason about and monitor system properties during runtime. Formal methods provide a way to perform a more thorough and exhaustive validation of software systems than traditional methods like unit testing [9].

Formal methods allow *interactive theorem-proving*, which is the process of formally specifying properties, and then proving that these properties are correct within a *proof assistant* [2, 4, 12, 44]. Using a proof assistant is much like developing a proof of correctness on paper, except that the proof assistant ensures that proofs are complete, *i.e.*, that theorems are logical consequences of axioms. Interactive methods are interesting because they can be used without sacrificing the undoubtedly-useful automated methods. Tools like Satisfiability Modulo Theory (SMT) [5] 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". As such, a high-level interactive theorem-proving system 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 [9].

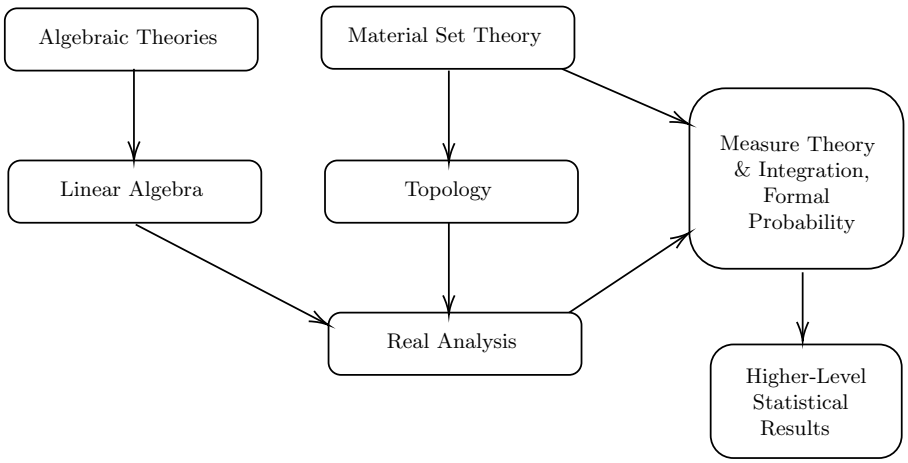

Fig. 2: Hierarchical nature of theories for reasoning about high-level stochastic properties.

Although formal methods can be extremely beneficial in verifying the properties of dynamic data-driven aerospace systems, there are some barriers to using such tools, as they would be for any system operating under dynamic conditions

without statically-predetermined outcomes. The nature of foundational verification is a somewhat daunting task in the formal setting of a proof assistant. Fig. 2 shows the hierarchy of mathematical theories [18, 26, 43, 61] that must be (in full or in part) formalized to allow high-level reasoning about statistics and properties of stochastic systems. The complexity of stochastic formal methods resides beneath even the simplest statements, 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, a large number of definitions from across mathematics are needed: the real numbers (*e.g.,* Cauchy sequences [53] or Dedekind cuts [57] or otherwise) and various algebraic properties thereof (*e.g.,* the Lebesgue integral [11], some amount of measure theory and topology to express the prior, etc) (Fig. 2). To manage this underlying complexity, a divide-and-conquer approach that synthesizes both "bottom-up" and "top-down" proof development may be adopted. Sometimes, libraries like Coquelicot [8] may be used to develop required formalizations, but to quickly demonstrate the applicability of the methods involved, certain high-level statistical theorems may be taken as axioms, affording the ability to prove interesting properties about DDDAS. At the same time, the lower-level theory modules for algebra, analysis, measure theory, etc. can be developed, with the goal of eventually replacing the axiomatization of high-level propositions with actual proofs.

This chapter presents the concept of *formal correctness envelopes* (FCE) and *correctness sentinels* for dynamic data-driven aerospace systems [9, 13, 48]. Aerospace systems of the future will be capable of performing a wide spectrum of functions ranging from flight-diagnosis to autonomous navigation. These functions will utilize different types of algorithms which will have different criteria for correctness (*e.g.,* an algorithm that computes emergency trajectories will be correct if it can compute safe trajectories within some predefined time limit while an algorithm that detects sensor failures will be correct if it can successfully detect any error in a sensor data-stream). When the correctness guarantees of software systems are expressed formally and proven with the help of proof assistants, the proofs are successful only if certain logical conditions hold in the underlying context. For dynamic data-driven aerospace systems, these conditions imply that the properties guaranteed by the proofs may hold only under a computable subset of the application state space. Correctness envelopes represent the constraints on the operating conditions under which the correctness guarantees of a property of a system are valid. These constraints can be continuous (*e.g.,* membership in some interval of the state space) or non-continuous (*e.g.,* the Gaussian distribution [40] of the sample history of a sensor). The presence of dedicated correctness envelopes for different properties provides the opportunity for monitoring the system state during runtime to detect which properties can be guaranteed. Correctness sentinels are special runtime accessible programs that can analyze the system state in runtime and provide information about the status of the system state with respect to the correctness envelopes for different system properties. FCEs and sentinels allow the development of highly-adaptive

DDDAS applications that can monitor formal guarantees in real-time and adjust their operations accordingly to remain within the envelopes of those guarantees.

The rest of the chapter is divided as follows: Section 2 explores some examples of applications of DDDAS in aerospace systems; Section 3 introduces formal correctness envelopes, their potential applications in dynamic data-driven aerospace systems, and correctness envelope sentinels; Section 4 discusses experiments and results; Section 5 discusses related work; and finally, Section 6 concludes the chapter with a discussion about potential future directions of work.

2 DDDAS in Safety-Critical Aerospace Systems

Akin to biological organisms, autonomous *fly-by-feel* aerospace systems of the future will be able to detect changes in their operation, environment, and their own structural health to make appropriate changes in their behavior or operations [25]. Moreover, the future of the NAS will witness a significant rise in aircraft density (augmented further with the advent of civilian and other classes of drones), thus making it necessary to develop smarter flight-control systems that can autonomously navigate through the airspace, plan trajectories during emergencies, and maintain *standard separation* from aircraft to avoid *near mid-air collisions* (NMAC) [49]. This section showcases a few examples of the application of DDDAS in modern aerospace systems for purposes like self-diagnosis, improved situational awareness, and decision-support for avionics and pilots.

2.1 Fly-by-feel state awareness method for stall detection

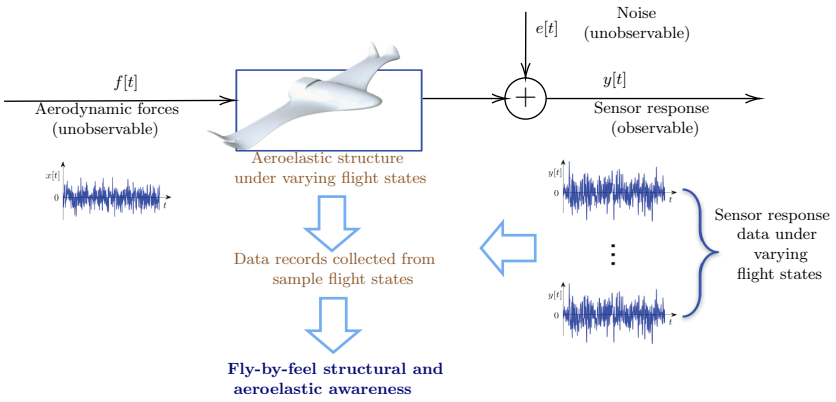

Fig. 3: Data-driven fly-by-feel structural and aeroelastic awareness.

Traditional aircraft state awareness methods, in addition to altitude, airspeed, acceleration, turbulence, etc, with respect to stalling awareness situations,

are limited to the attitude information and do not take into account structural and aeroelastic feedback from the aircraft [30]. A data-driven fly-by-feel state awareness method for stall detection uses response signals recorded from piezoelectric sensors placed on the wings of an aircraft [31] to detect the stall states of an aircraft. Fig. 3 shows the schematic representation of a framework which uses data from the sensors under varying flight states. Given dynamic noise-corrupted response-only data records collected from a sample of the admissible flight states, each state can be characterized by a specific airspeed and *angle of attack* (AoA) and kept constant for the duration of the data collection. Hence, it is possible to develop appropriate fly-by-feel methods capable of monitoring and detecting aerodynamic stall without the use of attitude information [32]. The

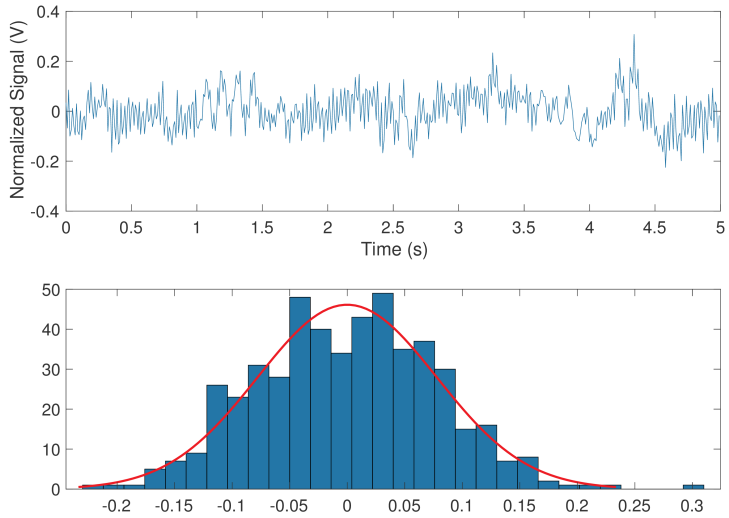

Fig. 4: Visualization of experimental signal energy used to train model.

mathematical model for stall detection is based upon experimental wind tunnel data, obtained by controlling the AoA and airspeed configuration and recording the mean and variance of signal energy. Fig. 4 illustrates a visualization of the signal energy collected from one sensor on the wing (for a fixed airspeed of 15 m/s and 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.

2.2 Emergency trajectory generation

In aircraft loss-of-thrust emergencies (resulting from partial or complete loss of engine power), the response time is critical and it is imperative to quickly

provide pilots with feasible landing trajectories. A dynamic data-driven approach (Fig. 5) can be employed for generating emergency trajectories for a fixed-wing aircraft that has sustained physical damages [45–47]. There are four components

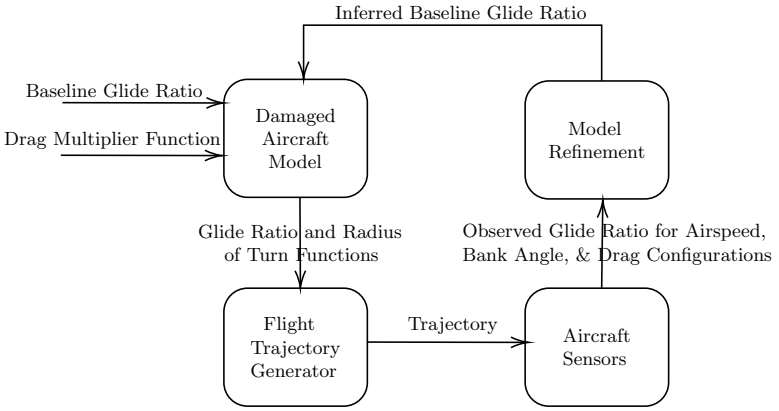

Fig. 5: Dynamic data-driven feedback loop for emergency trajectory generation.

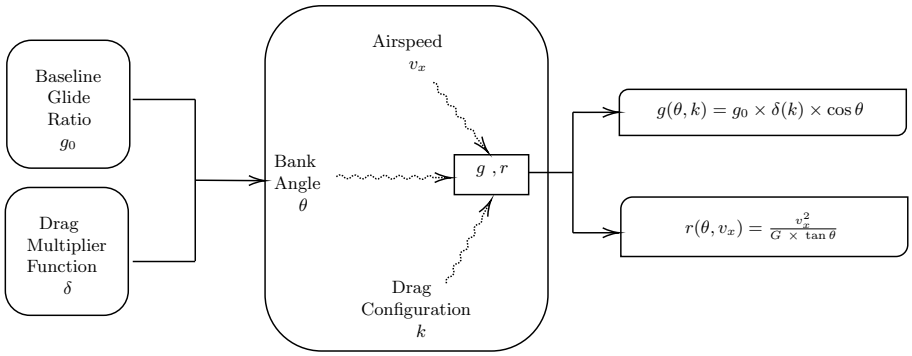

Fig. 6: The damaged aircraft model.

to consider for emergency response: the *damaged aircraft model*, *flight trajectory generator*, *aircraft/sensors* and the *model refinement component*. The damaged aircraft model (Fig. 6) takes as input the new baseline glide ratio g_0 and a drag multiplier function δ. For every possible bank angle θ, airspeed v_x, and drag configuration k, the model computes the corresponding glide ratio $g(\theta, k)$ and the radius of turn $r(\theta, v_x)$ and sends it to the trajectory generator. A continuous stream of the pressure altitude z and the airspeed v from the sensors are used to estimate the observed glide ratio $\hat{g}(\theta, k)$. For a given instant t_i, $\hat{g}_{t_i}(\theta, k)$ is the ratio of the horizontal distance traveled to the altitude lost in the preceding η

seconds.

$$\hat{g}_{t_i}(\theta, k) = \frac{\Delta x(\eta, i)}{\Delta z(\eta, i)} = \frac{\sum_{i-\eta}^{i} v_{t_i}}{z_{t_{i-\eta}} - z_{t_i}}$$

Anomalous data is filtered by detecting *stable windows of descent* (SWD), which satisfy the following conditions:

- they are intervals of steady descent with no vertical acceleration;
- the distribution of $\hat{g}_{t_i}(\theta, k)$ in a SWD has a standard deviation within a known threshold σ_τ.

When a SWD ω is detected, the observed glide ratio $\hat{g}_\omega(\theta, k)$ is computed for ω by taking a mean of all values of $\hat{g}_{t_i}(\theta, k)$ observed in ω.

$$\hat{g}_\omega(\theta, k) = \frac{\sum_{i=1}^{n} \hat{g}_{t_i}(\theta, k)}{n} \text{ for all } t_i \in \omega$$

The dynamic data-driven feedback loop in Fig. 5 allows the use of real-time sensor data to compute the glide ratio of an accident aircraft during an emergency. The resulting trajectories are high fidelity and practical, as they take into consideration the real-time flight capabilities of the accident aircraft.

2.3 Conflict-aware flight planning

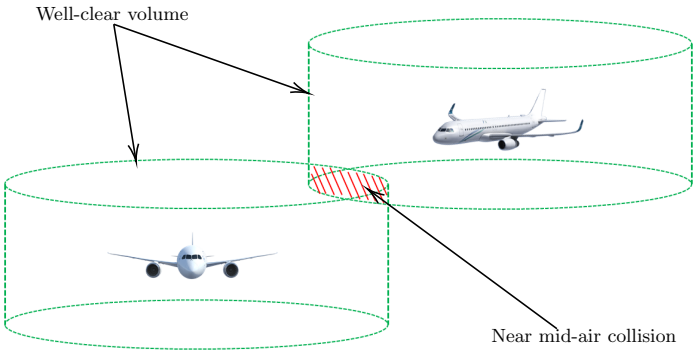

Fig. 7: An NMAC is caused by the intersection of the WCVs of two aircraft.

Loss of *standard separation* between two aircraft can be potentially catastrophic since it can cause an NMAC or a wake-vortex induced roll [49]. The *well-clear volume* (WCV) of an aircraft is a cylindrical volume of space (with

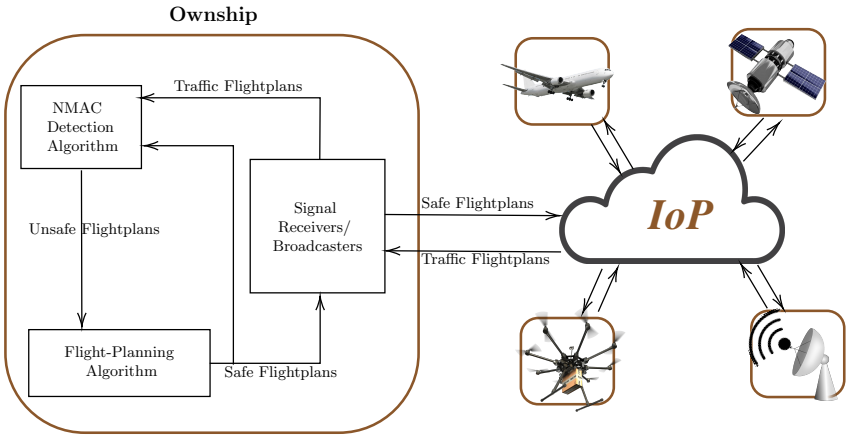

Fig. 8: Dynamic data-driven feedback loop for conflict-aware flight planning.

diameter D and height H) surrounding the aircraft [42]. Two aircraft are in an NMAC if their WCVs intersect (Fig. 7).

Given the 2D horizontal position and velocity vectors of two aircraft a and b at time t_0 as $\mathbf{s}^a_{t_0}$, $\mathbf{s}^b_{t_0}$, $\mathbf{v}^a_{t_0}$, and $\mathbf{v}^b_{t_0}$, and their vertical position and velocity vectors as $s^a_{t_0}$, $s^b_{t_0}$, $v^a_{t_0}$, and $v^b_{t_0}$ [41], then, assuming both aircraft maintain constant velocity flight, their relative position and relative velocity in the horizontal (xy) and vertical dimensions (z) at time t_0 can be obtained by the following equations:

$$\mathbf{s}_{xy,t_0} = \mathbf{s}^a_{t_0} - \mathbf{s}^b_{t_0}$$
$$\mathbf{v}_{xy,t_0} = \mathbf{v}^a_{t_0} - \mathbf{v}^b_{t_0}$$
$$s_{z,t_0} = s^a_{t_0} - s^b_{t_0}$$
$$v_{z,t_0} = v^a_{t_0} - v^b_{t_0}$$

Their relative horizontal and vertical positions at any time $t \geq t_0$ can now be computed as:

$$\mathbf{s}_{xy,t} = \mathbf{s}_{xy,t_0} + (t - t_0)\mathbf{v}_{xy,t_0}$$
$$s_{z,t} = s_{z,t_0} + (t - t_0)v_{z,t_0}$$

An NMAC is possible at time t if the following conditions simultaneously hold:

$$\|\mathbf{s}_{xy,t}\| < D$$
$$|s_{z,t}| < H$$

For conflict-aware flight planning [49], an aircraft (the *ownship*) collects air-traffic data from a network of ground-stations, satellites, and aircraft called the *Internet-of-Planes* (IoP) [48, 50]. The dynamic data-driven feedback loop for conflict-aware flight planning (Fig. 8) updates the NMAC detection algorithm by using real-time data from the IoP, which can enhance the sensing range of the ownship. The real-time data allows the ownship to monitor and resolve (if possible) potential NMACS between its current flight-plan and traffic aircraft.

3 Formal Correctness Envelopes for DDDAS

As discussed in Section 1, correctness properties of a system can be logically verified using machine-checked formal proofs. Formal proofs are written and checked using proof assistants and each proof depends on certain logical preconditions. Some of these preconditions are data-driven and can be quantified and measured using data collected from local and remote sensors. Such data-driven preconditions allow defining FCEs for a system as computable subsets of the DDDAS state space where the correctness properties of the system can be irrefutably guaranteed. For specific system properties, the FCEs are determined by specific types of data. At any time, the system may be well within the boundaries of the FCE for one property, while being outside the FCE for some other property.

This section describes two use cases to showcase the use of FCEs in dynamic data-driven aerospace systems – the first use case has been motivated by the use of local sensor data to determine the characteristics of the flight-profile of an aircraft during flight, as discussed in Section 2.1; the second use case has been motivated by the use of remote sensor data (sensors, here, being other aircraft, ground-stations, and satellites) for conflict-aware flight-planning application in the IoP, as discussed in Section 2.3.

3.1 A formal safety envelope

Considering the model used for fly-by-feel state awareness described in Section 2.1 [9], there are two distinct scenarios where one would like to verify correctness properties. First, given the assumption that the experimental signal energy data collected from the piezoelectric sensors is normally distributed, one would 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, the model can be represented 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 the model $n = 90000$). This function can be uncurried[1] to a function

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

which allows us to treat pairs of training data and runtime signal energies as system states.

A formal *safety*[2] *envelope*, analogous to a flight safety envelope describing the safe operating conditions of an aircraft, is a computable subset of the DDDAS

[1] A function which takes all its arguments at once.
[2] "Safety" in this context implies correct operation of a system

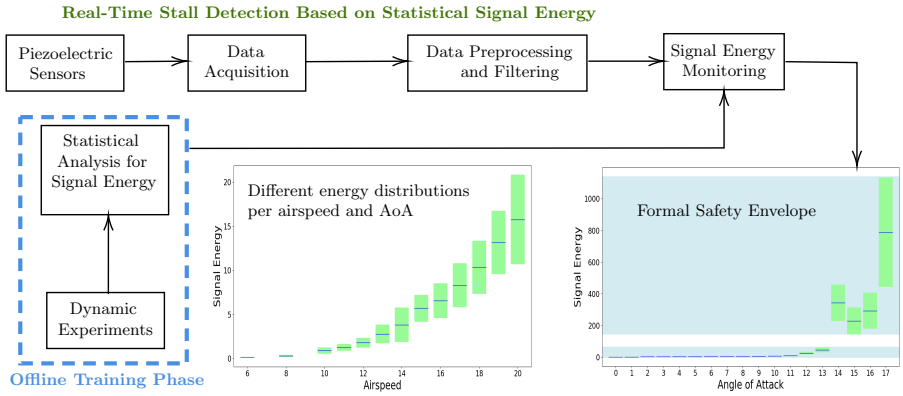

Fig. 9: Safety envelope for data-driven stall awareness model.

state space that describes the (ideally weakest possible) constraints on the operating conditions under which a correctness guarantee is valid. A 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). Fig. 9 shows 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.

Assuming both the continuous and non-continuous constraints hold for a given training data/runtime signal energy pair, the goal is to prove that the model always correctly classifies that runtime signal energy. The proof relies on some extensional properties of the model function m'. Here, m' is treated 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 airspeed/angle of attack configurations correspond to stall (determined using physical properties, or even by observation), and therefore, a certain subset of signal energy means and variances $S(T) \subseteq D(T)$ corresponds 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 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, 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 it is known that the experimental data follows a normal distribution, it is also known that each distribution in $D(T)$ is normal. The goal is to prove that for all signal energies in a given interval, the model behaves in a predictable way. Formally, one can express a proposition for the no-stall classification as

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

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

$$(\forall d \in D(T) : \mathcal{N}(d)) \wedge (\exists d \in D(T) \setminus S(T) : f(x, d)) \wedge (\forall d \in S(T) : \neg f(x, d))$$

where $\mathcal{N}(d)$ is a predicate that returns `True` if the distribution d is Gaussian in nature and $f(x, d)$ is a predicate that returns `True` if the signal energy x is likely to belong to the distribution d, given the mean and standard deviation of d. Analogously, the corresponding proposition for the stall classification is similar except for an inversion of the roles of $S(T)$ and $D(T) \setminus S(T)$:

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

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

$$(\forall d \in D(T) : \mathcal{N}(d)) \wedge (\exists d \in S(T) : f(x, d)) \wedge (\forall d \in D(T) \setminus S(T) : \neg f(x, d))$$

As an example, to check if x lies within 2 standard deviations from the mean of the distribution d, the function $f(x, d)$ can be defined as follows:

$$f(x, d) \equiv |x - \mu(d)| \leq 2\sigma(d)$$

where $\mu(d)$ and $\sigma(d)$ are the mean and standard deviation of d, respectively. Therefore, for the Safe subset for no-stall classification, the required condition is:

$$\begin{aligned}&(\forall(d \in D(T)) : \mathcal{N}(d))\\ &\wedge (\exists(d \in D(T) \setminus S(T)) : |x - \mu(d)| \leq 2\sigma(d))\\ &\wedge (\forall(d \in S(T)) : |x - \mu(d)| > 2\sigma(d))\end{aligned}$$

Similarly, for the Safe$'$ subset for stall classification, the required condition is:

$$\begin{aligned}&(\forall(d \in D(T)) : \mathcal{N}(d))\\ &\wedge (\exists(d \in S(T)) : |x - \mu(d)| \leq 2\sigma(d))\\ &\wedge (\forall(d \in D(T) \setminus S(T)) : |x - \mu(d)| > 2\sigma(d))\end{aligned}$$

The formal safety envelope $\mathbb{S}$ can now be defined as the union of these subsets:

$$\mathbb{S} = \text{Safe} \cup \text{Safe}'$$

The proofs of these rely on the previously expressed properties of m': one can simply compute the *cumulative distribution function* (CDF) of each Gaussian distribution (known from the first assumption in the safe subsets) given minimum/maximum values of x.

3.2 A formal progress envelope

In the conflict-aware flight planning application described in Section 2.3, after an ownship computes a safe set of flight-plans, there needs to be a *consensus*

regarding that safe set among a subset of participants in the IoP [50]. Consensus is required because the algorithm for generating a conflict-free flight-plan for an ownship assumes that the traffic aircraft do not deviate from their flight-plans. If the set of traffic flight-plans changes, then the previously computed flight-plan for the ownship may no longer be conflict-free and a new flight-plan must be computed by considering the new traffic flight-plans.

Consensus algorithms are widely used in distributed lock services [10], cryptocurrency networks [63], smart power grids [62], etc. The problem of achieving consensus among a network of aircraft, however, is not trivial. The challenges include, but are not limited to: (1) *the dynamic nature of IoP* - aircraft may temporarily become disconnected from ground-stations and other aircraft; (2) *limitations of the network* - message loss and message propagation delays may lead to deadlocks and livelocks in consensus algorithms; (3) *lack of a global clock* - makes it difficult to determine the total ordering of events across multiple systems; and (4) *asynchronicity of the network* - unknown message propagation time makes it difficult to differentiate node failures from message delays. Consensus algorithms like *Paxos* [35] solve the problem of distributed consensus by electing a leader and guaranteeing the properties of consistency and progress under the assumption of a single leader [19]. However, this leader-follower method creates a communication bottleneck and makes the leader a unique point of failure for progress. The *Synod* algorithm [34], which can guarantee both consistency and progress (under certain conditions) in the absence of a unique leader, is, therefore, better suited for achieving consensus in the IoP.

The Synod protocol assumes an asynchronous, non-Byzantine system model in which agents operate at arbitrary speed, may fail and restart, and have stable storage. In this system model, messages can be duplicated, lost, and have arbitrary transmission times, but cannot be corrupted [35]. There are two logically separate sets of agents: *proposers* - the set of agents that can propose values to be chosen; *acceptors* - the set of agents that can vote on which value should be chosen; and *learners* - the set of agents that learn when a value is chosen by a majority of acceptors. The proposers choose a number called the *proposal number* which is used by the acceptors to vote on values. It can be formally proven that the Synod algorithm makes progress under the following conservative conditions:

- *some proposal number successfully completes both phases of the protocol,*
- *enough agents are non-faulty, i.e., they perform their expected actions, and*
- *all messages sent by non-faulty agents are eventually delivered.*

Progress envelopes for data-driven systems are defined as computable subsets of the system state space in which progress can be guaranteed [48]. For the Synod algorithm, the conditions affected by network uncertainties (particularly the conditions that enough agents are non-faulty and all messages sent by non-faulty agents are eventually delivered) can be definitively quantified, measured, and used to classify the network characteristics into distinct subsets where progress guarantees may or may not hold.

To illustrate an example of how progress envelopes may be used, consider the case of the Synod algorithm where the network transmits data using *Automatic*

Dependent Surveillance-Broadcast (ADS-B). ADS-B uses radio signals for communication, the propagation of which is affected by the *total electron content* (TEC) in the atmosphere [60]. Also represent the TEC of the atmosphere by τ. If all other factors affecting propagation delay can be represented by a constant K, then one can represent propagation delay Δ_t using a function Φ of TEC.

$$\Delta_t = \Phi(\tau, K)$$

Also represent the set of proposers by $\mathcal{P}$, the set of acceptors by $\mathcal{A}$, the set of learners by $\mathcal{L}$, the set of all possible TEC by Γ, the set of all points in time as $\mathcal{T}$, and the predicate that checks the availability of an agent x at any given time t by $\alpha(x, t)$. It is possible to quantify and measure both $\Phi(\tau)$ and $\alpha(x, t)$. Now, the data-driven progress envelope for the Synod algorithm can be expressed as the set of network characteristics satisfying the following:

$$\forall \tau \in \Gamma : \big(\Phi(\tau, K) < \infty\big) \wedge \forall t \in \mathcal{T}, p \in \mathcal{P}, a \in \mathcal{A}, l \in \mathcal{L} : \big(\alpha(p, t) \wedge \alpha(a, t) \wedge \alpha(l, t)\big)$$

The above envelope is a conjunction of the conditions that affect eventual message delivery and availability of all agents at all times.

3.3 Correctness sentinels

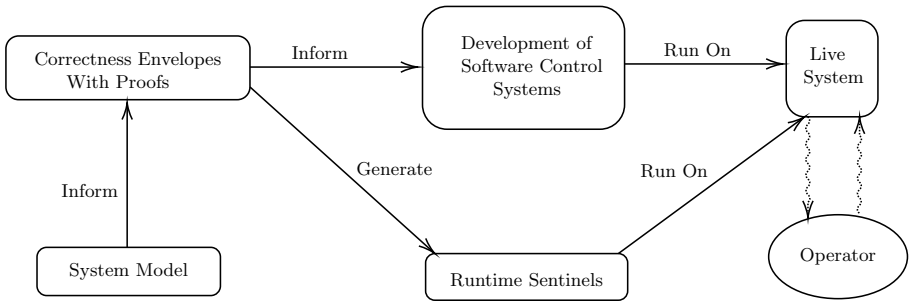

Fig. 10: Workflow for verification with runtime sentinels.

A correctness sentinel is a runtime-accessible program that can monitor real-time data-streams to detect if the data satisfies correctness envelope constraints [9, 13]. The ability to automatically generate sentinels from proof-accessible domain-specific languages is desirable as it allows the correctness of the sentinels to be guaranteed just like the code generation module of a verified compiler [36]. Integrating sentinels as a part of the larger data-driven system provides the ability to dynamically monitor the system state for runtime-awareness of which formal properties can be guaranteed (Fig. 10) at any time.

It should be noted that the sentinels may not correspond exactly with the formal assumption. For example, a test for normality up to some significance level

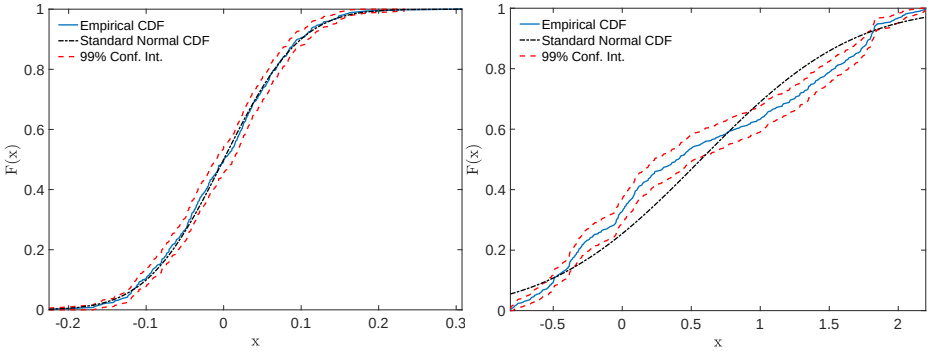

Fig. 11: Normality visualization for preprocessed (left) and raw (right) data.

does not imply that the data is normal. Furthermore, preprocessing steps like filtering and downsampling can have significant effects on normality assumptions (Fig. 11). Additionally, floating-point arithmetic is not the same as arithmetic over real numbers, making it difficult to reason about it in a formal setting. Therefore, the sentinels, for now, provide only a useful estimate of the assumption validity at runtime.

4 Experiments and Results

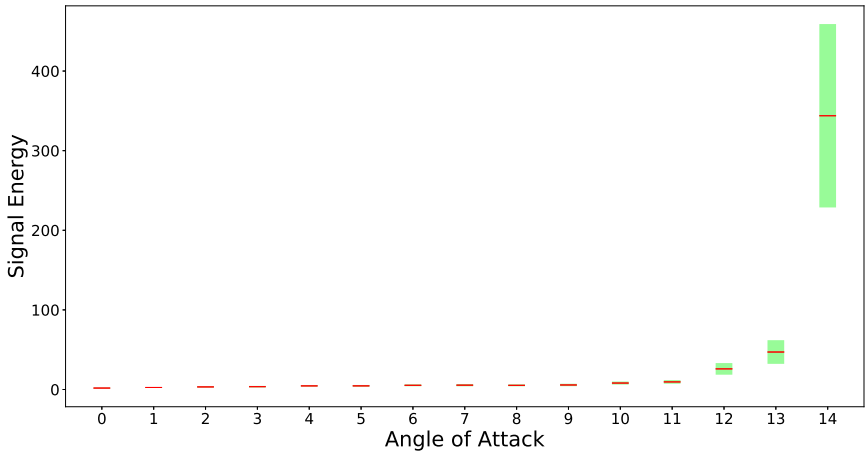

Fig. 12: Signal energy from sensor 3 for a constant airspeed of 15 m/s.

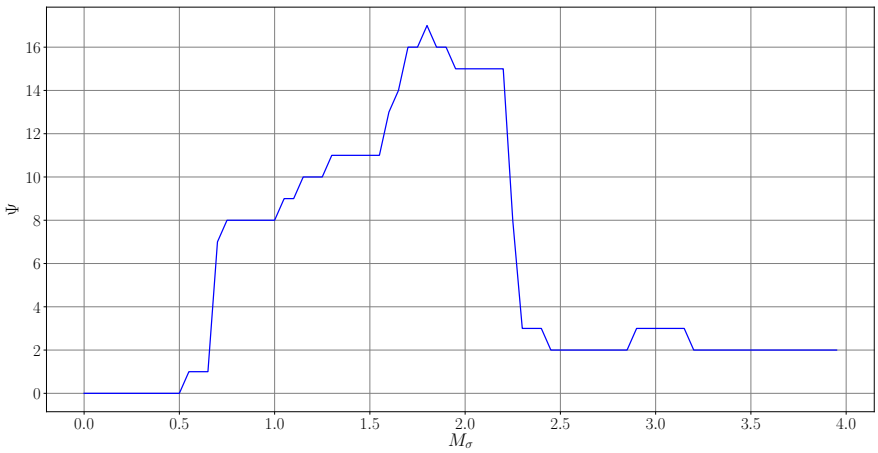

Fig. 13: The envelope metric Ψ for varying values of the M_σ multiplier.

To verify the effectiveness of the correctness envelopes and correctness envelope sentinels, safety envelope sentinels have been implemented for the stochastic state awareness application of Section 3.1.

Data was collected from 8 piezoelectric sensors along a prototype composite wing which was tested in an open-loop low-turbulence wind tunnel that has a square test section of 0.84 m $\times$ 0.84 m and can achieve continuous flow speeds of approximately 30 m/s [32]. A series of wind tunnel experiments were conducted by varying the AoA (discrete values from 0° to 17°) and airspeed (discrete values from 6 m/s to 22 m/s). For each AoA, data for 91 seconds was collected with a sampling frequency of 1000 samples per second. For stall detection, a constant airspeed of 15 m/s was considered (Fig. 12) and the corresponding signal energy values for each AoA was split into windows of 1000 samples. The mean and standard deviation of the distribution corresponding to each window was calculated and used to form the set $D(T)$ described in Section 3.1.

The sentinel, which was written in the Python programming language [55], checked the signal energy values to determine if they were within the Safe/Safe$'$ envelopes, under the assumption that the distribution d is Gaussian in nature. To capture the effect of varying levels of confidence on the safety envelopes, the value of the sigma multiplier M_σ was varied from 0 to 4 with a step-size of 0.5 in the experiments and an envelope metric Ψ was introduced to compare the envelope for each value of M_σ. Ψ was defined as the total number of values of AoA corresponding to which there were no signal energy values which fell outside the envelope. Formally, one can interpret Ψ as:

$$\Psi = |\{\theta \in \mathbb{A} : (\forall x \in \mathbb{X}(\theta) : x \in \mathbb{S}(M_\sigma))\}|$$

where $\mathbb{A}$ is the set of all AoA, $\mathbb{X}(\theta)$ is the set of all signal energies corresponding to the AoA value θ, and $|\ |$ represents set cardinality.

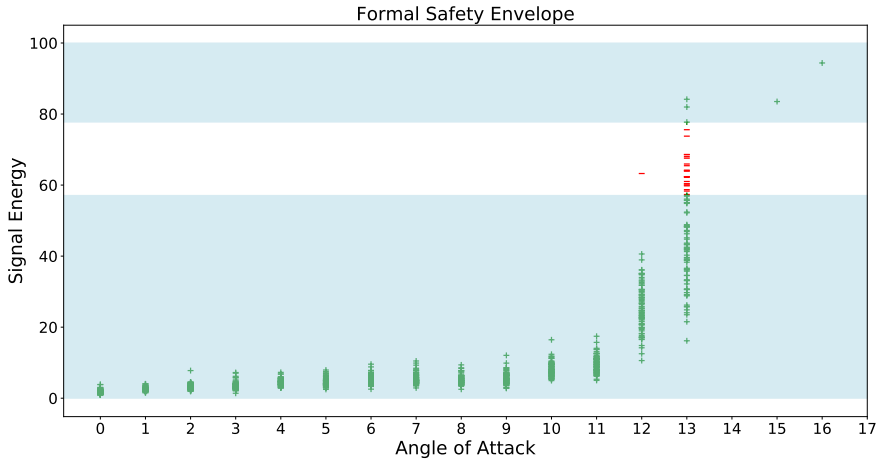

Fig. 14: A portion of the safety envelope (shaded region) detected from the input signals for $M_\sigma = 2$ ($+$ = `True`, $-$ = `False`).

From Fig. 13, which shows the variation of Ψ with varying values of M_σ, it is clear that the current model for stall detection works best with $M_\sigma = 1.8$, giving the best value of Ψ for the safety envelope. In this case, 17 of the 18 values of AoA considered for the experiment, fall completely under the safety envelope, resulting in the most flexible envelope for the different values of M_σ considered in the experiment. The outputs of the sentinel corresponding to the input signal energies makes it possible to clearly split the range of signal energies into separate regions where the safety envelope holds or does not hold. Fig. 14 represents a portion of the safety envelope detected by the sentinels for $M_\sigma = 2$. The smaller values of Ψ correspond to higher confidence levels and indicate that the current model is not very efficient and there is a need for more robust and advanced methodologies. However, these results clearly show the effectiveness of this approach in detecting the Safe and Safe$'$ subsets from runtime data, even though there is room for improvement.

5 Related Work

Typically, verification efforts focused on hardware and software systems have been primarily concerned with applying automated methods, for example, model checking [56]. More recently, however, there has been interest in applying interactive or human-guided techniques. For example, the VeriDrone project [54] builds upon existing work using differential dynamic logic [52] to verify properties of hybrid systems [17] using an interactive theorem-proving system. There is existing work on integrating automated tools with interactive theorem-provers [3, 6, 16, 38], allowing the use of a theorem-prover for high-level organization and proofs of difficult lemmas while leaving the mundane and well-understood work to au-

tomated tools. Previous work on runtime monitors for verified systems exist. Mitsch *et al.* [39] have proposed "Modelplex", which analyzes a real system implementation during runtime to monitor if the system is in compliance with the verified models. Pike *et al.* [51] have investigated the applications of "Copilot", a language and compiler designed for generating monitors for distributed, real-time systems. They generate C [27] runtime monitors from specifications written in a Haskell embedded domain-specific language [59] and can verify code-generation backends using CompCert [37] and CBMC [33].

Formal progress properties for consensus protocols have been previously investigated. Hawblitzel *et al.* [19] introduce IronFleet, a framework for proving progress and consistency properties of distributed systems. Konnov *et al.* [29] propose a model checking framework for verifying the safety and progress property of distributed algorithms like Paxos. Dragoi *et al.* [15] introduce PSYNC, for the writing, execution, and verification of distributed algorithms.

A multifidelity approach for DDDAS, that draws upon information from multiple modelling and sensing options with varying levels of fidelity, has been used for providing stochastic predictions for real-time decision-support systems [1]. It uses a resource-allocation procedure that supports decision-making about when and what to measure from sensors, which model to use, and what are the current quantities of interest. From the perspective of FCEs, each model may be associated with a different envelope for the same system property, where the quality of the envelope (weak/strong) is proportional to the fidelity level of the model.

The work presented in this chapter improves upon the state of the art by defining correctness envelopes and extending runtime monitors from detecting whether all properties of a system hold to identifying what properties of a system can be guaranteed under the current operating conditions. The formal correctness envelope (FCE) approach allows for providing probabilistic guarantees of system properties which can be associated with confidence values if desired.

6 Conclusion

This chapter has presented the concept of formal correctness envelopes (FCE) for dynamic data-driven aerospace systems. Correctness envelopes enable data-driven systems to analyze if a particular correctness guarantee holds for a given system state. A class of runtime accessible programs called correctness sentinels, which can monitor system state in real-time and detect if the state satisfies the envelope constraints, has also been presented. The chapter also discusses the wide-ranging applications of DDDAS in smart aerospace systems of the future and the scope of formal verification of such safety-critical systems using FCEs.

It should, however, be noted that correctness envelopes for a particular property can vary from weak to strong. For any given property, a stronger envelope imposes more stringent restrictions on the system than a weaker envelope. It is usually desirable to ensure that the envelopes are the weakest possible so that the corresponding guarantees are tolerant to more variations in the operating conditions of the system. It may be possible to drive the decision-making pro-

cedure of the multifidelity model approach for DDDAS (discussed in Section 5) based on the desired quality of envelopes for a given set of system properties.

A formidable challenge is to generate correct sentinels from the system specifications, as discussed in Section 3.3. Therefore, an important topic for future research is to investigate efficient procedures to connect the formal proof development with correctness sentinels. The stochastic nature of data-driven aerospace systems also implies that certain system properties can not only be expressed as binary values but can also be expressed by continuous values with associated confidence levels. Hence, another potential direction of future research is the development of non-binary envelopes so that the corresponding sentinels can analyze real-time data-streams and provide stochastic information about the status of the system with respect to the correctness proofs during runtime. Investigating the integration of the FCE approach with the multifidelity approach for DDDAS to facilitate envelope-dependent decision-making considerations for resource allocation is another interesting future direction of work.

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. Allaire, D., Biros, G., Chambers, J., Ghattas, O., Kordonowy, D., Willcox, K.: Dynamic data driven methods for self-aware aerospace vehicles. Procedia Computer Science **9**, 1206–1210 (2012)
2. Arkoudas, K.: Athena, `http://proofcentral.org/athena`
3. Armand, M., Faure, G., Grégoire, B., Keller, C., Théry, L., Werner, B.: A modular integration of sat/smt solvers to coq through proof witnesses. In: International Conference on Certified Programs and Proofs. pp. 135–150. Springer (2011)
4. Barras, B., Boutin, S., Cornes, C., Courant, J., Filliatre, J.C., Gimenez, E., Herbelin, H., Huet, G., Munoz, C., Murthy, C., et al.: The Coq Proof Assistant Reference Manual: Version 6.1 (1997)
5. Barrett, C., Tinelli, C.: Satisfiability modulo theories. In: Handbook of Model Checking, pp. 305–343. Springer (2018)
6. Blanchette, J.C., Böhme, S., Paulson, L.C.: Extending sledgehammer with smt solvers. In: International Conference on Automated Deduction. pp. 116–130. Springer (2011)
7. Blasch, E., Bossé, É., Lambert, D.A.: High-level information fusion management and systems design. Artech House (2012)
8. Boldo, S., Lelay, C., Melquiond, G.: Coquelicot: A user-friendly library of real analysis for Coq. Mathematics in Computer Science **9**(1), 41–62 (2015)
9. 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)
10. Burrows, M.: The chubby lock service for loosely-coupled distributed systems. In: Proceedings of the 7th symposium on Operating systems design and implementation. pp. 335–350. USENIX Association (2006)

11. Carter, M., Van Brunt, B.: The lebesgue-stieltjes integral. In: The Lebesgue-Stieltjes Integral, pp. 49–70. Springer (2000)
12. Chaudhuri, K., Doligez, D., Lamport, L., Merz, S.: Verifying Safety Properties with the TLA+ Proof System. In: International Joint Conference on Automated Reasoning. pp. 142–148. Springer (2010)
13. Cruz-Camacho, E., Paul, S., Kopsaftopoulos, F., Varela, C.A.: Towards provably correct probabilistic flight systems. In: Darema, F., Blasch, E., Ravela, S., Aved, A. (eds.) Dynamic Data Driven Application Systems. pp. 236–244. Springer International Publishing, Cham (2020)
14. 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)
15. Drăgoi, C., Henzinger, T.A., Zufferey, D.: PSync: A Partially Synchronous Language for Fault-Tolerant Distributed Algorithms. In: ACM SIGPLAN Notices. vol. 51, pp. 400–415. ACM (2016)
16. Fontaine, P., Marion, J.Y., Merz, S., Nieto, L.P., Tiu, A.: 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. pp. 167–181. Springer (2006)
17. Ghorbal, K., Jeannin, J.B., Zawadzki, E., Platzer, A., Gordon, G.J., Capell, P.: Hybrid theorem proving of aerospace systems: Applications and challenges. Journal of Aerospace Information Systems **11**(10), 702–713 (2014)
18. Halmos, P.R.: Measure theory, vol. 18. Springer (2013)
19. Hawblitzel, C., Howell, J., Kapritsos, M., Lorch, J.R., Parno, B., Roberts, M.L., Setty, S., Zill, B.: IronFleet: Proving Safety and Liveness of Practical Distributed Systems. Communications of the ACM **60**(7), 83–92 (2017)
20. Imai, S., Blasch, E., Galli, A., Zhu, W., Lee, F., Varela, C.A.: Airplane flight safety using error-tolerant data stream processing. IEEE Aerospace and Electronic Systems Magazine **32**(4), 4–17 (2017)
21. Imai, S., Chen, S., Zhu, W., Varela, C.A.: Dynamic data-driven learning for self-healing avionics. Cluster Computing **20**, 1–24 (Nov 2017)
22. Imai, S., Galli, A., Varela, C.A.: Dynamic data-driven avionics systems: Inferring failure modes from data streams. In: Dynamic Data-Driven Application Systems (DDDAS 2015). pp. 1665–1674. Reykjavik, Iceland (Jun 2015)
23. Imai, S., Varela, C.A.: A programming model for spatio-temporal data streaming applications. In: Dynamic Data-Driven Application Systems (DDDAS 2012). pp. 1139–1148. Omaha, Nebraska (Jun 2012)
24. Insaurralde, C.C., Costa, P.C., Blasch, E., Sampigethaya, K.: Uncertainty considerations for ontological decision-making support in avionics analytics. In: 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC). pp. 1–9. IEEE (2018)
25. James, A., Kopsaftopoulos, F.: Data-driven stochastic identification of a uav under varying flight and structural health states. Structural Health Monitoring 2019 (2019)
26. Kechris, A.: Classical descriptive set theory, vol. 156. Springer Science & Business Media (2012)
27. Kernighan, B.W., Ritchie, D.M.: The C Programming Language (2006)
28. Klockowski, R.S., Imai, S., Rice, C., Varela, C.A.: Autonomous data error detection and recovery in streaming applications. In: Proceedings of the International

Conference on Computational Science (ICCS 2013). Dynamic Data-Driven Application Systems (DDDAS 2013) Workshop. pp. 2036–2045. Barcelona, Spain (May 2013)
29. Konnov, I., Lazić, M., Veith, H., Widder, J.: A Short Counterexample Property for Safety and Liveness Verification of Fault-tolerant Distributed Algorithms. In: ACM SIGPLAN Notices. vol. 52, pp. 719–734. ACM (2017)
30. 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. p. 1534 (2019)
31. Kopsaftopoulos, F., Chang, F.K.: 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, pp. 697–721. Springer (2018)
32. Kopsaftopoulos, F., Nardari, R., Li, Y.H., Chang, F.: Data-driven state awareness for fly-by-feel aerial vehicles: Experimental assessment of a non-parametric probabilistic stall detection approach. Structural Health Monitoring 2017 (shm) (2017)
33. Kroening, D., Tautschnig, M.: Cbmc–c bounded model checker. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 389–391. Springer (2014)
34. Lamport, L.: The Part-Time Parliament. ACM Transactions on Computer Systems (TOCS) **16**(2), 133–169 (1998)
35. Lamport, L.: Paxos Made Simple. ACM Sigact News **32**(4), 18–25 (2001)
36. Leroy, X.: Formal verification of a realistic compiler. Communications of the ACM **52**(7), 107–115 (2009)
37. Leroy, X., et al.: The compcert verified compiler. Documentation and user's manual. INRIA Paris-Rocquencourt (2012)
38. Merz, S., Vanzetto, H.: Automatic verification of TLA+ proof obligations with SMT solvers. In: International Conference on Logic for Programming Artificial Intelligence and Reasoning. pp. 289–303. Springer (2012)
39. 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)
40. Murphy, K.P.: Conjugate bayesian analysis of the gaussian distribution. def **1**($2\sigma 2$), 16 (2007)
41. Narkawicz, A., Munoz, C., Dutle, A.: Coordination logic for repulsive resolution maneuvers. In: 16th AIAA Aviation Technol., Integr., and Operat. Conf. p. 3156 (2016)
42. Narkawicz, A., Muñoz, C., Dutle, A.: Sensor uncertainty mitigation and dynamic well clear volumes in daidalus. In: 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC). pp. 1–8. IEEE (2018)
43. Noble, B., Daniel, J.W., et al.: Applied linear algebra, vol. 3. Prentice-Hall Englewood Cliffs, NJ (1977)
44. Padon, O., McMillan, K.L., Panda, A., Sagiv, M., Shoham, S.: Ivy: Safety Verification by Interactive Generalization. ACM SIGPLAN Notices **51**(6), 614–630 (2016)
45. Paul, S.: Emergency Trajectory Generation for Fixed-Wing Aircraft. Master's thesis, Rensselaer Polytechnic Institute (Dec 2018)
46. Paul, S., Hole, F., Zytek, A., Varela, C.A.: Flight trajectory planning for fixed wing aircraft in loss of thrust emergencies. In: Dynamic Data-Driven Application Systems (DDDAS 2017). Cambridge, MA (Aug 2017)

47. 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 (DASC 2018). pp. 558–567. London, England (Sep 2018)
48. Paul, S., Kopsaftopoulos, F., Patterson, S., Varela, C.A.: Dynamic data-driven formal progress envelopes for distributed algorithms. In: Darema, F., Blasch, E., Ravela, S., Aved, A. (eds.) Dynamic Data Driven Application Systems. pp. 245–252. Springer International Publishing, Cham (2020)
49. 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 (DASC 2019). San Diego, CA (Sep 2019)
50. Paul, S., Patterson, S., Varela, C.A.: Collaborative situational awareness for conflict-aware flight planning. In: The 39th AIAA/IEEE Digital Avionics Systems Conference (2020)
51. 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)
52. Platzer, A.: Differential dynamic logic for hybrid systems. Journal of Automated Reasoning **41**(2), 143–189 (2008)
53. Reilly, I.L., Subrahmanyam, P., Vamanamurthy, M.: Cauchy sequences in quasi-pseudo-metric spaces. Monatshefte für Mathematik **93**(2), 127–140 (1982)
54. Ricketts, D., Malecha, G., Alvarez, M.M., Gowda, V., Lerner, S.: Towards verification of hybrid systems in a foundational proof assistant. In: Formal Methods and Models for Codesign (MEMOCODE), 2015 ACM/IEEE International Conference on. pp. 248–257. IEEE (2015)
55. Sanner, M.F., et al.: Python: a programming language for software integration and development. Journal of Molecular Graphics and Modelling **17**(1), 57–61 (1999)
56. Sen, K., Viswanathan, M., Agha, G.: Statistical model checking of black-box probabilistic systems. In: International Conference on Computer Aided Verification. pp. 202–215. Springer (2004)
57. Soare, R.I.: Recursion theory and dedekind cuts. Transactions of the American Mathematical Society **140**, 271–294 (1969)
58. Sommerville, I.: Software engineering. Addison-Wesley/Pearson (2011)
59. Thompson, S.: Haskell: the craft of functional programming, vol. 2. Addison-Wesley (2011)
60. Van Der Pryt, R., Vincent, R.: A simulation of the reception of automatic dependent surveillance-broadcast signals in low earth orbit. International Journal of Navigation and Observation (2015)
61. Willard, S.: General topology. Courier Corporation (2012)
62. Yang, S., Tan, S., Xu, J.X.: Consensus based approach for economic dispatch problem in a smart grid. IEEE Transactions on Power Systems **28**(4), 4416–4426 (2013)
63. Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H.: An overview of blockchain technology: Architecture, consensus, and future trends. In: 2017 IEEE International Congress on Big Data (BigData Congress). pp. 557–564. IEEE (2017)