Runtime Assurance for Safety-Critical Systems

AN INTRODUCTION TO SAFETY FILTERING APPROACHES FOR COMPLEX CONTROL SYSTEMS

KERIANNE L. HOBBS, MARK L. MOTE, MATTHEW C.L. ABATE, SAMUEL D. COOGAN, and ERIC M. FERON

More than three miles above the Arizona desert, an F-16 student pilot experienced a gravity-induced loss of consciousness, passing out while turning at nearly 9Gs (nine times the force of gravity), flying over 400 kn (over 460 mi/h). With its pilot unconscious, the aircraft turn devolved into a dive, dropping from over 17,000 ft to lower than 8,000 ft in altitude in less than 10 s. An auditory warning in the cockpit called out to the pilot "altitude, altitude" just before he crossed through 11,000 ft, switching to a command to "pull up" around 8,000 ft. Meanwhile, the student's instructor was watching the event unfold from his own aircraft. As the student's aircraft passed through 12,500 ft, the instructor called over the radio "two recover," commanding the student ("two") to end the dive. As the student's aircraft passed through 11,000 ft, the instructor's "two recover!" came with increased urgency. At 9,000 ft, and with terror rising in his voice, the instructor yelled "TWO RECOVER!" Fortunately, at the same time as the instructor's third panicked radio call, a new runtime assurance (RTA) system kicked in to automatically recover the aircraft. The Automatic Ground Collision Avoidance System (Auto GCAS), an RTA system integrated on the jets fewer than two years earlier, in fall 2014, detected that the aircraft was about to collide, commanded a roll to wings level and pull-up maneuver, and recovered the aircraft fewer than 3,000 ft above the ground. The event described here occurred in May 2016. A video from the event was declassified and publicly released in September 2016, and the footage can be found at [1]. While Auto GCAS monitored the behavior of a safety-critical cyberphysical system with

Cockpit: 2nd Lt. Ryan Collins demonstrates an automatic fly up maneuver generated by the Automatic Ground Collision Avoidance System in a research simulator at the Air Force Research Laboratory at Wright-Patterson Air Force Base.

Digital Object Identifier 10.1109/MCS.2023.3234380
Date of current version: 24 March 2023

1066-033X/23©2023IEEE

a human providing the primary control functions, the same concept is gaining attention in the autonomy community looking to assure safety while integrating complex and intelligent control system designs.

RTA systems are online verification mechanisms that filter an unverified *primary* controller output to ensure system safety (see "Summary"). The primary control may come from a human operator, an advanced control approach, and an autonomous control approach that cannot be verified to the same level as simpler control systems designs. The critical feature of RTA systems is their ability to alter unsafe control inputs explicitly to assure safety (see "Runtime Assurance"). In many cases, RTA systems can functionally be described as containing a *monitor* that watches the state of the system and output of a *primary controller* and *backup controller* that replaces and modifies control input when necessary to assure safety. Note that RTA and the controllers within the architecture go by many different names, as described in "Runtime Assurance Aliases." RTA designs specifically to bound the behavior of neural network control systems used as the primary controller are discussed in "Shielded Learning." An important quality of an RTA system is that the assurance mechanism is constructed in a way that is entirely agnostic to the underlying structure of the primary controller. By effectively decoupling the enforcement of safety constraints from performance-related objectives, RTA offers a number of useful advantages over traditional (offline) verification. Another way to think of RTA is to consider designing controllers so that there is always a plan B, as discussed in "The Case for Plan B."

Summary

This article provides an introductory tutorial on runtime assurance (RTA) concepts and their role in ensuring the safety of complex control systems. Assuming an undergraduate-level understanding of control theory and state-space concepts, this article provides theory and examples for RTA architectures, four types of RTA approaches, and a number of systems engineering and practical considerations for employing RTA in safety-critical cyber-physical systems.

RICHARD ELDRIDGE, AIR FORCE RESEARCH LABORATORY

Verification, validation, assurance, and certification methods present the largest barriers to the operational use of autonomous control in safety-critical systems, such as passenger aircraft, vehicles, medical devices, and nuclear power plants. For example, the commercial aviation domain requires showing stringent safety constraint satisfaction for any failure that is more likely than "extremely improbable," defined as an event that occurs with a probability of 10^{-9} [2], (more than one in 1 billion flight hours). While verification, validation, assurance, and certification are related, there are slight variations in their meaning. Additionally, these are also confused with concepts described in "Safety, Reliability, and Security." *Verification* is an activity that determines whether a system meets requirements [3], in effect answering the question, "Did we build the system right?" *Validation* assesses whether a system meets the needs of the end user [4], answering the question, "Did we build the right system?" *Model validation* refers to evaluating how well a model represents reality. *Assurance* is justified confidence that the system functions as intended with limited vulnerability to uncertainty, hazards, and threats based on evidence generated through development activities [5]. *Certification* determines whether a system conforms to a set of criteria or standards for a class of similar systems [6], [7], [8], [9], [10], [11], [12], [13]. The verification, validation, certification, and assurance of safety-critical dynamical systems requires the development of techniques that incorporate

comprehensive analysis based on rigorously developed specifications as well as techniques that continuously evaluate system behavior at runtime.

A significant benefit of RTA is that it alleviates the need to design the primary controller in a way that conforms to traditional safety standards, which may not be directly applicable to the primary control design. Practical consequences of this are that RTA provides a means of testing new control algorithms on hardware platforms, without compromising safety potential, as well as a *near-term certification path* for autonomous controllers and safety-critical human-controlled systems. Additionally, the verification of an assurance mechanism is generally simpler than the verification of a performance-based controller, as it does not require consideration of the potentially complex performance-related objectives. That is, safety verification does not become more difficult as the primary controller grows more complex, and the verification

Runtime Assurance

Runtime assurance (RTA) systems guarantee the safety of increasingly complex and intelligent control system designs by monitoring the state of the system and intervening when necessary. RTA acts as a safety filter that sits between the primary controller and plant to assure that the state of the system adheres to safety properties. The design of the RTA safety mechanism is decoupled from the design of the primary controller, allowing RTA to focus on safety while the primary controller optimizes performance. Historically, verification techniques for novel control approaches trail years to decades behind the development of the approaches themselves [66]. RTA provides a path to introduce the use of these advanced controllers sooner, even in safety-critical applications. This article provides a tutorial on many ways to design RTA, including explicit and implicit monitoring, Simplex and active set invariance filter (ASIF) intervention approaches, and uncertainty, with a variety of examples. Explicit monitors define forward invariant and control invariant safe sets offline that are enforced online at runtime, while implicit monitors project the finite time trajectory of a backup controller online. Simplex interventions switch from the primary controller to a backup controller, while ASIF interventions gradually and optimally modify the control signal, subject to safety constraints. Conceptual and mathematical descriptions are provided for each. Where possible, the Automatic Ground Collision Avoidance System, an implicit Simplex RTA design, is used as an example to illustrate several RTA concepts. The article assumes an introductory understanding of control theory and state-space concepts.

Runtime Assurance Aliases

Runtime assurance (RTA) provides a conceptually simple and highly effective approach to enforcing safety constraints that has frequently been explored, utilized, and independently reinvented in a wide variety of areas. Thus, RTA and its applications go by many names in the literature, including active set invariance, active collision avoidance, advanced driver assistance, active safety, safety filtering, emergency braking, sandboxing, conflict resolution, and constraint-based planning. The primary controller goes by several names in the literature, including advanced controller [S1], advanced system [41], experimental control module [152], baseline controller [152], unverified controller [S2], [174], nominal controller [90], complex subsystem [S2], [154], and complex controller [60]. Finally, the backup controller goes by many names in the literature, including recovery mechanism, reversionary controller [S1], reversionary system [41], safety control module [152], verified controller [174], backup controller [90], safety remediation controller [154], recovery controller [S2], [61], [108], [159], and safety controller [60].

Part of the different naming conventions comes from the different intentions and resulting implementation of the RTA intervention, which are grouped here as recovery and reversionary controllers. *Recovery controllers* intervene in emergency situations and return control authority to the primary controller as soon as possible. Recovery controllers are not intended to perform the wide range of tasks a dynamical system is designed to perform but instead focus solely on recovery maneuvers from an unsafe condition (e.g., collision avoidance, adherence to keep-out zones, and maintaining state limits). By contrast, *reversionary controllers* detect inappropriate behavior of the primary controller and revert to a simpler verified controller to complete the rest of the tasks. The reversionary controller may sacrifice performance for assurance. This article focuses on the design of recovery controller-based RTA systems.

REFERENCES

[S1] J. D. Schierman, D. Neal, E. Wong, and A. K. Chicatelli, "Runtime assurance protection for advanced turbofan engine control," in *Proc. AIAA Guid., Navig., Contr. Conf.*, 2018, p. 1112, doi: 10.2514/6.2018-1112.

[S2] M. Clark et al., "A study on run time assurance for complex cyber physical systems," *Air Force Res. Lab. Aerosp. Syst. Directorate*, Defense Tech. Inf. Center, Fort Belvoir, VA, USA, Tech. Rep. ADA585474, 2013.

process does not need to be repeated when changes are made to the primary controller. Note that RTA is envisioned to increase safety confidence, not provide a substitute for having some level of confidence in the safety of a primary controller. Additionally, as described in "To Use Runtime Assurance or Not to Use Runtime Assurance? That Is the Question," there are times when RTA is not appropriate.

RTA is emerging as the dominant approach for enforcing safety in real-world autonomous systems and semiautonomous systems. Although not always directly associated with the term *RTA*, notable autonomous systems applications are seen in road vehicles [14], [15], bipedal walking [16], [17], air traffic control [18], fixed-wing aircraft collision avoidance [19], [20], [21], [22], [23] and flight envelope protection [24], vertical-takeoff-and-landing (VTOL) aircraft [25], [26], spacecraft collision avoidance [27], manipulator arms [28], and propulsion [29], to name a few.

The rise in popularity of RTA can be attributed to many factors, including 1) the emergence of mobile autonomous systems operating in safety-critical environments, e.g., away from very specialized and remote applications to

Shielded Learning

No discussion of runtime assurance (RTA) for complex and autonomous control systems would be complete without discussing the emerging challenges of using machine learning and, in particular, reinforcement learning (RL), where a dynamical system learns from experience collected over many episodes [S3]. The use of RL has attracted attention for its ability to tackle complex tasks with high-dimensional state spaces from world experts in Go [S4], [S5] and *StarCraft II* [S6]. In addition, it is becoming popular in robotics research to operate in unknown environments and learn state representations for many tasks [S7]. Both control theory and RL share a common concept of a system *state* (usually, $x \in \mathcal{X}$ in control theory and $s \in \mathcal{S}$ in RL). A control input ($u \in \mathcal{U}$ in control theory) is formulated as an *action a*, or $A \in \mathcal{A}$, in RL, and the actions are determined by a *policy* (i.e., controller) that maps states to actions. State action sequences $(S_0, A_0, S_1, A_1, \ldots S_n, A_n)$ are called *trajectories* in RL. Both fields have a notion of partial observability. A measurement *y* may come from sensors in control theory, and an *observation O* in RL may be used to estimate the system state (for example, $\hat{x} \in \mathcal{X}$). In control theory, a controller is optimized to minimize a cost function, while in RL, a policy is optimized to maximize a *reward function*. In both control theory and RL, *dynamics* describe how the state evolves in the environment. RL approaches may be *model-based*, where the agent learns a model of the environment, and *model-free*, where dynamics are not explicitly modeled and the action is learned directly through interactions with the environment.

A challenge in RL, especially when it takes place on a physical agent, is ensuring the safety of that agent. Safe RL is defined as the process of learning a policy that maximizes the expected return while ensuring adherence to safety constraints [34]. Safe RL usually employs either *reward shaping* or *shielding* (i.e., RTA) to incorporate safety in the training process. Reward shaping factors safety into the reward function to reduce risk. However, it is difficult to properly weight safety rewards, and reward shaping can sometimes have unintended impacts on performance. In addition, reward shaping does not guarantee safety during operations. By contrast, RTA can be used to filter the actions of the RL algorithms to ensure safety. A variety of RTA approaches have been explored in the literature, including humanlike intervention [S8], Lyapunov-based approaches [S9], [S10], barrier functions [S11], and formal verification of safety constraints [S12], [S13]. Determining the appropriate way to include RTA in the training process is still an area of active research.

REFERENCES

[S3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[S4] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016, doi: 10.1038/nature16961.

[S5] D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017, doi: 10.1038/nature24270.

[S6] O. Vinyals et al., "Grandmaster level in Starcraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019, doi: 10.1038/s41586-019-1724-z.

[S7] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: Lessons we have learned," *Int. J. Robot. Res.*, vol. 40, nos. 4–5, pp. 681–831, Apr. 2021, doi: 10.1177/0278364920987859.

[S8] W. Saunders, G. Sastry, A. Stuhlmüller, and O. Evans, "Trial without error: Towards safe reinforcement learning via human intervention," in *Proc. 17th Int. Conf. Auton. Agents MultiAgent Syst.*, Jul. 2018, pp. 2067–2069.

[S9] T. J. Perkins and A. G. Barto, "Lyapunov design for safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 3, pp. 803–832, Mar. 2003.

[S10] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based approach to safe reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2018, pp. 8092–8101.

[S11] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proc. AAAI Conf. Artif. Intell.*, Jan. 2019, vol. 33, no. 1, pp. 3387–3395, doi: 10.1609/aaai.v33i01.33013387.

[S12] A. Murugesan, M. Moghadamfalahi, and A. Chattopadhyay, "Formal methods assisted training of safe reinforcement learning agents," in *Proc. NASA Formal Methods Symp.*, J. Badger and K. Rozier, Eds. Cham, Switzerland: Springer Nature Switzerland AG, 2019, pp. 333–340.

[S13] N. Fulton and A. Platzer, "Safe reinforcement learning via formal methods: Toward safe control through proof and learning," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 6485–6492, doi: 10.1609/aaai.v32i1.12107.

The Case for Plan B

A fundamental concept underlying runtime assurance is that of always maintaining a safe "plan B" during mission execution, with the purpose of executing "plan B" as soon as the nominal activity appears to be dangerous. Considered from that perspective, maintaining a "plan B" makes sense, and hundreds of examples can be found in everyday life. For example, financial advisers always recommend that their clients keep a few weeks' worth of salary in a savings account so as to be ready to face unexpected financial expenses, such as the medical costs following an accident, the replacement of a commuting vehicle following a collision, a few weeks at a hotel if the home is damaged by a hurricane, and so on. Defensive driving is another example, whereby a driver is trained to always be aware of his or her surroundings and be ready to take appropriate action in case of unexpected events. Currently, U.S. regulations specify three levels of "circuit breakers" that are used to halt trading in stock exchanges when the Standard and Poor's 500 index drops below certain critical levels. One of the oldest examples of an engineered "plan B" can be found in Greek mythology, in the story of Ariadne's thread [S14], [S15], [S16]. In the story, Ariadne, daughter of Minos, the king of Crete, conceived of using a thread as a "plan B" mechanism that would allow her lover to exit Daedalus' Labyrinth after killing the Minotaur. Similar themes appear in folktales, such as "Le Petit Poucet" ("Hop-o'-My-Thumb") [S17], [S18] and "Hansel and Gretel."

One may also observe this paradigm in nature. It is now known that acacia trees, whose leaves are one common form of food for giraffes and other herbivorous fauna in Africa, are capable of exercising a kind of "plan B," making these leaves lethal by raising the tannin-C in their leaves to deadly levels in a matter of minutes if they are overgrazed [S19]. Likewise, rodents very often create burrows with emergency exits, or "bolt holes," to escape predators, flooding, and other events that may block their main entrances [S20]. The sympathetic nervous system in mammals acts as a physiological "plan B" by preparing the body for a "fight or flight" response. Upon the detection of a threat, glucose is transmitted from storage sites, blood is shifted to the organs that are most essential for physical exertion, the heart rate is increased, adrenaline is released, and cognition is sharpened [S21].

REFERENCES

[S14] S. Cristello, "Ariadne's Thread: The Needle, The Haystack, The Thread//Arts Club of Chicago," 2018. Accessed: Dec. 7, 2020. [Online]. Available: https://www.artsclubchicago.org/exhibition/the-needle-the-haystack-the-thread/

[S15] N. M. Jensen. "Philosophy in labyrinths." Lavigne. Accessed: Dec. 7, 2020. [Online]. Available: http://www.lavigne.dk/labyrinth/e1a_phil.htm

[S16] A. Stieger. "Myth and creativity: Ariadne's thread and a path through the Labyrinth." The Creativity Post. Accessed: Dec. 7, 2020. [Online]. Available: https://www.creativitypost.com/article/myth_and_creativity_ariadnes_thread_and_a_path_through_the_labyrinth

[S17] "Contes de Perrault/Le Petit Poucet." Wikisource. Accessed: Nov. 9, 2020.[Online].Available:https://fr.wikisource.org/wiki/Contes_de_Perrault_(%C3%A9d._1902)/Le_Petit_Poucet

[S18] I. Opie Peter, *The Classic Fairy Tales*. London, U.K.: Oxford Univ. Press, 1974, p. 21.

[S19] J. Flowers and P. Shalgar, "The incredible killer tree that broadcasts an SOS to its neighbours," *Blasting News*, 2015. Accessed: Dec. 7, 2020. [Online]. Available: https://uk.blastingnews.com/world/2015/10/the-incredible-killer-tree-that-broadcasts-an-sos-to-its-neighbours-00610971.html

[S20] R. G. Pisano and T. I. Storer, "Burrows and feeding of the Norway rat," *J. Mammalogy*, vol. 29, no. 4, pp. 374–383, Dec. 1948, doi: 10.2307/1375126.

[S21] R. M. Sapolsky, "Stress in the wild," *Scientific Amer.*, vol. 262, no. 1, pp. 116–123, Jan. 1990, doi: 10.1038/scientificamerican0190-116.

Safety, Reliability, and Security

The primary function of a runtime assurance system is to ensure safety (and, in some cases, security), not reliability. Safety and reliability are two different concepts. Reliability is freedom from errors and failures, while safety is freedom from harm. The *reliability* of a system is its ability to perform consistently over time. For example, physical components may fatigue and degrade over time, and reliability focuses on how long performance is consistent and failure is unlikely [S22], [40]. The reliability of software, which does not degrade over time (unless it changes over time, as in the case of intelligent control that continues to learn online), is based on its freedom from design errors. In a related sense, *security* is freedom from harm from others. The difference in safety and security is in intent, severity, and the likelihood of harm.

REFERENCE

[S22] K. Wiegers and J. Beatty, *Software Requirements*. London, U.K.: Pearson Education, 2013.

the vicinity of humans; 2) the increasing complexity of the primary control system in these systems, making traditional verification techniques prohibitively difficult; 3) the growing desire to quickly update complex system software, without compromising safety and repeating a costly verification process; 4) the ability to perform fast computation online, e.g., integrating trajectories and reachable sets online, which has only recently become a tractable task for nontrivial systems; and 5) the emergence of control barrier function (CBF) and active set invariance filtering (ASIF) methods that give smooth and *minimally invasive* modifications to the desired actions. Additionally, RTA seems to provide an attractive solution to bounding the behavior of machine learning and artificial intelligence systems. While some notable efforts have been made in the direction of generating and verifying provably safe deep neural network controllers [30], [31], [32], [33], the complexity and dynamic nature of these systems often precludes the ability to generate rigorous safety guarantees on the primary controller. The RTA paradigm enables an alternate path without a need to compromise

on safety. Reinforcement learning (RL) applications on safety-critical systems often fall into the domain of safe RL, described as a process of policy learning that maximizes the expected return while ensuring system performance and/or respecting safety constraints [34]. The safe RL community generally takes one of two approaches: modifying the optimality criterion to reduce risk and using an RTA mechanism during the training process, often referred to as a shield [35].

This article provides a tutorial on developing RTA systems. First, a description of the basic architecture, modeling framework, and fundamental definitions is introduced. Second, categories and properties of RTA systems are identified, and systems engineering and human–machine interaction considerations beyond the system dynamics are discussed. Third, a particular category of RTA systems, called the Simplex architecture, is described. Fourth, a second category of RTA systems, called active set invariance filters, is described. Fifth, implicit and explicit variations of both the Simplex architecture and active set invariance filter RTA approaches are described for the canonical double-integrator system. Sixth, approaches for RTA system performance under uncertainty are described. Seventh, verification approaches for RTA algorithms and architectures are presented. Additional topics, applications, definitions, examples, and supplemental information are found in sidebars throughout.

RTA FOR SAFETY-CRITICAL CYBERPHYSICAL SYSTEMS

Cyberphysical systems feature computing devices that interact with the physical world via actuators and sensors [36], and such systems are *safety critical* when failure would result in loss of life, damage to property, and other unacceptable damages, such as environmental harm [37]. An RTA architecture acts as an online verification and enforcement tool for cyberphysical systems that guarantees that certain system-level safety requirements are met at runtime. In this section, RTA problem formulation, appropriate RTA employment in the control loop, and other important design considerations are discussed.

The RTA Architecture

RTA presents an approach to control design that allows a designer to sidestep the common tradeoff between performance and safety. The central idea behind RTA is to decouple the task of enforcing safety constraints from all other objectives of the controller. This is achieved by splitting the control system into two components: a performance-driven *primary* controller and safety-driven *RTA mechanism*. The RTA mechanism preempts the desired inputs from the primary controller when necessary for ensuring the safety of the system and otherwise lets the input pass through unaltered. This approach is associated with the feedback control architecture shown in Figure 1(b). The output of the performance-driven controller is referred to as the *desired* input and assigned the variable u_{des}, and the output of the RTA mechanism is referred to as the *actual* input (or assured input) and assigned the variable u_{act}. At its core, an RTA mechanism is a mapping from a state $x \in \mathbb{R}^n$ and desired

To Use Runtime Assurance or Not to Use Runtime Assurance? That Is the Question

Runtime assurance (RTA) is not a panacea for controller safety. Employing RTA incurs a cost–benefit tradeoff. Implementing RTA can increase system cost, complexity, the amount of testing and evidence required for certification, the number of hardware and software components that can fail, and the attack surface when cybersecurity is a concern. Designers must ask themselves critical questions before deciding to use an RTA system. First, *Could a simple controller meet design requirements*? If a simple control design is sufficient, traditional simulation and testing can provide adequate confidence, and the primary controller can be exhaustively verified, an RTA system should not be used. Second, *Is the system safety or mission critical*? If the system is not safety or mission critical, an RTA could be an unnecessary design complication.

If the system under consideration is safety or mission critical and the control functions can be achieved only by a complex or neural network control scheme, and certification of that high performance controller is not possible via traditional simulation, testing, and exhaustive analysis, an RTA system may be appropriate.

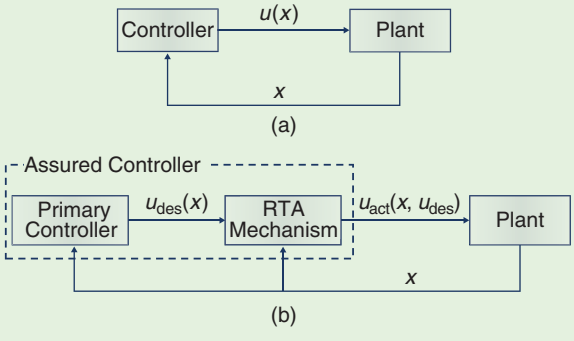

FIGURE 1 To illustrate where RTA sits within the control system architecture, (a) depicts a prototypical feedback control system architecture, and (b) shows the insertion of an RTA mechanism. The primary controller (e.g., a human operator, an advanced control approach, or an autonomous control approach) is performance driven and outputs a desired input u_{des}. The RTA mechanism maps the state $x \in \mathbb{R}^n$ and a desired input $u_{\text{des}} \in \mathbb{R}^m$ to an actual input $u_{\text{act}} \in \mathbb{R}^m$, with the objectives of 1) enforcing state constraints and 2) outputting u_{act} as close as possible to u_{des}. RTA can be thought of as a safety filter on the performance controller output.

input $u_{\text{des}} \in \mathbb{R}^m$ to an actual input $u_{\text{act}} \in \mathbb{R}^m$ and has the objectives of 1) enforcing the safety of the system as defined by state constraints and 2) having u_{act} as close as possible to u_{des} so long as it does not conflict with the first objective. A common choice for measuring how *close* the two signals are is the norm of the difference between the two variables; however, this measurement may also take other forms, such as the integral of the deviation. An RTA mechanism is said to be *assured* when the output u_{act} is always verifiably safe, as defined in later sections. Moreover, an assured RTA mechanism guarantees safety for the entire system and for all time, and this is true regardless of the chosen primary controller. Although not considered an RTA architecture under the definitions in this article, a related concept for an alternative safety architecture is described in "Reference and Command Governors."

Modeling Cyberphysical Systems

Mathematical models that provide an abstraction of real-world system behavior may be defined at varying levels of complexity based on the contrasting needs of accurately predicting system behavior and conducting mathematical analysis. This article focuses mainly on cyberphysical systems modeled as *dynamical systems*; hereafter, the variable $x \in X \subseteq \mathbb{R}^n$ denotes the state vector of the system model, and the variable $u \in U \subset \mathbb{R}^m$ denotes a bounded control input, where X is the set of all possible system states and the set U is the *admissible* set of controls, determined a priori by the actuation constraints of the real-world system. *Continuous-time* system models appear as systems of ordinary differential equations, as in

$$\dot{x} = F_{\text{c}}(x, u) \tag{1}$$

and *discrete-time* system models appear as state update maps, as in

$$x^+ = F_{\text{d}}(x, u). \tag{2}$$

While the computational elements responsible for the control of a cyberphysical system live in a discrete world, the physical laws of motion tend to take the form of ordinary differential equations. In many cases, systems of the form (2) are obtained from discretizing equations of the form (1) over the controller update period.

A natural generalization is to consider *hybrid* systems that combine continuous-time and discrete-time elements [38], [39]. In fact, as seen in the following, some RTA mechanisms are naturally interpreted as inducing a hybrid system in a closed loop, even when the open-loop system is of the form (1). For the purposes of this article, attention is restricted to systems with dynamics governed by (1) or (2) and will not formally characterize hybrid systems, but we note instances where RTA tools have been specifically extended to such systems in the literature.

Throughout this article, it is discussed how one may verify and assure system models of the form (1) or (2) against safety constraints. However, (1) and (2) are only models, and real-world systems generally deviate from their models. When this deviation is significant, safety guarantees derived from the models are invalid. One method for addressing this limitation is to instead consider a *nondeterministic model* that explicitly considers uncertainties that account for the deviation between the real-world system and *deterministic models* presented in (1) and (2). This article first focuses on the deterministic system models (1) and (2) and discusses extensions to nondeterministic models in the "Assurance in the Presence of Uncertainty" section.

Reference and Command Governors

While this article focuses entirely on runtime assurance designs that monitor the primary controller and intervene before the control signal reaches the plant, it is worth noting that other approaches, such as *reference and command governors*, can be used for safety assurance. Instead of modifying the control signal of the primary controller before it reaches the plant, reference and command governors wrap around the closed-loop system and change the reference signal [S23]. The reference governor acts as a prefilter on the desired reference command $r(t)$ and state measurement or estimate $x(t)$ under noise $w(t)$, resulting in a modified reference command $v(t)$. A command, or reference, governor architecture is depicted in Figure S1.

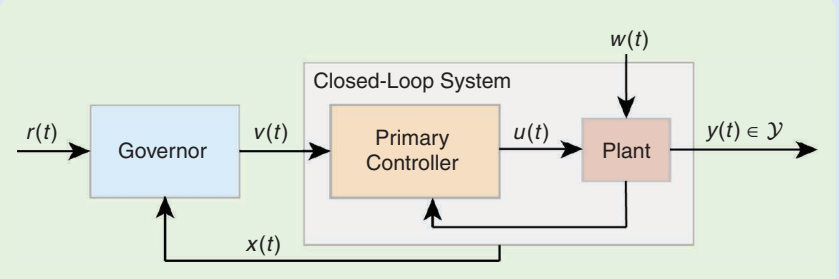

FIGURE S1 A reference governor that modifies the input to the primary controller rather than the primary controller output to assure safety.

REFERENCE

[S23] E. Garone, S. Di Cairano, and I. Kolmanovsky, "Reference and command governors for systems with constraints: A survey on theory and applications," *Automatica*, vol. 75, pp. 306–328, Jan. 2017, doi: 10.1016/j.automatica.2016.08.013.

Admissible Control of Cyberphysical Systems

The main focus of this article is on devising *safe* controllers for cyberphysical systems modeled as in (1) and (2). Safety control schemes must conform to the boundaries of admissible control. That is, a control law can be realized on a physical system only if its outputs are bounded to a set of signals that respect the actuation constraints of that system. A feedback *control law* u denotes a mapping to $\mathbb{R}^m$ from either the state domain or an augmentation of the state domain. For instance, in the context of RTA, many primary control laws will take the form $u : X \to \mathbb{R}^m$, while RTA control laws will generally take the form $u : X \times \mathbb{R}^m \to \mathbb{R}^m$, and either can be extended to include explicit dependence on time. Control laws are said to be *admissible* when their range is the admissible input set U; for example, $u : X \to U$ and $u : X \times \mathbb{R}^m \to U$.

Any control law u can be made admissible via composition with a saturation (also called clamping) function; a *saturation function* $\sigma : \mathbb{R}^m \to U$ is such that $\sigma(u)$ is approximately equal to u when u is in the interior of U and $\sigma(u)$ is approximately the projection of u onto the boundary of U when u is outside of U. Here, there are two main approaches: 1) *hard clamping*, whereby the saturation function $\sigma(u) = u$ for all $u \in U$ and $\sigma(u)$ is a projection onto the boundary ∂U, and 2) *smooth clamping*, whereby σ takes the form of a differentiable function such that $\sigma(u) \approx u$ for $u \in U$. For example, for the case of scalar input and $U = [-1, 1] \subset \mathbb{R}$, $\sigma_1(u) = \max(-1, \min(1, u))$ is a hard clamping function, while $\sigma_2(u) = \tanh(\mathrm{u})$, $\sigma_3(u) = (2/\pi)\tan((\pi/2)u)$, $\sigma_4(u) = u(1 + u^2)^{-1/2}$, and $\sigma_5(u) = \tanh(u + 0.5u^3)$ are examples of smooth clamping functions.

Safety and Invariance of Cyberphysical Systems

In a colloquial context, safety means freedom from harm and danger. For safety-critical systems, safety is freedom from conditions that would, in a worst-case environment, lead to an unacceptable loss, such as loss of control, physical damage to the system under control, loss of human life, human injury, property damage, environmental pollution, and failure of the mission [40], [41]. In the context of a dynamical system, safety is a characterization of the state of the system and its evolution. While alternate approaches to forming safety specifications exist (for example, temporal logic), this research focuses specifically on set invariance requirements derived from static properties on the state. Specifically, safety properties are specified with state constraints, and safety relates to whether a particular initial condition will lead to the safety constraints being satisfied for all time.

In this article, *safety constraints* are defined with inequality constraints on the state. In particular, we consider $\varphi_i : X \to \mathbb{R}$ for $i \in \{1, \ldots, M\}$, where M is the number of safety constraints, and it is always required that $\varphi_i(x) \geq 0$ for each i. The set of states that satisfies all the safety constraints is referred to as the *constraint set* (sometimes called the *allowable set* or constraint space) and is given by

$$C_A := \{x \in X \mid \varphi_i(x) \geq 0, i \in \{1, \ldots, M\}\}. \tag{3}$$

Example

Consider a wingman aircraft with position (x_W, y_W) flying coaltitude in formation with a lead aircraft with position (x_L, y_L), and the requirement is that the two aircraft never go within 30 m of each other. A constraint set over the states $x = [x_L, y_L, \dot{x}_L, \dot{y}_L, x_W, y_W, \dot{x}_W, \dot{y}_W]^T$ is given by

$$C_A = \left\{x \in \mathbb{R}^8 \mid \sqrt{(x_L - x_W)^2 + (y_L - y_W)^2} - 30 \geq 0\right\}. \tag{4}$$

◊

Importantly, there may exist states in C_A that obey the safety constraints at a given time but inevitably lead to violations in the future. For example, in the aircraft case, it is possible for the wingman aircraft to have just more than 30 m of separation from the lead aircraft initially but be traveling too fast to be able turn, climb, and descend to avoid a collision. Although the state falls in the constraint set C_A at a given time, it is not "safe," as it will inevitably lead to a departure from the set. A meaningful definition of safety must encode additional information relating to whether the safety constraints will continue to be satisfied for all time with a particular control law and subject to a particular set of dynamics and actuation constraints.

Informally, a system is safe if the state belongs to C_A for all time. This concept is formalized with the notion of *set invariance*: a state $x(t_0)$ is safe with respect to a closed-loop dynamical system if that state lies in a forward invariant subset of the constraint set, that is, if

$$x(t_0) \in C_S \implies x(t) \in C_S \quad \forall t \geq t_0 \tag{5}$$

where $C_S \subseteq C_A$. In this case, C_S is said to be a *safe set*. Furthermore, control laws are said to be safe when they render some nonempty subset of C_A forward invariant. It is generally desirable that a control law be designed such that the safe set is as large as possible. However, it is generally not possible to find an admissible control law that will render C_A itself forward invariant. The distinction between C_S and C_A is necessary, as the system is subject to actuation and dynamics constraints, meaning that the control signal has only a limited amount of influence over the evolution of the state.

It is important to note that forward invariance, and by extension, safety, are properties of the closed-loop system and not defined in the absence of a controller. The existence, size, and shape of C_S are all dependent on the controller. The question of whether it is possible to find a control law that will render a particular set invariant is addressed with the concept of control invariance. A set $\mathcal{S} \subseteq X$ is said to be *control invariant* (also called viable) if

there exists an admissible control law that renders $\mathcal{S}$ forward invariant. The largest control invariant subset of C_A is known as the *viability kernel* [42]. Intuitively, the viability kernel is the largest achievable safe set, and it forms the boundary between the states for which it is and is not possible to find a control input that will keep the system safe for all time. A fact that will become important in later sections is that control invariant sets are forward invariant under controllers that take the form of certain optimization-based procedures.

Identifying Safe Sets

Computing invariant sets has been the subject of a wide variety of research. A common approach to identifying an invariant region is to find a Lyapunov function. Since stability is a stronger condition than invariance, any stable region under a particular controller must also be invariant under that controller. Moreover, the invariance of a set under an admissible control law $u: X \to U$ also implies control invariance under U. Invariant regions may also be computed by identifying barrier certificates [43]. While identifying Lyapunov functions and barrier certificates is typically a difficult task, they may be computed automatically for polynomial dynamics using sum-of-squares optimization [15], [44], [45] at moderate to high computational costs. Other approaches include using discretized solutions the Hamilton–Jacobi equations [46] and sampling [47]. Safe sets may also be computed for discrete-time systems using approaches in computational geometry [48]. These approaches typically rely on convexity of the set C_A and linearity of the dynamics for the purposes of implementing set operations. See [49], [50], and [51] for a more extensive overview of techniques for invariant set computation.

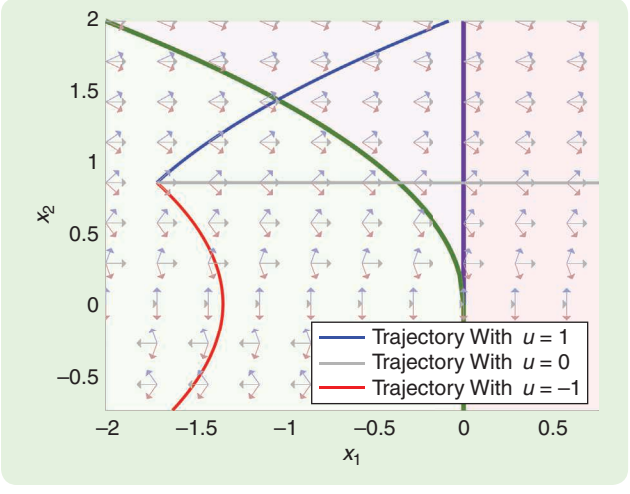

FIGURE 2 A car moving straight toward a wall can be modeled as a double-integrator system. This figure shows a phase plot for a double-integrator system with $u = -1, 0, 1$, where x_1 is the position, x_2 is the velocity, $u = 1$ represents full forward acceleration, $u = 0$ represents continuing at a constant velocity, and $u = -1$ represents full acceleration away from the wall. The viability kernel C_S is shaded green and represents the safe states where it is possible to apply full negative acceleration to avoid hitting the wall. Collision states are shaded red and represent states in which the car has hit or driven past the wall $X \backslash C_A$. The inevitable collision states $C_A \backslash C_S$ are shaded purple and represent cases where the car has not yet hit the wall but is traveling too fast toward the wall to be able to stop in time.

Example

Consider a double-integrator system—for example, a spacecraft in linear motion—described by

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u \end{aligned} \tag{6}$$

where the state vector $x = [x_1, x_2]^T \in \mathbb{R}^2$ is composed of a position state x_1 and velocity state x_2 and $u \in [-1, 1]$ is the acceleration due to thrust. The safety constraint $\varphi(x) = -x_1 \geq 0$ is imposed on the system, reflecting the requirement that the system avoid collision with an object located at $x_1 = 0$ and extending to $x_1 \geq 0$. The constraint set is

$$C_A = \{x \in \mathbb{R}^2 \,|\, -x_1 \geq 0\}. \tag{7}$$

The largest control invariant subset (that is, the viability kernel) of C_A is $C_S = \{x \in \mathbb{R}^2 \,|\, h(x) \geq 0\}$, where

$$h(x) = \begin{cases} -2x_1 - x_2^2 & \text{if} \quad x_2 > 0 \\ -x_1 & \text{if} \quad x_2 \leq 0 \end{cases}. \tag{8}$$

The unsafe states in the constraint set $C_A \backslash C_S$ represent states for which a future collision is inevitable. Intuitively, $x_2 = \sqrt{-2x_1}$ represents the maximum safe velocity at x_1. If this approach speed is exceeded, there will not be a sufficient distance to stop before a collision occurs. This result is made apparent by considering the flow of the system under a recovery maneuver $u = -1$, which applies maximum thrust force away from the obstacle. It can be seen that C_S is a forward invariant set under this control law. Figure 2 provides a depiction of the described sets and the flow under various inputs. ◊

Implicit and Explicit Definitions of the Safe Set

Given a constraint set C_A and an admissible control law $u: X \to U$, the question arises of how one can determine whether a given state $x \in X$ is safe. In many cases, the set of safe states C_S can be identified *explicitly* with a functional representation; e.g.,

$$C_S = \{x \in X \,|\, h(x) \geq 0\} \tag{9}$$

with $h: X \to \mathbb{R}$. Checking whether $x \in C_S$ is, then, equivalent to checking whether $h(x) \geq 0$. The safe set boundary may be determined in a number of ways, such as through Lyapunov arguments and reachability analysis. Additionally, for continuous-time systems, as in (1), Nagumo's

theorem states that under appropriate regularity assumptions on f, u, and h, a necessary and sufficient condition for the invariance of a candidate set C_S, as in (9), is $(\partial h/\partial x)(x)^T F_c(x, u(x)) \geq 0$ for all x such that $h(x) = 0$ [52], and thus, Nagumo's theorem can be used to verify that a candidate safe set is invariant. Moreover, as discussed in the "Assurance in the Presence of Uncertainty" section, it is possible to consider nondeterministic effects through the identification of sets that are *robustly* forward invariant, that is, that are forward invariant under any realization of the nondeterminism.

Example

Consider the double-integrator system (6), now paired with the constraint set $C_A = \{x \in \mathbb{R}^2 \,|\, 1 - \|x\|_\infty \geq 0\}$. Suppose that a control law is constructed such that for all $x \in C_A$, $u = -x_1 - x_2$. Note that actuation constraints can be ignored in C_A if a hard clamping function is used and that for all $x \in C_A$, $u(x) \in U$. In this case, the closed-loop dynamics are

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} x \tag{10}$$

for all states in C_A. A valid safe set can be constructed by identifying a region of attraction contained in C_A. For instance, note that

$$V(x) = 1.2x_1^2 + 0.2x_1x_2 + 1.1x_2^2 \tag{11}$$

is a Lyapunov function for the closed-loop system, and the level curve $V(x) = 1$ is contained in C_A. Thus, $C_S = \{x \in \mathbb{R}^2 \,|\, h(x) \geq 0\}$, where $h(x) = 1 - V(x)$ defines an invariant set contained in C_A. The sets C_S and C_A are depicted along with the system flow in Figure 3(a). Note that while C_S is invariant, it is not the largest forward invariant set contained in C_A. It is said to be conservative. ◊

The explicit identification of forward invariant subsets is typically obtained only at the expense of conservatism. In particular, the methods used to identify forward invariant sets are not generally scalable to complex and high-dimensional systems, and the safe sets obtained with these methods may greatly underapproximate the largest safe set obtainable. However, an explicit (functional) representation of the safe set boundary is not always necessary. One may *implicitly* define C_S in terms of the closed-loop trajectories under the control law. For example, consider a *backup* control law $u_b : X \to U$, and let $\phi^{u_b}(t; x)$ represent the state reached after starting at $x \in X$ and applying u_b for t units of time. Then, the set

$$C_S = \{x \in X \,|\, \forall t \geq 0, \phi^{u_b}(t; x) \in C_A\} \tag{12}$$

is, by definition, an invariant set under u_b, and it is entirely contained in C_A. The utility of this definition arises from the observation that it is possible to check whether *individual states* are safe simply by integrating the dynamics forward from those states. For example, if $C_A = \{x \in X \,|\, \varphi(x) \geq 0\}$, the following are equivalent:

$$1)\ \forall t \geq 0,\ \ \phi^{u_b}(t; x) \in C_A, \qquad 2)\ \inf_{t \in [0,\infty)} \varphi(\phi^{u_b}(t; x)) \geq 0. \tag{13}$$

It is interesting to note that in special cases where the infimum can be solved in closed form, the solution can be used to define an explicit safe set, as shown in (8) and the following example.

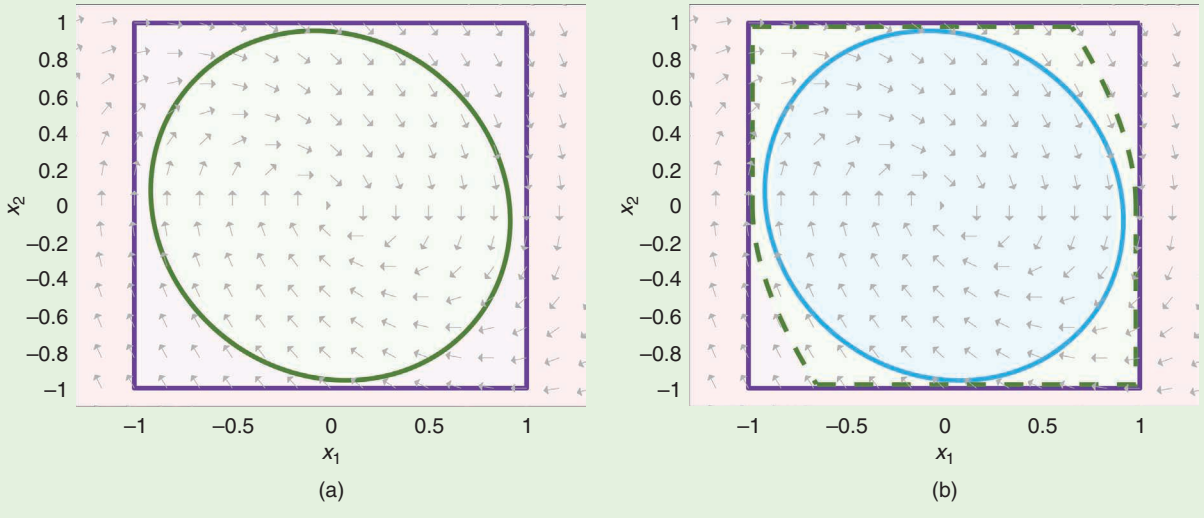

FIGURE 3 Phase plots for the mass-spring-damper system (10), modeled as a double-integrator with the constraint set $C_A = \{x \in \mathbb{R}^2 \,|\, 1 - \|x\|_\infty \geq 0\}$, where the complement of the constraint space C_A is shaded red and $C_A \backslash C_S$ is shaded purple. (a) The explicit safe set C_S (green). (b) The implicit safe set C_S (green) and backup set C_b (blue).

Example

Consider the system

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}\cos(x_2)\\ u\end{bmatrix} \tag{14}$$

with $u\in[-1,1]$, the constraint set $C_A=\{x\in\mathbb{R}^2|x_1\geq 0\}$, and the admissible control law $u_b=1$. The solution for the flow from initial state $x_0=[x_{0,1}, x_{0,2}]^T$ is given by

$$\begin{bmatrix}\phi_1^{u_b}(t;x_0)\\ \phi_2^{u_b}(t;x_0)\end{bmatrix}=\begin{bmatrix}x_{0,1}-\sin(x_{0,2})+\sin(x_{0,2}+t)\\ t+x_{0,2}\end{bmatrix} \tag{15}$$

and x_0 is safe if $\inf_{t\in[0,\infty)}\phi_1^{u_b}(x_0,t)=x_{0,1}-\sin(x_{0,2})-1\geq 0$. Using this information, a safe set under this maneuver can be defined explicitly as

$$C_S=\{x\in\mathbb{R}^2|x_1-\sin(x_2)-1\geq 0\}. \tag{16}$$

The sets C_A and C_S are depicted in Figure 4 along with the flow of the system under $u_b=1$. ◊

Importantly, the system flow $\phi^{u_b}(t;x)$ can readily be evaluated over a *finite* time horizon $\mathcal{T}=[0,T]$ by numerically integrating the dynamics. In this case, the safety of a trajectory can be shown by ensuring that 1) the trajectory lies in C_A over $\mathcal{T}$ and 2) the endpoint of the finite trajectory T lies in a known invariant subset $C_b\subseteq C_A$. That is,

$$C_S=\{x\in X|\forall t\in\mathcal{T},\ \ \phi^{u_b}(t;x)\in C_A\wedge\phi^{u_b}(T;x)\in C_b\}. \tag{17}$$

In this context, a safe terminal set $C_b\subseteq C_A$ is referred to as a *backup set*, and the set (17) is the *safe backward image* (*SBI*) of C_b. Note that numerical integration over a continuous interval $[0,T]$ requires that the trajectory be approximated with a finite number of points sampled along this interval; for example, $\mathcal{T}=\{t_1,\ldots,t_K\}\subset[0,T]$. Testing with the discrete set of points does not strictly prove invariance, as it does not ensure that the trajectory will not leave between the sampled points. While the approximation is sufficient for most practical purposes, this may be addressed with a finer time discretization and by deriving bounds for an inner approximation of the constraint set, as in [53].

While a single case is presented here, various forms of implicit safety may arise under modified assumptions. For instance, [54] considers the case where C_b is not forward invariant and guarantees are in finite time. As discussed in the "Assurance in the Presence of Uncertainty" section, this concept may be extended to nondeterministic systems by approximating the forward reachable sets and invariant tubes around the recovery trajectories. The topology of the constraint and safe sets for both implicit and explicit cases is illustrated in Figure 5.

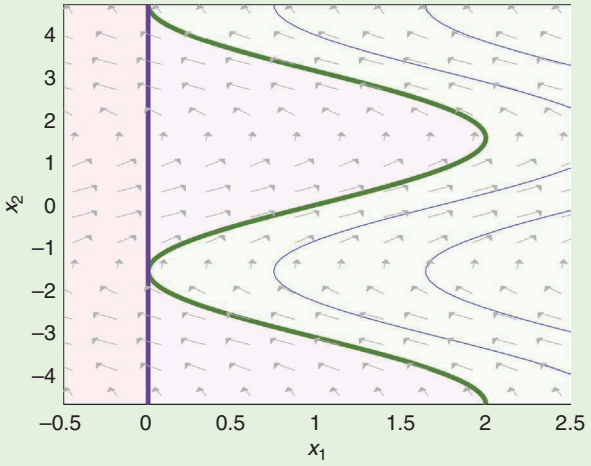

FIGURE 4 The phase plot for system (14) under $u_b=1$. The safe set C_S is shaded green, the complement of C_A is shaded red, and $C_A\backslash C_S$ is shaded purple.

Example

Consider the system (10) along with the associated constraint set $C_A=\{x\in\mathbb{R}^2|\varphi(x)\geq 0\}$, with $\varphi(x)=1-\|x\|_\infty$. We now consider the safe set from the previous example as the backup set. That is, consider the backup set $C_b=\{x\in\mathbb{R}^2|h_b(x)\geq 0\}$, with $h_b(x)=1-V(x)$ and $V(x)$ described by (11). The flow for this system can be evaluated exactly as

$$\phi^{u_b}(t;x)=e^{At}x \tag{18}$$

where A is the system matrix from (10). Consider the backup time horizon $\mathcal{T}=[0,T]$. A state $x\in\mathbb{R}^2$ is in the SBI if the following conditions are met:

$$1)\ \forall t\in\mathcal{T},\ \ \varphi(e^{At}x)\geq 0;\qquad 2)\ h_b(e^{AT}x)\geq 0. \tag{19}$$

The sets for this case are depicted in Figure 3(b). ◊

Identifying Safe Backup Sets

Backup sets C_b are safe terminal sets that act as *seeds* of safety. As seen in Figure 3, the backup set may be used to implicitly identify a larger safe region via trajectories under a backup controller. This is useful because small invariant sets are generally much easier to find than large ones. Furthermore, a finite time trajectory can be used to show safety over an infinite horizon when it is shown that it the trajectory safely reaches such a set. This concept has been frequently explored in the context of trajectory optimization and *model predictive control* (*MPC*) [23], [55]. In the literature related to MPC, the backup set is often referred to as a *terminal feasible invariant set*. Backup sets *are* safe sets and may be computed as any safe subset of C_A, using the methods described previously. However, it is common to construct these from domain knowledge of safe configurations of the particular system.

For an example of a safe backup set, consider that ground vehicles and VTOL air vehicles may bring themselves to

rest in a safe location. In this case, the role of u_b is to generate a safe stopping maneuver, for example, [56]. Individual rest states are invariant points in the state space, and sets of adjacent rest states form invariant surfaces and volumes. In many cases, it is not possible and practical to bring a vehicle to rest, and a backup set may instead be identified from safe periodic trajectories. For example, fixed-wing aircraft may compute safe loiter circles [18], [22], [23], [57] and spacecraft may compute a set of safe orbits [58]. In many cases, the invariant set is a region without volume, and only an infinitesimal perturbation is required to cause a departure from the set. In practice, it is desirable to show that such surfaces are locally attractive under some backup control law and explicitly identify a larger invariant region around the initial set. For example, [58] develops a control law that stabilizes a spacecraft to a set of safe orbits.

Example

Consider the damped linear system

$$\ddot{x} = -\dot{x} + u \tag{20}$$

with the constraint set given by $\mathcal{C}_A = \{x \in \mathbb{R} \,|\, x \geq 0\}$. A safe backup set is given by the subset of $\mathcal{C}_A$ with zero velocity: $\mathcal{C}_b = \{x \in \mathbb{R} \,|\, x \geq 0, \dot{x} = 0\}$. Note that $\mathcal{C}_b \subset \mathcal{C}_A$ and that $\mathcal{C}_b$ is invariant when $u = 0$. ◊

Example

Consider two spacecraft in planar orbit around a central body: 1) a *target* spacecraft, which is in a fixed circular orbit with period τ, and 2) a *chaser* spacecraft of mass m. In this setting, the dynamics of the chaser spacecraft are given by the Clohessy–Wiltshire–Hill equations, as developed in [59]

$$\begin{aligned}
\dot{x}_1 &= x_3 \\
\dot{x}_2 &= x_4 \\
\dot{x}_3 &= 3n^2 x_1 + 2n x_4 + \frac{1}{m} u_1 \\
\dot{x}_4 &= -2n x_3 + \frac{1}{m} u_2 \\
\dot{x}_5 &= -|u_1| - |u_2|
\end{aligned} \tag{21}$$

with state $x \in \mathbb{R}^5$ and control input $u \in [-u_{\max}, u_{\max}]^2$. In this setting, x_1 and x_2 denote the relative distances between the spacecraft, x_3 and x_4 denote the relative velocities, and x_5 denotes a fuel state. Additionally, $n = 2\pi/\tau$ [59]. The constraint set is defined by the constraints that the chaser must not run out of fuel and must stay at a distance greater than $R_{\min}$ from the target spacecraft. The constraint set is $\mathcal{C}_A = \{x \in \mathbb{R}^5 \,|\, x_1^2 + x_2^2 - R_{\min}^2 \geq 0, x_5 \geq 0\}$.

Backup Set From Invariant Points

Invariant points exist as zero-velocity states in the x_2–x_5-plane; hence, $\{x \in \mathbb{R}^5 \,|\, x_1 = x_3 = x_4 = 0\}$ is invariant for $u \equiv 0$. A safety set is obtained from this set as $\mathcal{C}_{b,1} = \{x \in \mathbb{R}^5 \,|\, x_1, x_2, x_3 = 0, x_5 \geq 0, |x_2| \geq R_{\min}\}$.

Backup Set From Periodic Trajectories

It can be shown that the chaser spacecraft is on an elliptical orbit around the target spacecraft whenever $u = 0$ and $x_3 = (n/2)x_2$, $x_4 = -2n x_1$, with each individual orbit

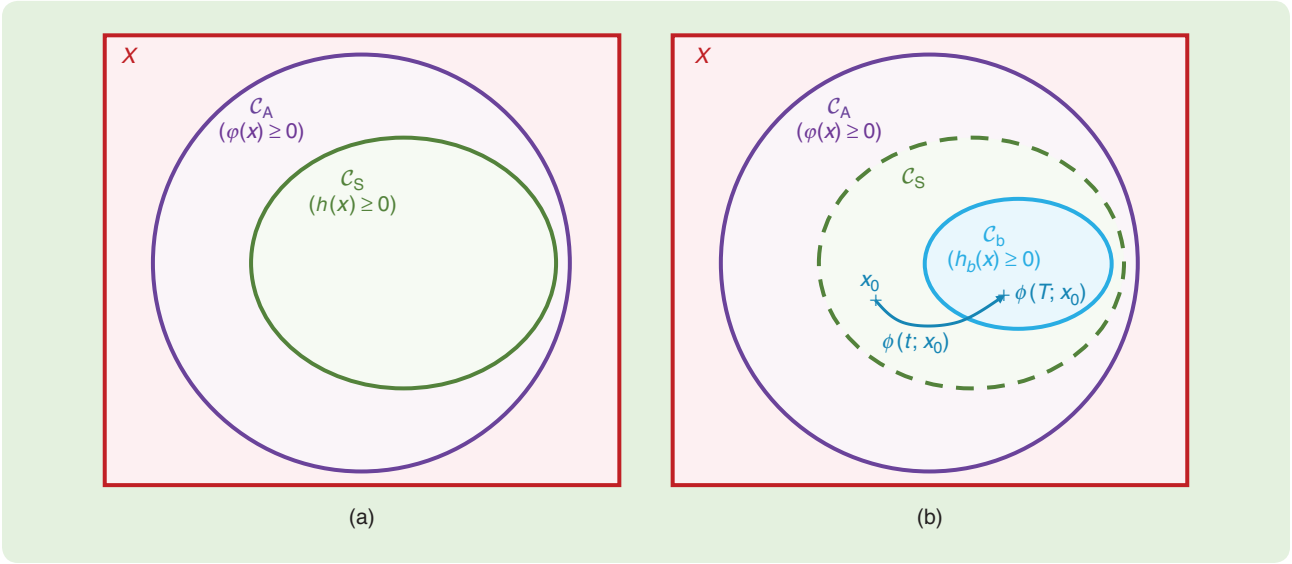

FIGURE 5 The safe set $\mathcal{C}_S$ and constraint set $\mathcal{C}_A$ topology for (a) explicit and (b) implicit cases. Here, X represents the set of all states, the constraint set $\mathcal{C}_A$ is the set of states considered safe when there are no controller limitations, and the safe set $\mathcal{C}_S$ is a subset representing the set of states that are safe under the limits of the control system. For example, a car approaching a wall is in the constraint set $\mathcal{C}_A$ so long as it does not hit the wall. However, if the car is 2 ft from the wall and moving at 100 mi/h, it can be in the constraint set $\mathcal{C}_A$ but not in the safe set $\mathcal{C}_S$. To be in $\mathcal{C}_S$, the car must be able to stop before hitting the wall. In cases where it is hard to define $\mathcal{C}_S$ but a predefined backup control is known (e.g., the car slamming on the brakes), an implicit backup set $\mathcal{C}_b$ may be used instead of $\mathcal{C}_S$.

occupying the position states $x_1^2 + (1/4)x_2^2 = b^2$, where $b \geq 0$ is the semiminor axis of the orbit. Hence, the linear subspace $\{x \in \mathbb{R}^5 | x_3 = (n/2)x_2, x_4 = -2nx_1\}$ is an invariant manifold composed of all such closed orbits. A backup set can be constructed from this subspace by considering a range of orbits that fall into the constraint set $C_{b,2} = \{x \in \mathbb{R}^5 | x_3 = (n/2)x_2, x_4 = -2nx_1, x_1^2 + (1/4)x_2^2 \geq R_{\min}^2\}$. ◊

PROPERTIES OF RTA SYSTEMS

RTA has evolved over the past several decades, stemming from research in control theory and computer science and electrical, mechanical, and aerospace engineering. In this section, each of several basic classifications and properties of RTA mechanisms are defined, concluding with RTA–human interaction considerations. RTA approaches can be classified as explicit or implicit, zero or first order, and latched or unlatched. Explicit approaches precisely define a specific safe set, while implicit RTA approaches define recovery trajectories under a predefined backup control law. Zeroth-order RTA methods do not use gradients of the dynamics and are often associated with RTA systems that use methods described in the "The Simplex Architecture" section, while first-order methods take advantage of gradient computations and are used in methods such as those described in the "Explicit and Implicit ASIF" section. Latched RTA systems switch to a backup controller and remain under its control until a specified release condition is met, while unlatched implementations return to the primary control as soon as the system returns to a safe set. In addition to these classifications, RTA systems feature safety and performance properties described in this section, including innocuity, viability, nuisance freedom, and integrity monitoring. Finally, this section discusses important concepts with respect to RTA–human interaction, including variable risk tolerance, the ability to turn the RTA off, transparency, and missed detection/false alarm considerations.

Implicit and Explicit RTA Approaches

Fundamentally, an RTA mechanism is a control law that renders a subset of the constraint space forward invariant, activates a recovery response near the boundary of that set, and passes through the desired input from the primary controller everywhere else. The set made safe under the RTA is referred to as the *safe operational region* of the RTA system, and it is denoted by C_S. In cases where the recovery response relies on switching to a backup control law u_b, the safe operational region is a subset of the constraint set C_A that is forward invariant under that control law. Approaches of this type are considered in the "The Simplex Architecture" section. Alternatively, in cases where the recovery response is determined as the solution to an optimization program and safety is enforced with the constraints of that program, the safe operational region is a control invariant subset of C_A. This is because the optimization program searches over all feasible control actions, that is, control invariant sets are forward invariant under certain optimization procedures. Approaches of this nature are considered in the "Explicit and Implicit ASIF" section.

As with the safe sets discussed previously, the safe operational region can be identified explicitly and implicitly. *Implicit* (or trajectory-based) approaches compute finite time trajectories under a backup controller $\{\phi^{u_b}(t; x) \in X | t \in \mathcal{T}\}$ *online* (at runtime) and use the information to decide when and how to intervene. *Explicit* (or region-based) approaches may or may not switch to a backup controller but do not rely on simulating trajectories of the backup controller online. Typically, explicit approaches identify trusted regions *offline* via the construction of either a large forward invariant or control invariant set. That is, implicit RTA approaches are those that utilize an online *look ahead* and will bound the system to an implicitly defined safe set, such as (17), while explicit RTA approaches determine safe states a priori and will bound the system to an explicitly defined safe set, such as (9). This concept is demonstrated in the following example.

Example

Consider the mass-spring-damper system described by (10), with the constraint space $C_A = \{x \in \mathbb{R}^2 | 1 - \|x\|_\infty \geq 0\}$. An RTA mechanism can be constructed by requiring that the backup controller be applied near the boundary of a set C_S, where C_S is safe under the backup controller. Let the RTA mechanism be

$$u(x) = \begin{cases} u_b & \text{if } \vartheta(x) \\ u_{des} & \text{otherwise} \end{cases} \tag{22}$$

where u_{des} is the desired control input and $\vartheta(x)$ represents the activation condition (or switching condition). Considered in the following are implicit and explicit activation conditions.

Explicit Condition

Let $C_S = \{x \in \mathbb{R}^2 | h(x) \geq 0\}$, with $h(x) = 1 - V(x)$ and $V(x)$ being the Lyapunov function (11). The set C_S is said to be trusted, as it is safe under u_b. The activation condition

$$\vartheta(x) : h(x) \geq \varepsilon \tag{23}$$

ensures the safety of C_S with respect to u_{act} by activating u_b near the boundary, where $\varepsilon > 0$ defines the size of the RTA activation region. Since $V(x)$ defines a Lyapunov function for the closed-loop dynamics under u_b, it is known that $h(x)$ is increasing whenever u_b is applied. Furthermore, since u_b is activated before the boundary $h(x) = 0$, (22) enforces that $h(x)$ is always positive, meaning that the system will be forward invariant in $C_S \subset C_A$.

Implicit Condition

Let $C_b = \{x \in \mathbb{R}^2 | h(x) \geq 0\}$, with $h(x) = 1 - V(x)$ and $V(x)$ being the Lyapunov function (11). Then, a condition for invariance in the SBI of C_b is

$$\vartheta(x) : [\forall t \in [0, T], \ \varphi(e^{At}x) \geq \varepsilon_1] \ \wedge \ [h_b(e^{AT}x) \geq \varepsilon_2] \tag{24}$$

where ε_1, $\varepsilon_2 > 0$. The SBI, in this case, defines the safe operational region for the RTA. It is depicted as the green region in Figure 3(b). ◊

There are important tradeoffs associated with implicit and explicit approaches. While identifying invariant sets explicitly offline generally reduces the online computational burden of the algorithm, the task may become intractable for complex and high-dimensional systems. In such cases, implicit trajectory-based approaches become favorable. The key advantage to assessing safety through trajectories online is that, in the simplest case, it requires only a finite-time simulation of the recovery maneuver, which is generally a tractable task. A secondary benefit is that it handles dynamic environments and changes in the model better; that is, rather than needing to recompute an invariant subset of C_A, one needs only to update the simulation parameters. However, an important practical consideration is that trajectory-based approaches rely on the ability to accurately forecast recovery trajectories, and the predicted behavior may be increasingly sensitive to noise (that is, less accurate) as the length of the backup trajectory is increased.

It is interesting to note that since a backup set C_b is an explicitly defined invariant subset of the constraint set C_A, an implicit approach to safety will typically involve a combination of 1) explicitly identifying an invariant set C_b offline and 2) finding a safe trajectory to this set online. In this sense, the explicit approach can be viewed as a special case of the implicit approach, where the backup trajectory consists of only a single point. Furthermore, any explicitly defined safe set can be used as part of an implicit approach. The implicit approach allows for the problem to be solved by taking full advantage of both offline and online resources. Specifically, finding a larger backup set offline reduces the online computational burden by reducing the length of the trajectory required to obtain the same operational region; likewise, integrating backup trajectories over a longer horizon reduces the need to identify large safe sets offline. A depiction of this tradeoff appears in Figure 6. Generally, for simple systems, the problem can be solved offline, and as systems become more complex, one must rely more and more on online simulations.

Zero-Order and First-Order Methods

Filtering approaches in RTA may be classified into zero-order and first-order methods. Zero-order algorithms are derivative-free. They are typically, though not necessarily, associated with the *Simplex* architecture and choosing from a discrete set of possible control signals. For example, the RTA mechanism may choose between applying the input from the primary control law u_{des} and a backup control law u_b at any time. In this case, one may observe the effects of chatter; that is, variations in the output of the RTA mechanism are not smooth with respect to variations in the initial state and desired input. This may have practical implications for real-world systems. For instance, the action may cause excessive wear to the actuators, onboard components may experience interruption to their operations, and passengers may observe the intervention as being more abrupt, resulting in a less pleasant experience (for example, consider an automobile that intervenes only through applying the brakes at maximum capacity). These effects may be handled with various heuristics, for example, hysteresis and interpolating between u_b and u_{des} as the state approaches the boundary of the safe operational region. Alternatively, one may turn to a first-order method.

First-order methods require computing derivatives and are typically associated with methods described in the "ASIF" section. In this case, the gradient information is used in a *barrier condition*, which constrains the set of available

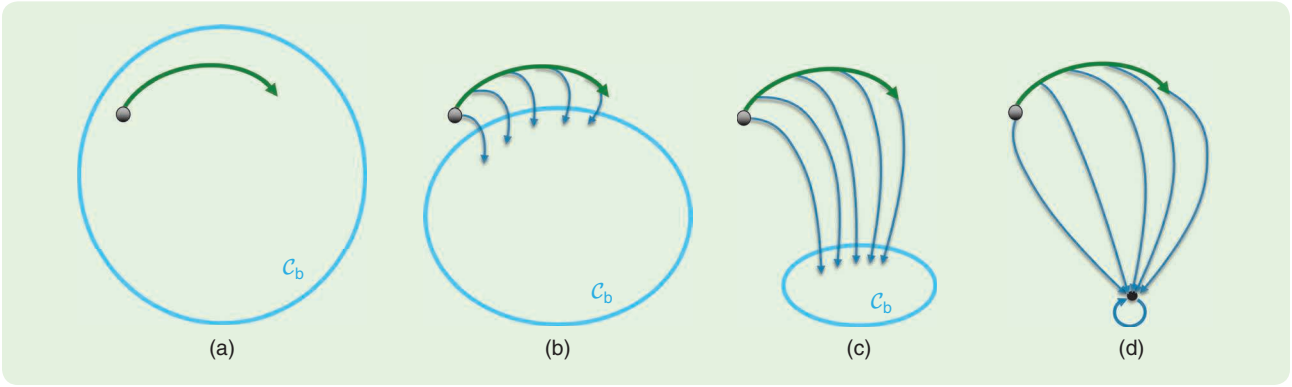

FIGURE 6 A notional depiction of the tradeoff between a longer integration horizon and larger backup set. The thick green arcing arrow represents the desired trajectory of the primary controller, the teal region represents a backup safe set C_b, and the thin teal arcing arrows represent trajectories to the backup set. Large explicit backup sets C_b reduce the need for the online computation of trajectories to the backup set but may be intractable for complex and high-dimensional systems. Computing an online finite-time simulation of a recovery maneuver may be more tractable in some cases but is more vulnerable to noise and model inaccuracies. (a) An explicit approach (no trajectory forecasting). (b) An implicit approach with large backup set and short time horizon on the backup controller. (c) An implicit approach with small backup set and long time horizon on the backup controller. (d) An invariant point or periodic trajectory.

inputs to a subset of admissible and safe inputs. An optimization framework may be used to choose the input within this set according to some cost function. A common cost specification requires that the RTA choose the control that is closest (in a two-norm sense) to the desired input. RTA filters constructed in this way will have smooth variations in the output signal u_{act}. While this eliminates the effect of chatter, it introduces added complexity into the problem and generally requires greater computational resources. The Robotarium is an example of explicit ASIF RTA implementation and discussed further in "The Robotarium: A Runtime Assurance-Enabled Remote Access Swarm Robotics Testbed."

Latched and Unlatched Implementations

RTA systems based on those described in the "The Simplex Architecture" section activate a recovery maneuver when the monitor detects that one of the boundary violation conditions has become active. At this point, the decision logic must specify when to return control to the primary controller. In an *unlatched* implementation, the decision logic releases control of the recovery controller as soon as the boundary violation condition becomes inactive. For example, an unlatched RTA implementation that switches from the primary control law u_{des} to the backup control law u_{b} whenever a condition ϑ is satisfied will take the form of a hybrid system, as in (22). Alternatively, a *latched* implementation will hold its activation state until a separate release condition is met. The release condition may, for example, consist of following the backup trajectory for some fixed amount of time, bringing the system to a particular configuration, and having a human operator return control after assessing the problem. Auto GCAS is an example of a temporary latched system that holds the maneuver until the aircraft is on a collision-free course [61]. Other implementations of latched RTAs may require a system reset before switching back, such as a computer reset supervised by a human operator. Note that this requires that the decision logic remember the current activation state.

Properties and Systems Engineering Considerations for RTA Design

While it is convenient to treat safety as a binary "safe" or "unsafe" property, safety is often a complex weighting of risk among primary and secondary safety constraints. This section provides a few definitions of design considerations for incorporating RTA solutions into the larger control system. In practice, RTA system design ultimately relies on domain-specific knowledge to achieve acceptable performance. However, there are common safety and performance properties of good RTA designs. To keep the system safe, the RTA mechanism must select control inputs that are *innocuous* and *viable*. In addition to safety, the performance of the RTA may be evaluated by its adherence to a *nuisance-free* property.

Innocuity

Ideally, the entire state of the system is contained in one vector x, and the entire constraint set can be explicitly defined in a set C_{A}. However, in practice, this is rarely the case. Instead, RTA decisions are not solely based on system dynamics but must factor in the states of interacting subsystems, human interactions, fault management, and interlock condition management. *Innocuity* is the property that each component of the design has no unacceptable impact on the larger system design [89]. The innocuity of RTA systems implies that the selected control action does not have unintended behavior, such as causing a violation of a secondary safety constraints. An interlock is a set of two mutually exclusive states, such as mechanisms that prevent elevator

The Robotarium: A Runtime Assurance-Enabled Remote Access Swarm Robotics Testbed

The Robotarium [83], [84], [85], [86] is a remotely accessible swarm robotics testbed with the stated mission to *democratize robotics by providing remote access to a state-of-the-art multirobot research facility*. The lab consists of a 12 × 14-ft custom arena and a number of differential drive robots, known as GritsBots. The testbed exhibits features such as Vicon-based real-time motion capture, wireless inductive charging, video capture of experiments, and an above-mounted projector that allows users to project time-varying environmental projections onto the testbed during experiments. Users may submit code to control the robots through MATLAB and Python scripts that are submitted through the Robotarium website.

The open access nature of the lab presents unique challenges in terms of safety. Experiments must remain as faithful as possible to the user-specified behavior while being provably safe so as to prevent damage to the lab equipment. Offline verification of the user code would be a prohibitively difficult and time-consuming task for the operators of the lab and place an unnecessary burden of correctness on its users. Consequently, a runtime assurance solution to safety is utilized, and user-submitted algorithms are treated as black-box functions that can be probed at runtime. Specifically, control barrier function-based quadratic programs [explicit active set invariance filters (ASIFs)] are used to prevent collisions among the various robots involved in an experiment. An explicit ASIF-based approach is attractive in this case, as it is minimally invasive to the desired inputs and provides a smooth overriding behavior. Since the planning is conducted under a single-integrator dynamics model, the explicit identification of viable sets is a fully tractable problem offline.

doors from opening when the elevator is in motion and prevent the elevator from moving when the doors are open. This relationship between RTA-level and system-level constraints may result in a combination of RTA system modes (described as a finite-state machine), if statements, and dynamics considerations [90].

For example, in the development of Auto GCAS, the high-level requirements, in order of precedence, were do no harm, do not interfere, and prevent collisions [61], [91]. The "do no harm" and "do not interfere" requirements in Auto GCAS are examples of innocuity properties. These requirements do not directly define the RTA response but describe special considerations to prevent unintended consequences of the RTA design. This requirement ordering might seem counterintuitive, as one might expect preventing collisions to be the top priority of a collision avoidance system. However, this order of requirements proved critical to acceptance of the system [92]. To give a little more insight, selected generalized conceptual requirements for Auto GCAS are presented in the following and divided in terms of "do no harm," "do not interfere," and "prevent collisions." The "prevent collisions" requirements are typical of what is generally considered an RTA requirement. However, the "do no harm" and "do not interfere" requirements describe how the RTA system should operate in the context of the aircraft control system that interacts with human operators, experiences failures, and has a defined set of interlock conditions:

- *Do no harm*:
  - The automatic recovery will not cause harm to the pilot, aircraft, and components.
  - The automatic recovery maneuver will not place the aircraft in an uncontrollable state.
  - The pilot will be able to interrupt a maneuver.
  - The pilot will be able to manually engage a maneuver.
  - Subsystem monitors will determine whether a failure exists and be provided to Auto GCAS.
  - Auto GCAS will not activate an automatic recovery maneuver if a failure of a subsystem supporting Auto GCAS exists.
  - Auto GCAS will leave the failed mode state if a failure does not exist.
  - The automatic recovery maneuver will not activate during aerial refueling.
  - The automatic recovery maneuver will not activate when the aircraft has an excessively high angle of attack.
  - The automatic recovery maneuver will not activate when the aircraft velocity is too low.
  - Auto GCAS will inform the pilot whether the aircraft speed is too low for the automatic recovery maneuver.
- *Do not interfere*:
  - The automatic recovery maneuver will not activate during landings.
  - The pilot will be able to turn the system off.
  - The pilot will be able to select the protection level.
  - Auto GCAS will notify the pilot when the automatic recovery maneuver initiates and terminates.
  - The system will record data about each automatic recovery maneuver activation.
  - The digital terrain elevation data used by the system will have a resolution and accuracy that supports safe and nuisance-free operation.
- *Prevent collisions*:
  - The system will conduct an automatic recovery maneuver when the projected recovery trajectory intersects the terrain profile contour with buffers.
  - The system will terminate an automatic recovery maneuver as soon as it is determined that the aircraft will clear the terrain threat.
  - The recovery maneuver will consist of a roll to wings level and a pull-up.

Systems engineering, computing hardware availability, and human–machine interaction all require practical considerations that play important roles in incorporating secondary safety constraints in RTA designs. From a computing hardware standpoint, assuming the availability of high-performance computing may not be valid. In some domains, design constraints, such as size, weight, and power, coupled with vibration and radiation tolerance limit the processing and memory of computing hardware onboard the dynamical system. In these cases, simpler RTA solutions may be favored over optimal solutions. An example of this occurs in aviation, where RTA designs are expected to be backward compatible with older aircraft and simple approaches may be to climb, descend, and turn right or left at a specified rate [93]. Additionally, when the automated backup control of the RTA systems impacts human operators, humans trust an easily understood maneuver consistent with human training, expectations, and preferences [94]. For Auto GCAS, the familiarity of the roll to wings level and pull maneuver and its consistency with pilot training and behavior resulted in strong positive perceptions of the system [92].

Viability

Viability ensures that the selected control action does not eliminate the availability of a safe action downstream; i.e., the RTA mechanism will always be able to identify a safe input in the future. This concept is formalized in Figure 2, where the green area under the curve describes the maximum velocity for any distance from the obstacle. One way to ensure viability in an implicit RTA design approach is to define a limited set of predetermined recovery actions. Simple predefined recovery responses can be preverified for use, reduce the online computation burden, and make the RTA system more easily understood and trusted by a human operator [94]. In Auto GCAS, rather than optimizing a trajectory for each specific collision avoidance scenario, a predefined set of verified maneuvers was created.

In F-16 Auto GCAS, a single roll to wings level and 5G pull maneuver is conducted [62]. In the Automatic Air Collision Avoidance System, nine different maneuvers are considered [95]. In patented [96] and experimental Auto GCAS designs for small unmanned aerial vehicles [97], lesser-capability aircraft [98], and cargo-class aircraft [99], three or more possible maneuvers may be considered. In the case of multiple maneuvers, one popular approach that combines viability with nuisance freedom is to engage the last available maneuver right before it is too late.

Nuisance Freedom

In a *nuisance-free* RTA system, the constraint set C_A and corresponding operating envelope C_S of the primary controller are as large as feasible. Other ways to describe this property is that the RTA should be minimally invasive and not interfere with the primary function of control system u_{des} unless absolutely necessary. In Auto GCAS, a collision avoidance maneuver is considered nuisance-free if it occurs after an aware pilot would have maneuvered to avoid a collision. To formally define nuisance-free operations, a *time-available* metric described zero time available as the point where an activation would just barely scrape the ground, and increasingly positive time available would result in maneuvers at greater altitudes above the terrain. To assign a value to an acceptable time-available activation range, a 1995 study measured when pilots felt an aggressive recovery should be activated to avoid a collision [61], [91]. Pilots flew toward the ground at a variety of dive angles, bank angles, air speeds, and load factors and activated a recovery maneuver when their comfort threshold was met. After each run, the pilots rated the timing of the recovery initiation, their anxiety level during the maneuver, and the precision/aggressiveness of the maneuver for each run. Based on this information, a 1.5-s time-available design criterion was identified, and collision avoidance maneuvers that activated with less than 1.5 s of time available were considered nuisance-free [61], [91].

More generally, the number of RTA activations, magnitude of the response, activation timing, and size of the safe set can be used to compare nuisance criteria across different RTAs:

- In many scenarios, the RTA should not come on often because activations interrupt the primary mission. An *RTA activations* metric tracks the number of seconds and discrete time steps that a recovery maneuver is in control. This metric allows comparing different RTA approaches, as it provides a measure of the relative conservatism of various RTA solutions in the presence of the same, possibly faulty, primary controller.
- The RTA should ideally not impart a large change in control, measurable via a *control magnitude* metric.
- An *RTA invasiveness* metric describes whether the RTA activates appropriately when near the safety constraint boundary while refraining from activating when far from the allowable boundary. A variety of application-specific measurements of this may be applied, such as the time-available metric in Auto GCAS.
- An *RTA safe set size* measures the volume of the safe set C_S. The larger the volume, the more space the primary controller has to operate optimally and the less invasive the RTA is. Different RTA approaches may be more or less conservative, which impacts the size of C_S.

RTA Integrity Monitoring

Recognizing and accommodating failures that could impact an RTA system is a critical element of RTA design. The failures could come from inside the RTA system and from externally provided information that the RTA uses to make a response decision. Possible software checks include monitoring for missing "heartbeat" signals from components and "reasonableness checks" that ensure that incoming data are within a reasonable range of values [97]. For example, a failure of the inertial navigation system should not allow Auto GCAS to roll inverted and pull down into the ground instead of pulling up to avoid a collision [61]. Auto GCAS uses system-wide integrity management hardware and software tests to mitigate single-point-of-failure risks [61], [95], [100].

Human Interaction With RTA Systems

When designing RTA systems, it is important to consider the interaction of the system with human operators. While the topic of human–machine interaction can swell quickly, this article focuses on a few key design considerations in RTA.

Variable Risk Tolerance and an "Off Switch"

One of the challenges of developing an RTA system is the need to balance nuisance freedom with the risk tolerance of a particular task and operator. Adding the ability to tune risk corresponding to safety buffers and other factors facilitates the flexibility and acceptability of the RTA in operational use. For example, Auto GCAS features two pilot-selectable modes that enable variable risk tolerance: a "norm" mode with adequate safety buffers for most cases and a "min" mode that minimizes the buffer size for nuisance-free low-level flying, at the tradeoff of reduced protection [61], [91]. In addition, the ability to turn the system off was an important feature that engendered pilot trust [92].

Transparency and Trust

The primary basis of trust in automation comes from dependability and predictability [101]. However additional factors with a measurable impact on human trust include past performance (for example, system pedigree and heritage), simplified and understandable performance, system intent, and reliability [94]. In Auto GCAS, transparency comes through pilot training before operations and through visual indicators on the aircraft's head-up display

[92], [102]. A discussion of the difference among safety, security, and reliability is provided in "Safety, Reliability, and Security." Reliability is important for human trust in RTA systems, as humans initially assume that machines are perfect, and any errors rapidly deteriorate trust [103]. Knowledge of Auto GCAS's 98% reliability increased test pilot trust of the system [92].

Missed Detections and False Alarms

It is important to consider an acceptable rate of missed detections and false alarms in the design of an RTA system. *Missed detections* are a failure of the monitor to detect an imminent safety violation. For example, if Auto GCAS fails to detect an imminent collision with terrain, it could lead to loss of life and the aircraft. *False alarms* occur when the RTA monitor believes the system is entering an unsafe state when, in reality, no safety violation is imminent. False alarms erode confidence in the RTA system. For example, during the psychological study of the development and deployment of Auto GCAS, one pilot indicated that a single false fly up (automatically maneuvering to avoid the ground when there was no imminent collision) would likely cause pilots to turn the system off and lose the protection it provided [61], [91]. Developing pilot trust in Auto GCAS hinged on the system's nuisance avoidance criteria, that is, keeping the number of false alarms very low [102].

In statistical hypothesis testing, missed detections and false alarms are type 1 and type 2 errors. Hypothesis testing is a formal technique to verify that a system meets a specification or requirement [104]. A hypothesis can have two possible outcomes: a null hypothesis H_0, such as a collision is not imminent, and an alternative hypothesis H_1, such as a collision is imminent. As summarized in Table 1, a type 1 error is a false alarm (that is, activate u_b when not near the constraint boundary ∂C_A), while a type 2 error is a missed detection (that is, failing to activate u_b near ∂C_A, resulting in a departure from the safe set). When considering a collision avoidance system, for example, a type 1 error occurs when a collision avoidance system maneuvers to avoid a collision that would not have happened, and a type 2 error occurs when a collision avoidance system does not maneuver to prevent a collision.

In some systems, such as Auto GCAS, the acceptable rates of false alarms α are very low because of the consequence of an unnecessary interruption of the system mission is high. In other systems, such as elevators, the acceptable rate of a missed detection β (not detecting that a human is in the way of the closing doors) would be very low, and false alarms would be more acceptable (there is almost always an option to take the stairs). During the development of RTA systems, studies are required to determine acceptable α and β rates.

THE SIMPLEX ARCHITECTURE

Simplex architecture RTA designs [60], [105], [106], [107] feature a *monitor*, or watchdog, that watches for imminent violations of safety constraints and a backup *safety controller* that provides a guaranteed safe control output to remedy the unsafe situation. A critical feature here is that the backup control is applied via a switch rather than gradually as in ASIF, described in the following. One way to describe Simplex RTAs is as hybrid dynamical systems [38] described by (22). The monitor and safety controller are inserted before the plant, as demonstrated in Figure 7. In this model, the following functional components describe specific functions and interactions that may be implemented on one or more physical components:

- *Plant*: This functional component is the system under control and could be a spacecraft, aircraft, boat, car, or other robotic system.
- *Primary controller*: This functional component is the primary high-performance controller of the system that may not be fully verified with traditional offline verification techniques. Depending on the application, this could be a human pilot, neural network-based control, or complex control software.
- *Backup controller*: This functional component continuously computes a safe control response to the state of the plant. For example, in Auto GCAS, the safety controller is a predefined roll-and-pull maneuver. One school of thought places a heavy verification burden on the safety controller, boundary monitor, and decision logic as a way to certify advanced controllers that cannot be verified to the current certification standards [108]. Another school of thought requires no assumptions on the safety and verification of the backup controller. That is, safety can be guaranteed through the online integration of the backup controller, and the system may revert at any time to the most recently computed safe trajectory. Formal guarantees are obtainable in both cases.
- *Boundary (safety) monitor*: This functional component monitors the state of the plant and nominal controller

TABLE 1 The false alarm and missed detection type 1 and type 2 errors, with collision avoidance as an example.

	H_0	H_1
	Collision	Collision
	Not imminent	Imminent
Do not reject H_0		Type 2 error
	Correct	False negative
System decides		Missed detection
Not imminent		β probability
Reject H_0	Type 1 error	
	False positive	Correct
System decides	False alarm	
Collision imminent	α probability	

for predetermined safety boundary violations. Examples of boundary violations might be a trajectory prediction that intersects the terrain (indicating an imminent ground collision) and a control input that would cause excessive acceleration (risking damage to the structure and components).

» *Decision logic*: The functional component takes inputs from all the other components pictured and determines which control output, complex or safety, to output to the plant.

Variations on Simplex

Variations on Simplex include *nested RTA*, *multirecovery RTA*, and *multimonitor RTA* designs. *Nested RTA* designs implement safety controls at each layer of the control architecture, as shown in Figure 8. *Multirecovery RTA* designs have multiple reversionary controllers at the same level of the control architecture and a switch that considers all possible recovery function options. The concept is discussed in a recent American Society for Testing and Materials (ASTM) standard for the use of RTA in unmanned aircraft [108] and depicted in Figure 9. The best approach to designing multiple recoveries is an area of active research. One approach is to combine multiple RTA systems as an integrated solution. For example, the Automatic Integrated Collision Avoidance System [110], [111] integrates ground and midair collision avoidance into a comprehensive collision avoidance system. The advantage of this integrated approach is that each system can consider possible conflicts among boundary violations and select a better solution than either might select on its own. For instance, in a situation where an aircraft flying close to the ground is on a collision course with another aircraft, a ground-aware midair collision avoidance system will filter

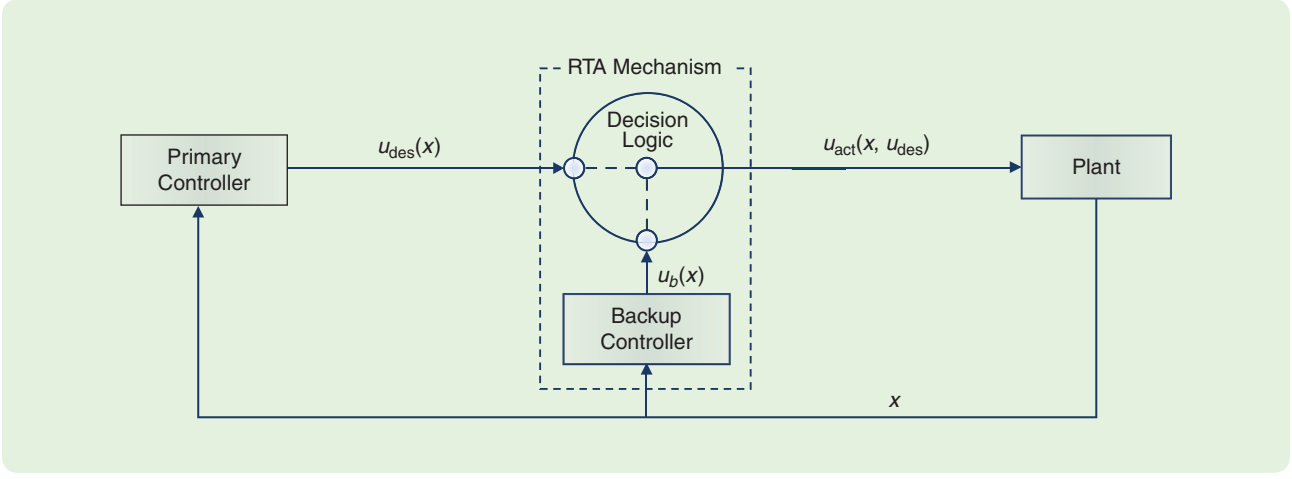

FIGURE 7 The Simplex architecture with a physical plant, a verified safety controller, a verified decision module and switch, and an unverified complex controller.

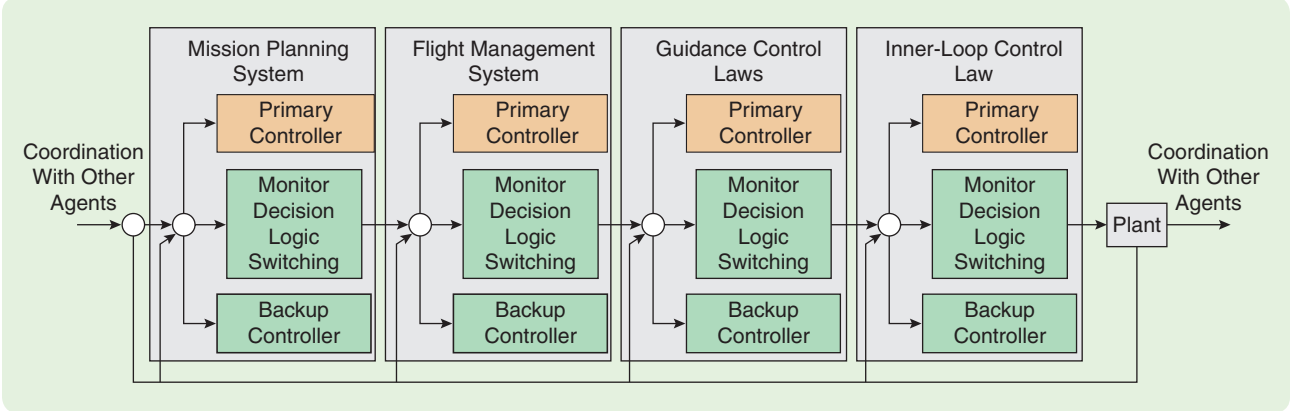

FIGURE 8 A nested feedback RTA architecture for an unmanned aerial system plant [109]. In a nested RTA architecture, the inner-loop control law RTA ensures aircraft stability, the guidance control law RTA ensures the safety of commands generated to follow waypoints, the flight management system RTA checks that the waypoints along the path are safe, and the mission planning system RTA ensures the safety of task allocations to meet mission goals of the plant. In each RTA layer, a monitor watches for unsafe conditions, and decision logic determines switching between the primary controller and backup controller.

from the possible maneuvers to select a maneuver that avoids the midair collision while, at the same time, ensuring that none of those maneuvers will fly the aircraft into the ground. An integrated solution has the benefit of generating a better solution than either separate system might otherwise choose. However, this increase in performance also comes at a large increase in cost and schedule. An alternative multiple recovery controller integration approach is to use an RTA network architecture in which each boundary monitor and recovery controller function is separate [112]. An example of an architecture with multiple safety monitors and recovery functions is the Expandable Variable Autonomy Architecture (EVAA) RTA network [112]. On the one hand, this modular approach eases the verification of each individual component and facilitates quick integration of new components and upgrades. On the other hand, additional stress is placed on the verification of the decision module/switching component to determine which action to take in the case where multiple safety boundaries may be violated at the same time. The EVAA [112] answer to this challenge is to approach the decision like a human pilot, who responds to the most critical boundary violation first before moving onto the next violation. A disadvantage of the modular approach is that it is possible to violate a safety boundary. For instance, if a ground collision and geofence collision were both imminent, the system might decide to engage ground collision avoidance to first clear the threat of the ground collision before engaging a geofence controller and violating the geofence boundary. Depending on the scenario, this may or may not be acceptable. An integrated solution could adjust the boundary of a geofence controller, based on awareness of ground and midair collision avoidance, and it could select a collision avoidance maneuver that also kept the aircraft within the geofence. *Multimonitor RTA* designs feature checks and balances to ensure the integrity of the RTA system. These designs may include monitors beyond a boundary and safety monitor. These monitors may include a *failure monitor*, an *interlock monitor*, and a *human supervisor*, among others [90]. The failure monitor watches for unreliable information coming into the RTA, such as a failed or stale sensor signal, and prevents the RTA from acting on incorrect state information. The interlock monitor watches for conditions in which it may be safe to compute a backup control solution but unsafe to engage a maneuver. For example, an interlock condition might occur when a robotic system is physically connected to another system and a rapid maneuver may result in damage to one or both robots. Finally, there are many circumstances where a cyberphysical system may be acting with a human teammate and under the supervision of a human. In these cases, a human supervisor may have roles, such as turning the RTA off, changing the RTA risk and sensitivity level, and manually activating the RTA recovery control function.

Simplex RTA Safety Monitor Approaches

As discussed previously, approaches may be considered implicit or explicit based on whether they rely on simulating backup trajectories online. Approaches to the design of a Simplex-based RTA safety monitor tend to fall in categories of *boundary violation* (i.e., explicit), *trajectory prediction* (i.e., implicit), and *reach tube prediction* (i.e., nondeterministic). As shown in Figure 12, boundary violation monitors activate a recovery controller after a safety boundary has been violated as depicted in Figure 12. These designs often feature a buffer to allow a recovery response time to complete before violating the actual safety boundary. Trajectory prediction monitors predict the trajectory that the recovery would take if it were to activate, and if that trajectory prediction would intersect the safe boundary with a small buffer, the recovery maneuver will activate [113], [114]. Reach tube-based monitors study a set of possible future states arising from uncertainty in the dynamics, and in instances where this prediction intersects the unsafe set, the RTA recovery function will activate to ensure safety [115], [116].

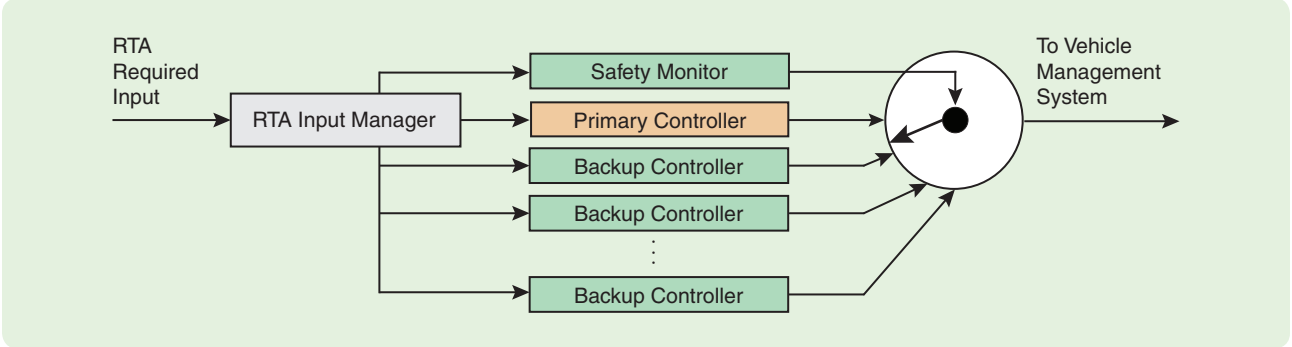

FIGURE 9 The ASTM F3269 RTA architecture with multiple recovery control functions at the same level in the control hierarchy [108]. An example of this might be an aircraft that avoids collisions with other aircraft, avoids collisions with the ground, and avoids flying into no-fly zones. Each of these boundaries is disjoint; however, it is possible that violations of multiple boundaries could be predicted to occur at the same time.

Simplex RTA as a Near-Term Certification Path

Within the past five years, progress has been made toward developing certification criteria for RTA [117] based on the Simplex architecture, resulting in the publication of the first ASTM standard for the use of RTA in unmanned aircraft [108]. A second iteration [118] of this standard opens the door for alternative filtering RTA designs.

Provably Safe RTA With Black-Box Backup Controllers

Implicit RTA systems rely on online simulations of a backup controller to assess safety. An important consequence of this architectural choice is that the backup controller need not be guaranteed to always produce a safe solution to guarantee the safety of the RTA system. That is, in the event that the backup controller fails to compute a new safe trajectory at any time, it can follow the most recently found safe trajectory until it reaches the backup set. In this sense, the job of the backup controller is to search for and *propose* candidate safe trajectories. A monitor can be used to determine whether this candidate trajectory is both feasible and safe. A practical consequence of this is that high-performing black-box controllers can safely be used to generate backup trajectories. This concept has been used in practice [18] and is thoroughly explored for the context of Simplex-based systems in [119]. Furthermore, the idea extends to other methods of RTA. For example, in [54], ASIF is constructed that utilizes a backup controller taking the form of a neural network approximation of an optimal control policy.

Canonical Algorithms

Included in this section is a set of canonical algorithms for Simplex-based RTA. In this case, we consider the deterministic discrete-time system

$$x(i+1) = F(x(i), u(i)) \tag{25}$$

where $x \in X \subseteq \mathbb{R}^n$ denotes the state, $u \in U \subset \mathbb{R}^m$ denotes the control input, and $i \in \mathbb{Z}$ denotes the time index. A constraint set is defined as $C_A \subseteq X$. It is assumed that a backup control law $u_b : X \to U$ is defined, and we let $\phi^{u_b}(i; x)$ be the state of (25) reached at $i \geq 0$ when beginning at state x at time $i = 0$ and evolving under u_b.

The idea behind the algorithms in this section is to assess the safety of a *probe step*. That is, at any state $x_{curr} \in X$, whether to accept or reject the desired control input $u_{des} \in U$ is based on whether the candidate next state reached under this input $x_{cand} := F(x_{curr}, u_{des})$ is safe. In the first case, an explicit safe set is assumed to be defined. The second case considers an implicit approach, which relies on online simulations under the backup dynamics. A deterministic discrete-time system is assumed for the clarity of the algorithms and their associated guarantees. However, the algorithms may be adapted to continuous-time systems, multiple backup controllers, latched implementations, and so on.

Explicit Simplex Filter

Algorithm 1 describes an explicit Simplex-based RTA algorithm, the Region-Based Simplex Filter (RBSF). In addition to a backup control law u_b, it is assumed that a safe set $C_S \subseteq C_A$ is defined such that C_S is invariant under u_b. The algorithm renders the RTA system forward invariant in C_S. This is apparent through the observations that 1) by the nature of the invariance property, applying u_b anywhere in C_S will lead to a next state that is still in the set and 2) the sample step allows for inputs that would cause a departure from C_S to be detected and replaced with u_b. In practice, one may choose to add conservatism to the approach by working with any underapproximation $\tilde{C}_S \subseteq C_S$. The region $C_S \setminus \tilde{C}_S$ is often referred to as a *buffer*. A practical consideration is that the backup controller should locally attract solutions outside of $\tilde{C}_S$ to $\tilde{C}_S$. That is, it is favorable to ensure that a recovery is possible in the case where the state is perturbed outside of the safe region. One way to do this is to ensure that all initial conditions in the buffer will lead

ALGORITHM 1 The RBSF.

Input : Current state $x_{curr} \in X$
: Desired input $u_{des} \in U$
Output : Safe control input $u_{act} \in U$
Predefined: Constraint set $C_A \subset X$
: Invariant set $C_S \subset C_A$
: Backup control law $u_b : X \to U$.

function $u_{act} = \text{RBSF}(x_{curr}, u_{des})$
  $x_{cand} \leftarrow F(x_{curr}, u_{des})$
  if $x_{cand} \in C_S$ **then**
    return u_{des}
  else
    return u_b

ALGORITHM 2 The SBSF.

Input : Current state $x_{curr} \in \mathbb{R}^n$
: Desired input $u_{des} \in U$
Output : Safe control input $u_{act} \in U$
Predefined: Constraint set $C_A \subset X$
: Invariant backup set $C_b \subseteq C_A$
: Backup control law $u_b : X \to U$.

function $u_{act} = \text{SBSF}(x_{curr}, u_{des})$
  $x_{cand} \leftarrow F(x_{curr}, u_{des})$
  compute: $\phi^{u_b}(i; x_{cand}) \quad \forall i \in \{0, \ldots, N\}$
  if $\phi^{u_b}(i; x_{cand}) \in C_A \quad \forall i \in \{0, \ldots, N-1\}$ AND $\phi^{u_b}(N; x_{cand}) \in C_b$ **then**
    return u_{des}
  else
    return u_b

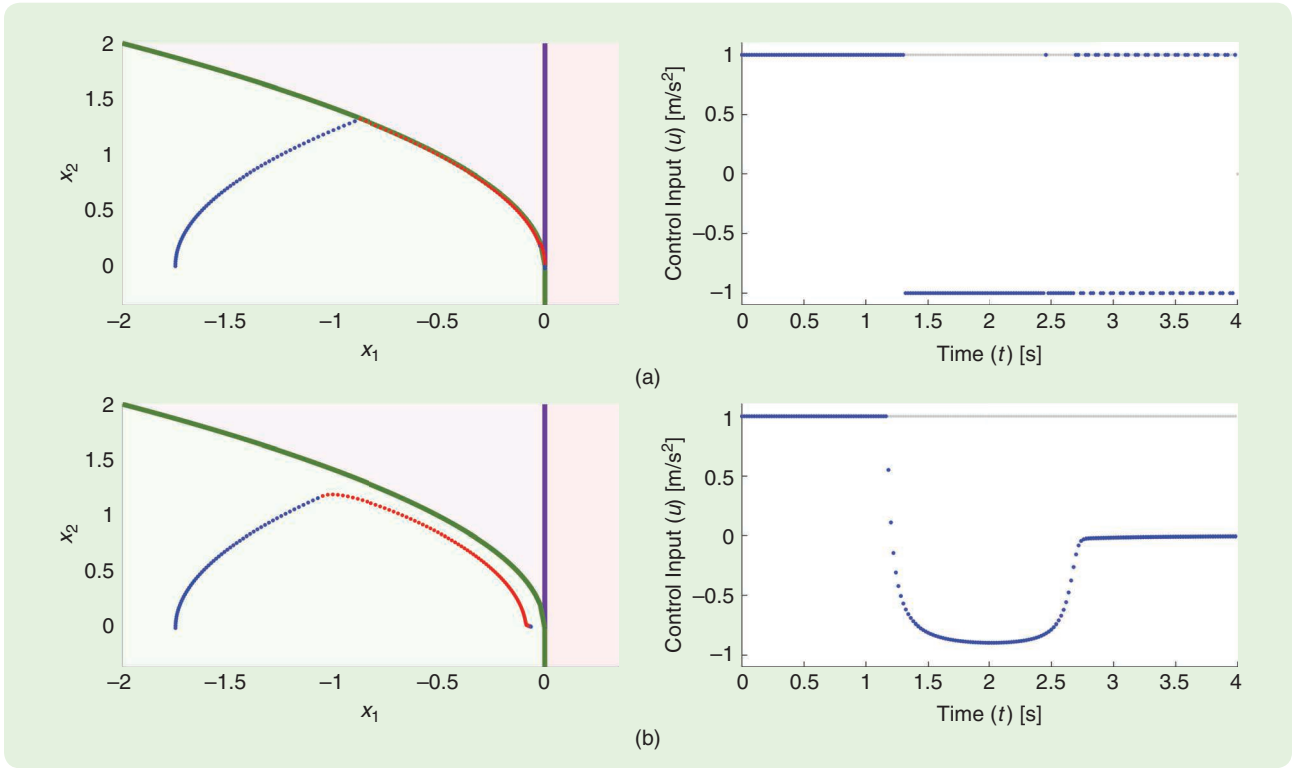

FIGURE 10 A comparison of explicit RTA algorithms on double-integrator system (6) with constraint set (7) and $U \in [-1, 1]$. The viability kernel is shaded green, collision states are shaded red, and the inevitable collision states are shaded purple. In the figures on the left, x_1 represents the distance to the collision state boundary, x_2 represents the velocity relative to the collision state boundary, the state under the desired control from the primary controller is blue, the state under the filtered control is red, and safety constraint boundaries are green. The figures on the right show the actual control output of the filtered signal. (a) The explicit simplex filter switches from primary control to filtered control according to Algorithm 1. (b) The explicit active set invariance filter offers a more gradual intervention that solves a quadratic program to minimize the difference of the actual control signal from the desired control signal of the primary controller while assuring safety according to Algorithm 3.

to $\tilde{C}_S$ if u_b is applied; for example, this is the case when C_S and $\tilde{C}_S$ are both sublevel sets of a Lyapunov function.

Example

Consider the double-integrator system described by (6), discretized over a period of 0.1 s, and consider the constraint set (7). The backup control law $u_{des} = -1$ renders the set $C_S = \{x \in \mathbb{R}^2 \,|\, -2x_1 - x_2^2 \geq 0\}$ forward invariant, and it is clear that $C_S \subseteq C_A$. Figure 10(a) is a simulation of the RBSF algorithm with these parameters under the desired control $u_{des} = 1$. ◊

Implicit Simplex Filter

Algorithm 2 describes an implicit Simplex-based algorithm, the simulation-based Simplex filter (SBSF). It is assumed that a backup set $C_b \subseteq C_A$ is defined such that C_b is invariant under a backup control law $u_b : X \to U$. No assumptions are made on the structure of u_b; it may come from a closed-form control expression, a black-box function, MPC, a motion primitive, and so on. In contrast to the RBSF algorithm, which constrains the system to evolve in the explicitly defined invariant region, the SBSF uses the smaller invariant set C_b as a means to access a larger implicitly defined invariant set. Specifically, the SBSF constrains the system to the SBI of C_b

$$C_S = \{x \in X \,|\, \phi^{u_b}(i; x) \in C_A \ \forall i \in \{0, \ldots, N-1\}, \phi^{u_b}(N; x) \in C_b\} \tag{26}$$

where $\phi^{u_b}(i; x)$ is the ith point in an $N + 1$ point trajectory obtained from simulating under the dynamics (25) and backup controller u_b. Since C_S is forward invariant under

ALGORITHM 3 The explicit ASIF.

Input : Current state $x_{curr} \in X$
: Desired input $u_{des} \in U$
Output : Safe control input $u_{act} \in U$
Predefined: Allowable set $C_A \subset X$
: Invariant Set $C_S \subset C_A$
: Extended class κ_∞ function $\alpha : \mathbb{R} \to \mathbb{R}$.

1: **function** $u_{act} = \text{EASIF}(x_{curr}, u_{des})$
2: **solve:** (28) subject to the constraint (36)
3: **return** $u_{act}(x_{curr}, u_{des})$

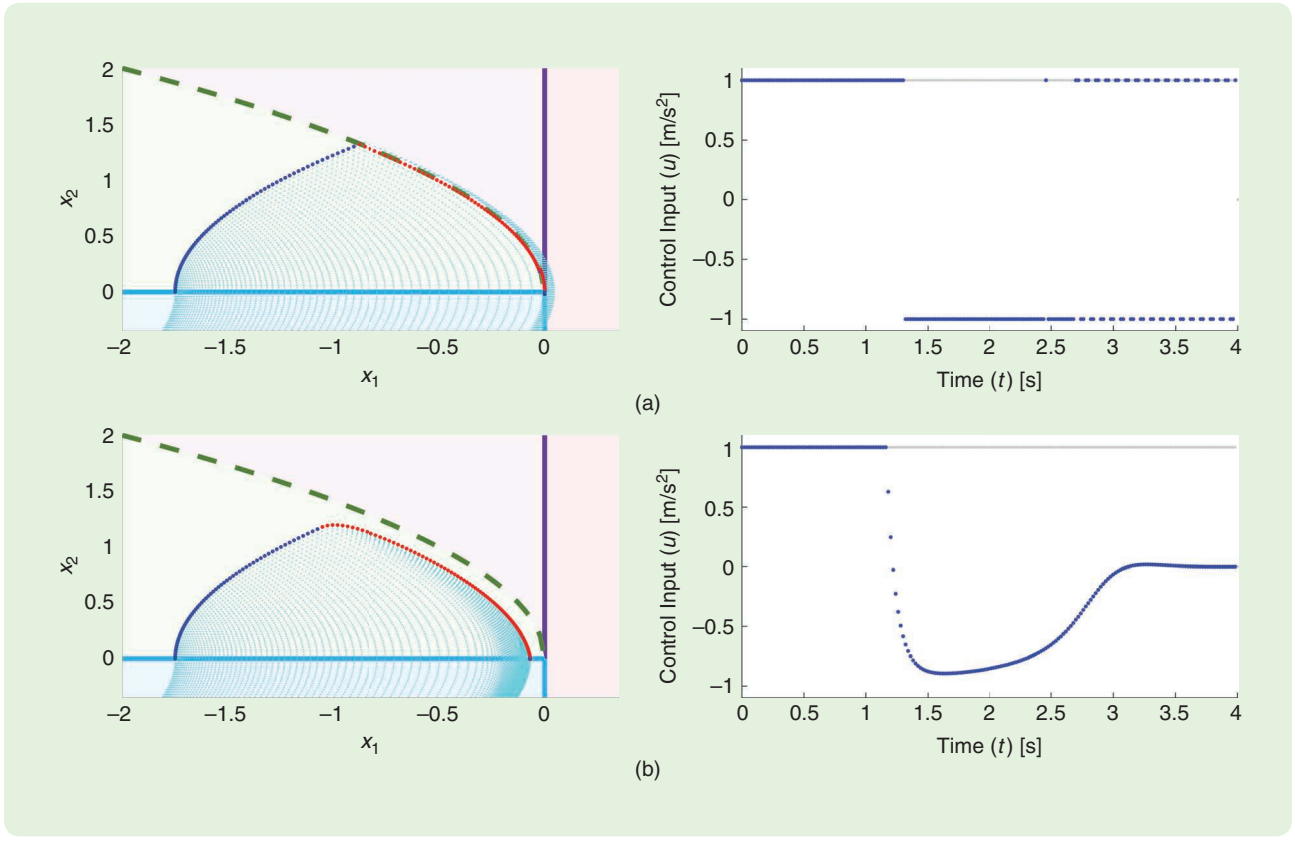

FIGURE 11 A comparison of implicit RTA algorithms on double-integrator system (6) with constraint set (7) and $U \in [-1, 1]$. The viability kernel is shaded green, collision states are shaded red, and the inevitable collision states are shaded purple. In the figures on the left, x_1 represents the distance to the collision state boundary, x_2 represents the velocity relative to the collision state boundary, the state under the desired control from the primary controller is blue, the state under the filtered control is red, implicit backup trajectories are cyan, and safety constraint boundaries are green. The figures on the right show the actual control output of the filtered signal versus the simulation time. (a) The implicit simplex filter switches from primary control (full acceleration) to backup control (full deceleration) as predicted by the path of the system under the backup control according to Algorithm 2. (b) The implicit active set invariance filter computes a backup trajectory by integrating under a backup control law and solving the quadratic program under implicit barrier constraints according to Algorithm 4.

ALGORITHM 4 The implicit ASIF.

Input : Current state $x_{\text{curr}} \in \mathbb{R}^n$
: Desired input $u_{\text{des}} \in U$
Output : Safe control input $u_{\text{act}} \in U$
Predefined: Allowable set $C_A \subset X$
: Invariant backup set $C_b \subseteq C_A$
: Smooth backup control law $u_b : X \to U$
: Extended class κ_∞ function $\alpha : \mathbb{R} \to \mathbb{R}$.

1: **function** $u_{\text{act}} = \text{IASIF}(x_{\text{curr}}, u_{\text{des}})$
2: **compute:** $\phi^{u_b}(t_b; x_{\text{curr}})$ $\forall t_b \in \{0, t_1, \ldots, t_{N-1}, T_b\}$ by integrating (27) under u_b
3: **evaluate:** $Df_{\text{cl}}(\phi^{u_b}(t_b; x_{\text{curr}}))$ $\forall t_b \in \{0, t_1, \ldots, t_{N-1}, T_b\}$
4: **compute:** $D\phi^{u_b}(t_b, x_{\text{curr}})$ $\forall t_b \in \{0, t_1, \ldots, t_{N-1}, T_b\}$ by integrating (48)
5: **solve:** (28) subject to the constraints (46) and (47)
6: **return** $u_{\text{act}}(x_{\text{curr}}, u_{\text{des}})$

u_b, the algorithm renders the RTA system forward invariant in C_S. As in the case of the RBSF algorithm, this becomes apparent from the observations that 1) applying u_b anywhere in C_S will lead to a state that is also in C_S and 2) the sample step under u_{des} can reliably be used to detect inputs that would cause a departure from C_S. Conservatism can be added to this approach by using underapproximations of C_A and C_b. Note that the preceding algorithm assumes a deterministic system and relies on simulating single trajectories under the backup dynamics. This can be extended to the nondeterministic case by simulating an overapproximation of the forward reachable set under the backup dynamics and requiring that C_b is robustly forward invariant.

Example

Consider the double-integrator system described by (6), discretized over a period of 0.1 s, and consider the constraint set (7). The backup control law $u_{\text{des}} = -1$ renders the set $C_b = \{x \in \mathbb{R}^2 \mid -x_1 \geq 0, -x_2 \geq 0\}$ forward invariant, and

it is clear that $C_b \subseteq C_A$. Figure 11(a) presents a simulation of the SBSF algorithm with these parameters under the desired control $u_{des} = 1$.

Example

From [58], consider the system (21) discretized in time to have a 1-s resolution and with $C_A = \{x \in \mathbb{R}^5 \mid \|r\|_\infty - r_{min} \geq 0, \kappa_1 + \kappa_2 \|r\|_\infty - \|v\|_\infty \geq 0\}$, where $r := [x_1, x_2]^T$ and $v := [x_3, x_4]^T$ are the relative position and speed vectors, respectively.

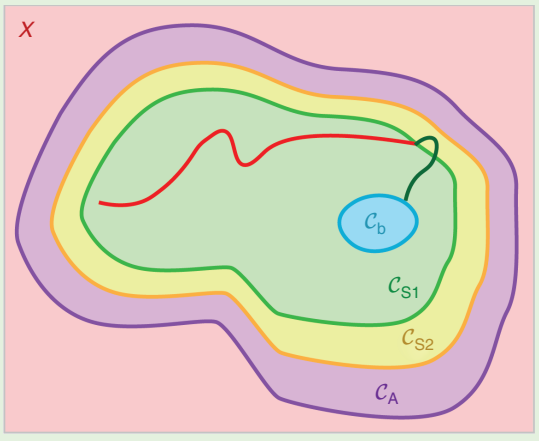

FIGURE 12 An RTA "amoeba" diagram depicts the entire state space X, containing the set of allowable states C_A in purple, a safety buffer region C_{S2}, and a safe zone with without switching C_{S1}. The trajectory of a primary controller is depicted in red. When the trajectory crosses the switching condition boundary between C_{S1} and C_{S2}, the RTA activates the backup controller, and the green trajectory shows backup control to a safe region C_b. Image inspired by [109].

The backup controller u_b is taken as the first input in the solution to a mixed-integer quadratic program (MIQP) solved subject to the constraints that 1) all intermediate points lie in C_A and 2) the endpoint lies in a set C_b that is defined from the set of natural motion trajectories lying outside of C_A (Figure 13). In this case, $x \in C_S$ is true whenever the MIQP has a feasible solution with initial state x. Figure 13 shows a simulation of the RTA system with a primary controller that is designed to drive the system to an invariant point along the y-axis. The simulation uses parameters $m = 50$ kg, $n = 0.001027$ rad/s, $u_{max} = 0.5$ N, $r_{min} = 0.5$ km, $\kappa_1 = 0.5$ m/s, and $\kappa_2 = 2n$ s^{-1}. ◊

EXPLICIT AND IMPLICIT ASIF

ASIF methods are RTA approaches associated with a class of first-order optimization-based algorithms. As with many Simplex-based techniques, they are system agnostic and provably correct, and they exhibit a number of robustness properties that make them attractive in practice. In addition, ASIF methods are minimally invasive with respect to the safety constraints. This results in a smoother and more gradual intervention than with approaches that rely on switching to backup controllers directly.

This section considers systems of the form

$$\dot{x} = f(x) + g(x)u. \tag{27}$$

When (1) is of the form (27), the dynamics are said to be *control affine*, and it is assumed throughout the following that $f(x): \mathbb{R}^n \to \mathbb{R}^n$ and $g(x): \mathbb{R}^n \to \mathbb{R}^{n \times m}$ are locally Lipshitz continuous in their inputs so that, in particular, solutions to

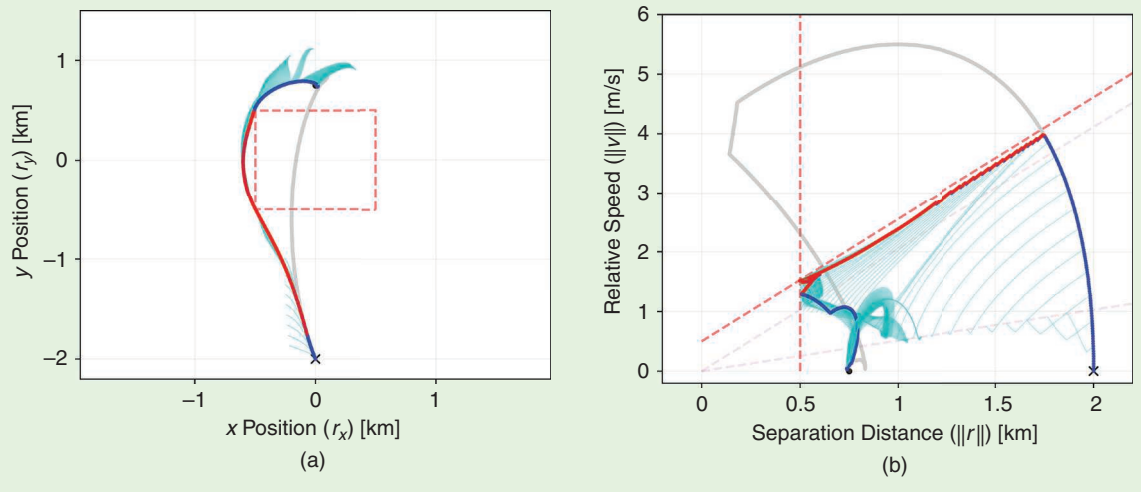

FIGURE 13 The projected trajectories of a chaser spacecraft relative to a target spacecraft at the origin under Clohessy–Wiltshire–Hill dynamics, described by (21), over a 2,400-s period. For safety, the chaser spacecraft is required to stay (a) at least 0.5 km away from the target in both the x and y directions (shown by dashed red lines) and (b) below a dynamic velocity limit (shown by dashed red lines), where the spacecraft's relative velocity must slow down as it gets closer. The chaser spacecraft trajectories without RTA are gray. The trajectories with am SBSF RTA are blue when $u_{act} = u_{des}$ and red when $u_{act} = u_b$. The backup trajectories (cyan) are shown at 8-s intervals.

(27) are unique when they exist. ASIF approaches are constructed as QPs, where the objective function is typically the l^2 norm of the difference between the desired input u_{des} and actual input u_{act} and where the constraints on the program, which are known as *barrier constraints*, take the form $BC_i(x,u) = a_i(x)u + b_i(x) \geq 0$, $i \in \{1,...,N\}$. The canonical ASIF controller is given in the following as an ASIF-QP:

ASIF-QP

$$u_{\text{act}}(x, u_{\text{des}}) = \text{argmin}_{u \in U} \| u - u_{\text{des}} \|_2^2 \tag{28}$$

$$\text{s.t. } BC_i(x, u) = a_i(x)u + b_i(x) \geq 0, \quad i = 1,...,N. \tag{29}$$

The preceding ASIF-QP guarantees system safety with respect to an operational region C_S when the properties ϑ_1^{BC} and ϑ_2^{BC} are satisfied by the barrier constraints:

- ϑ_1^{BC}: If (29) is satisfied, the system is forward invariant in some set $C_S \subseteq C_A$; i.e.,

$$\vartheta_1^{BC}: BC_i(x, u) \geq 0, \forall i \in \{1,...,N\} \Rightarrow C_S \text{ is forward invariant} \tag{30}$$

  where $C_S \subseteq C_A$.
- ϑ_2^{BC}: It is possible to satisfy (29) from any state in the set C_S; i.e., $\forall x \in C_S$, $\forall i \in \{1,\ldots,N\}$, and

$$\vartheta_2^{BC}: \sup_{u \in U} [BC_i(x, u)] \geq 0. \tag{31}$$

The first property ensures that the RTA mechanism will bound trajectories to the set C_S as long as there exists a feasible input satisfying the barrier constraint. The second property ensures that for all states in C_S, it is possible to find a feasible input that satisfies the barrier constraint. Since QPs can be solved in such a way that feasible solutions are found whenever they exist, guaranteeing the existence of a solution is sufficient for ensuring the feasibility of the ASIF-QP. When both properties are satisfied, C_S defines a safe set with respect to the ASIF-QP. Moreover, when the barrier constraints satisfy the preceding properties, the ASIF-QP is *minimally invasive* in the sense that the l^2 distance between u_{act} and u_{des} is minimized, subject to the barrier constraints.

As with the case of the Simplex, implicit and explicit approaches emerge for ASIF methods, based on how the safe operational region C_S is identified. First, *explicit* ASIF methods are studied. In this case, C_S is obtained by identifying a control invariant set explicitly as the superzero-level set of a smooth function. Next, in the case of *implicit* ASIF approaches, a control invariant set is identified implicitly through the trajectories of a backup control law. A more thorough review of explicit ASIF approaches is provided in [120], and implicit ASIF approaches are considered in [53]. See [49] for a review of both implicit and explicit ASIF approaches.

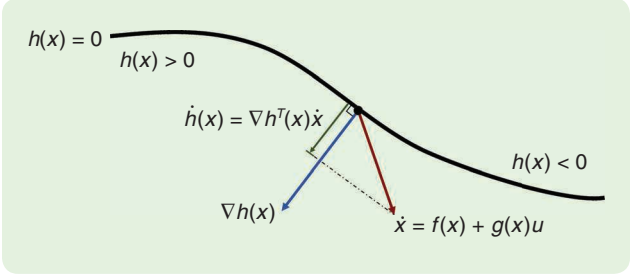

FIGURE 14 The barrier geometry for constraint function $h: \mathbb{R}^2 \to \mathbb{R}$.

Barrier Constraints From an Explicitly Defined Safe Set

Consider a set C_S that is defined as the superzero-level set of a continuously differentiable function $h: \mathbb{R}^n \to \mathbb{R}$ (Figure 14), i.e.,

$$C_S = \{x \in \mathbb{R}^n \mid h(x) \geq 0\} \tag{32}$$

$$\partial C_S = \{x \in C_S \mid h(x) = 0\} \tag{33}$$

and suppose that zero is a regular value of h; that is, the gradient of h does not vanish on the boundary. The goal is to design a barrier constraint $BC(x, u)$ that satisfies ϑ_1^{BC} and ϑ_2^{BC} and consequently makes the set C_S forward invariant under the ASIF-QP. One method for ensuring the forward invariance of C_S involves checking the subtangentiality condition over the boundary of C_S. Sometimes called Nagumo's condition, this result is as follows: if

$$L_f h(x) + L_g h(x) u(x) \geq 0 \tag{34}$$

holds for all $x \in \partial C_S$, C_S is forward invariant for the closed-loop dynamics of (27) under $u(x)$, where L_f and L_g in (34) denote Lie derivatives of h along f and g, respectively. Informally, (34) ensures that $\dot{h}(x) \geq 0$ for all $x \in \partial C_S$, and thus, any control law $u(x)$ satisfying (34) ensures the forward invariance of C_S when applied to (27).

The constraint (34) is not very practical for use directly in an optimization framework, as it is activated exactly on the boundary of C_S, which is a region without volume. The solution presented in [121] is to use a similar condition that is active everywhere in C_S and includes a strengthening term that relaxes the constraint away from the boundary: for all $x \in C_S$,

$$L_f h(x) + L_g h(x) u(x) + \alpha(h(x)) \geq 0 \tag{35}$$

where $\alpha: \mathbb{R} \to \mathbb{R}$ is an extended class κ_∞ function. A continuous function $\alpha: \mathbb{R} \to \mathbb{R}$ is class κ_∞ if α is strictly increasing and $\alpha(0) = 0$. Since $\alpha(h(x)) = 0$ on the boundary, the satisfaction of (35) implies the satisfaction of (34). Hence, (35) implies that C_S is forward invariant for the closed-loop dynamics of (27) under $u(x)$. For this reason, an attractive method for constructing the ASIF methods (28) and (29) is to form barrier constraints using

$$BC(x,u) = L_f h(x) + L_g h(x) u + \alpha(h(x)). \tag{36}$$

Applying the previous theoretical results, the resulting ASIF approach ensures the forward invariance of C_S and therefore satisfies ϑ_1^{BC}, as discussed in the preceding.

To satisfy the requirement ϑ_2^{BC}, the functions h and α must be chosen in such a way that ensures that there always exists $u \in U$ satisfying the barrier constraint; equivalently,

$$\sup_{u \in U} [L_f h(x) + L_g h(x) u] \geq -\alpha(h(x)) \tag{37}$$

must hold for all $x \in C_S$. Although it is, on occasion, possible to obtain h and α directly through intuitive reasoning, satisfying this requirement is accomplished, in general, through the computation of a *control invariant* subset of C_A. Specifically, if C_S is a control invariant set, it is always possible to find a strengthening function $\alpha(x)$ such that the viability property is satisfied [80]. In practice, α often takes the form of a linear or odd polynomial function, and it can, to an extent, be chosen in a way that shapes the response of the RTA mechanism to have some desired behavior. For example, a common tradeoff that occurs when shaping this function is in choosing between a more gradual response that activates further inside the safe set and a more abrupt intervention activating closer to the boundary of C_S. Alternatively, one may bypass choosing α altogether with a relaxed QP formulation (see [49] and [80] for more details). Identifying a large control invariant subset $C_S \subseteq C_A$ is the key challenge in implementing this method.

Example

Consider the double-integrator system described by (6) and the constraint set (7). A control invariant set is given by the superzero-level set of $h(x) = -2x_1 - x_2^2$, and given a class κ_∞ function α, the barrier constraint is $-2x_2(1+u) + \alpha(-2x_1 - x_2^2) \geq 0$. Note that $\forall x \in C_S$, $\alpha(h(x)) \geq 0$ and $\sup_{u \in U}[L_f h(x) + L_g h(x) u] = 0$; hence, the barrier constraint is viable. ◊

Example

Consider the dynamics of a spacecraft in unconstrained rotation

$$\dot{x} = J^{-1}(-x \times Jx) + J^{-1}u \tag{38}$$

where $x \in \mathbb{R}^3$ represents the angular velocity, $J = \text{diag}(J_1, J_2, J_3) \in \mathbb{R}^{3\times 3}$ is a diagonal inertia matrix, and $u \in [-1,1]^3$. A control invariant set for this system is given by $C_S = \{x \in \mathbb{R}^3 \mid h(x) \geq 0\}$, with

$$h(x) = K - x^T Jx \tag{39}$$

for any $K \in \mathbb{R}_+$. The fact that this set is control invariant is made apparent with the observation that the rotational kinetic energy $x^T Jx$ is constant when $u = 0$. Choosing $\alpha(x) = x$, the barrier constraint is $\nabla h^T(x)\dot{x} + h(x) \geq 0$, which simplifies to

$$K - x^T Jx - 2x^T u \geq 0. \tag{40}$$

It is easy to show that this constraint is viable everywhere within C_S. Namely, $K - x^T Jx \geq 0$ everywhere in C_S, and one can always find $u \in U$ such that $-2x^T u \geq 0$. ◊

If there exists an extended class κ_∞ function satisfying (37) for all $x \in X$, $h(x)$ is said to be a *CBF* [120], [122], [123]. Barrier functions were first developed in the context of nonlinear and hybrid system verification [43], [124], [125]. The idea was first extended to the RTA problem with QPs in [126], and it was developed more in [122]. In recent years, there has been a surge in interest in this topic. The theory has been extended to relax assumptions on smoothness [127] to discrete-time [128] stochastic [129] and nondeterministic systems and enforce temporal logic specifications [130], [131], [132]. Likewise, the practicality of the approach is demonstrated in a variety of real-world applications, as shown in Table 2. It is important to note that for (36) to depend on the control input

TABLE 2 **The classification of RTA applications in the literature.**

	Implicit	Explicit
Simplex	Spacecraft collision avoidance [27]	Waypoint tracking cyberphysical system [60]
	Fixed-wing aircraft ground avoidance (Auto-GCAS) [61], [62]	Advanced flight control systems [63]
	Fixed-wing aircraft obstacle avoidance [22], [23], [64]	Lane keeping [65]
	Safety against linear temporal logic specifications [66], [67]	Aircraft conflict resolution [68], [69], [70]
	Collision avoidance for high-speed quadrotor navigation [56], [71]	Decentralized collision avoidance for quadrotors [72], [73]
	Collision avoidance during lane-changing maneuvers [74]	Ground robot navigation around obstacles [75], [76]
ASIF	Bipedal walking/exoskeletons [77]	Bipedal walking/exoskeletons [78], [79]
	Robotic manipulator arms [28]	Lane keeping [14], [15]
	Mobile inverted pendulum (Segway) [54]	Mobile inverted pendulum (Segway) [80]
	Rapid aerial exploration of unknown environments [26]	Robotic grasping [81]
	Multirobot systems [82]	Swarm robotics (Robotarium) [83], [84], [85], [86]
	Fixed-wing aircraft collision avoidance [87]	Motorized rehabilitative cycling system [88]

u, the constraint function $h(x)$ must have a relative degree of one; that is, $L_g h(x) \neq 0$. Safety constraint functions with higher relative degrees are addressed in the literature on exponential CBFs [133] and high-order CBFs [134], [135].

Example

Consider the system $\ddot{x} = u$, with $u \in U = (-\infty, \infty)$, and constraint function $h(x) = -x_1$. Note that while actuation is unlimited in this case, it is not possible to enforce the barrier constraint $L_f h(x) + L_g h(x) u \geq \alpha(h(x))$, as $L_g h(x) = 0$. ◊

Barrier Constraints From an Implicitly Defined Safe Set

As discussed in the previous section, the barrier constraint is viable when it restricts the system to a control invariant set. If an explicit representation of a large control invariant subset of C_A can be obtained, with the appropriate choice of α, a valid barrier constraint is obtained through (36). The key idea behind the implicit approach to ASIF [53], [54], [80] is to filter with respect to an implicitly defined control invariant set. Specifically, given a backup control law u_b and backup set C_b that is invariant under u_b, one can develop barrier constraints that activate near the boundary of the SBI of C_b; see (17). The idea was first proposed in [80] and further developed in [53] and [54]. A tutorial on the topic is provided in [87], and some notable applications are listed in Table 2. A key advantage to this approach is that it does not require the computation of a large control invariant set.

Consider a smooth backup control law $u_b : \mathbb{R}^n \to U$, and suppose that the constraint set is given by

$$C_A := \{x \in \mathbb{R}^n \mid \varphi(x) \geq 0\} \tag{41}$$

and a backup set $C_b \subseteq C_A$ is given by

$$C_b = \{x \in \mathbb{R}^n \mid h_b(x) \geq 0\}. \tag{42}$$

The SBI of C_b is

$$C_S = \{x \in \mathbb{R}^n \mid \forall t \in [0, T_b],\ \varphi(\phi^{u_b}(t;x)) \geq 0 \wedge h_b(\phi^{u_b}(T_b;x)) \in C_b\} \tag{43}$$

where $\phi^{u_b}(t;x)$ is the flow of (27) under u_b, evaluated t units of time from x. Given an extended class κ_∞ function α, it can be shown that the following constraints are sufficient for invariance in C_S:

$$\frac{d\varphi(t_b;x))}{dt} + \alpha(\varphi(\phi(t_b;x))) \geq 0 \tag{44}$$

$$\frac{dh_b(\phi(T_b;x))}{dt} + \alpha(h_b(\phi(T_b;x))) \geq 0 \tag{45}$$

for all $t_b \in [0, T_b]$. The constraint (44) enforces that the points along the backup trajectory stay in the constraint space C_A, and (45) enforces that the endpoint of the backup trajectory stay in C_b. Expanding the derivative terms yields the implicit ASIF barrier constraints

$$\nabla\varphi(\phi^{u_b}(t_b;x))\, D\phi^{u_b}(t_b;x)[f(x) + g(x)u] + \alpha(\varphi(\phi^{u_b}(t_b;x))) \geq 0 \tag{46}$$

$$\nabla h_b(\phi^{u_b}(T_b;x))\, D\phi^{u_b}(T_b;x)[f(x) + g(x)u] + \alpha(h_b(\phi^{u_b}(T_b;x))) \geq 0 \tag{47}$$

for all $t_b \in [0, T_b]$. Note that since t_b lives in the interval $[0, T_b]$, (46) represents an uncountable number of constraints. In practice, one may approximate the constrained set of states by numerically integrating (27) forward under u_b and evaluating $\phi^{u_b}(t;x)$ at discrete times $t_b \in \{0, t_1, \ldots, t_{N-1}, T_b\}$. It is shown in [53] how the finite set of intermediate constraints may be tightened in such a way that is sufficient for satisfying the constraints with an infinite number of trajectory points. One may compute $D\phi^{u_b}(t_b;x)$ by integrating, along with (27), a *sensitivity* matrix $Q(t_b,x) = D\phi^{u_b}(t_b;x)$. As explained in [136], $Q(t_b,x)$ is obtained as the solution to the following differential equation:

$$\frac{dQ(t_b,x)}{dt_b} = Df_{cl}(\phi^{u_b}(t_b;x))\, Q(t_b,x) \tag{48}$$

with $Q(0,x) = I_{n\times n}$ and where $Df_{cl}(\phi^{u_b}(t_b;x))$ is the Jacobian of the closed-loop dynamics of (27) under the control law $u_b(x)$, evaluated at $\phi^{u_b}(t_b;x)$. For this term to be defined, it is necessary that the control law u_b be smooth. Thus, it is often necessary to consider smooth saturation functions for bounding the control law to the admissible domain U.

Example

Considered here is finite-time safety (collision avoidance) for two vehicles having combined dynamics

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -b_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -b_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{49}$$

where x_1 and x_3 denote positions and x_2 and x_4 denote velocities for the vehicles, $b_1 = 0.1$, and $b_2 = 0.25$. The safety constraint is collision avoidance $\varphi(x) = x_3 - x_1$, and the constraint set is $C_A = \{x \in \mathbb{R}^4 \mid \varphi(x) \geq 0\}$. To simplify the analysis, finite-time safety is considered, with the endpoint constraint (47) being omitted. The backup controller $u_b = [-1, 1]^T$ is integrated over a 10-s horizon, and $\alpha(x) = 2x$. Figure 15 has the result of a simulation with $u_{des}(t) = [1 - \exp(-0.1t), \exp(-0.25t) - 1]^T$.

Example

Consider the system given by (38) and constraint set $C_A = \{x \in \mathbb{R}^3 \mid \varphi(x) \geq 0\}$, with

$$\varphi(x) = \omega_{max}^2 - x_1^2 - x_2^2 - x_3^3 \tag{50}$$

where $\omega_{\max} \in \mathbb{R}_+$ represents the maximum allowable angular speed. It can be shown that this set is not itself forward invariant, in general. The backup control law

$$u_b = \tanh((x \times Jx) - k_d Jx) \tag{51}$$

with $k_d \geq 0$, is bounded to $[-1,1]$ and stabilizes system (38) to the origin. Furthermore, it can be shown that $C_b = \{x \in \mathbb{R}^3 | K - x^T Jx \geq 0\}$ is invariant under $u_b(x)$ [see from the previous example that this is invariant under the control law $u(x) \equiv 0$]. The largest ellipsoid of this form that is contained in C_A is obtained by choosing $K = \omega_{\max}^2 / \min(J_1, J_2, J_3)$.

Figure 16 provides an illustration of the filtered trajectory under the (unsafe) primary controller $u_d(x) = [\sin(t/2), \sin(t/2 - \pi/4), \sin(t/4 - \pi/4)]^T$ and with $\omega_{\max} = 1$ m/s, $k_d = 1$, $J_1 = 12$ kgm^2, $J_2 = 12$ kgm^2, $J_3 = 5$ kgm^2, and backup horizon $t_b \in \{0, 0.05, \ldots, 3\}$. The trajectory is colored blue where $u_{des} = u_{act}$ and red where the RTA mechanism is active. ◊

Additional Design Considerations

Recall that when the properties ϑ_1^{BC} and ϑ_2^{BC} are satisfied, a set C_S is forward invariant under the RTA control law u_{act} given by the ASIF-QP. While guarantees exist under the assumptions of the model—and, indeed, the formulations

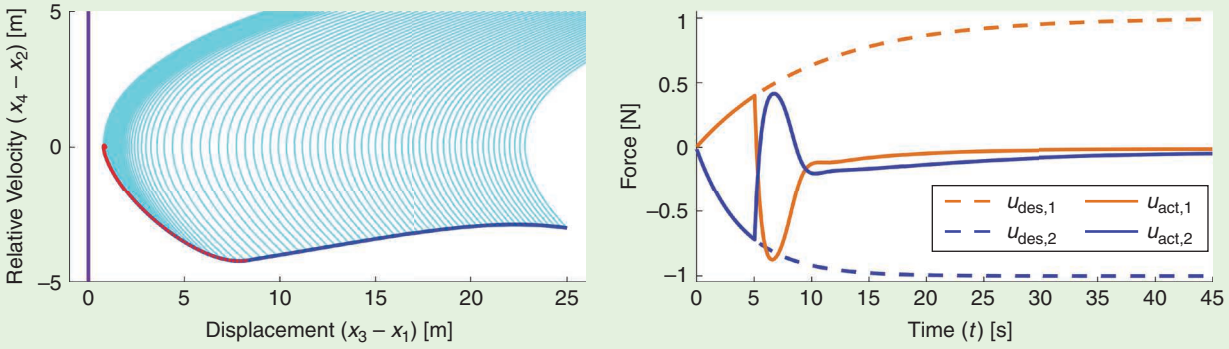

FIGURE 15 A simulation of the implicit ASIF algorithm for the cooperative collision avoidance of two vehicles.

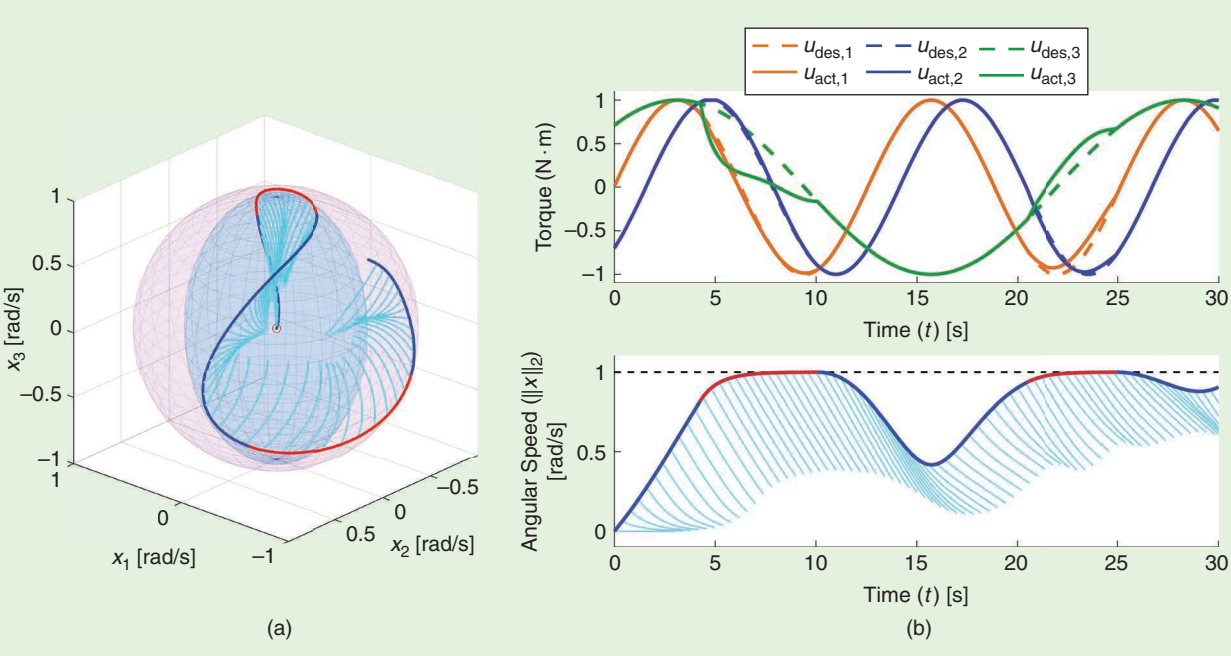

FIGURE 16 (a) A simulation of an implicit ASIF algorithm for spacecraft angular velocity dynamics. The constraint set C_A is represented by the purple sphere, and the backup set C_b is represented by the blue ellipsoid. The surface of this ellipsoid has constant kinetic energy. (b) The bottom-right plot projects this information into normed space, and the top-right plot shows the desired and actual control values.

presented exhibit some robustness properties—it is desirable in practice to consider the behavior of the system in the case where an unmodeled effect perturbs the state outside of C_S. A simple solution is to make C_S a locally attractive set by switching to a stabilizing backup controller whenever $x \notin C_S$. Alternatively, for explicit ASIF, *if h is a concave function*, the barrier constraint will cause u_{act} to attract solutions to the interior of C_S whenever the ASIF-QP is feasible. A backup controller should be present for the case where a feasible solution is not found. While beyond the scope of this article, it is interesting to note that attracting solutions outside of C_S to C_S does not necessarily correspond to the best response of the physical system. For instance, [137, Sec. V.B] presents a method for minimizing a function of *damage* in the presence of an inevitable collision.

DISCUSSION AND COMPARISON OF APPROACHES ON DOUBLE-INTEGRATOR SYSTEM

Figure 11 gives a comparison of the four algorithms presented in this article on the double-integrator system given by (6), with $u \in [-1, 1]$ and $C_A = \{x \in X | -x_1 \geq 0\}$ and a controller update period of 10 Hz. For the case of the Simplex algorithms, the continuous dynamics are discretized to fit this period. The simulations begin at state $x(0) = [-1.75, 0]^T$ and have desired input $u_{des} = 1$. Here, it can be seen that both Simplex-based algorithms exhibit nearly identical behavior, and likewise, both ASIF algorithms exhibit similar behavior. Despite the implicit methods lacking explicit knowledge of the boundary, all cases exhibit similar operational regions. Note that the backup trajectory under the implicit Simplex filter violates the safety constraint. This is by design, as the trajectories are integrated forward after taking a theoretical probe step. A buffer could be added to the switching region to assure safety. The ASIF approaches are slightly more conservative due to the presence of the α term in the barrier constraints. This function may be modified to shape the response.

While the Simplex and ASIF methods are the most distinctive in terms of behavior, the implicit and explicit methods are the most distinctive in terms of requirements. For instance, the key challenge in using the implicit methods is in the design of the backup controller and identification of the backup set. Once these have been identified, it is relatively straightforward to implement either approach. Likewise, the key challenge in implementing the explicit algorithms is identifying an invariant set. Importantly, the explicit ASIF approach is the only algorithm that does not require any backup controller to be defined.

ASSURANCE IN THE PRESENCE OF UNCERTAINTY

While cyberphysical systems may be *modeled* using dynamical systems, as in (1) and (2), real-world systems rarely, if ever, obey the dynamics of their model precisely. Furthermore, deterministic models do not capture the effects of external disturbances, which are universal in real-world systems.

One way to address model error is to construct a higher-fidelity system model, i.e., a higher-order parameterized model that can be tuned in system identification. However, producing higher-fidelity modes can become an exhaustive procedure and be unfruitful when a newly suggested complex model performs less accurately in comparison to its simpler predecessor. These hindrances have attracted the attention of numerous fields of study, for example, robust control [138] and Sim2Real [139].

A second way to address model error and external disturbances is to construct a *nondeterministic* system model, that is, a dynamical model that accounts for environmental and internal disturbances by incorporating one or more noise terms. In continuous time, one such a model is represented by

$$\dot{x} = f(x) + g_1(x)u + g_2(x)w \tag{52}$$

where $x \in X$ and $u \in U$ maintain their definitions as the system state and control input and where the term $w(t) \in W \subset \mathbb{R}^p$ is now included to represent a bounded nondeterministic input signal to the system. Employing a nondeterministic system model, as in (52), reduces the design burden involved in forming an accurate representation of real-world phenomena. When designed correctly, a nondeterministic model *will*, in general, describe the real-world behavior accurately.

This section presents RTA mechanisms in the context of uncertain systems, as in (52). First, an *explicit* RTA formulation is presented in which *robust CBFs* are employed to assure that system trajectories evolve in a given safe set. Next, an implicit solution to this problem is presented, where runtime safety is assessed through the online computation of reachable sets.

Explicit RTA for Uncertain Systems

In this setting of (52), an ASIF method provides a control policy u_{act} that renders a given safe set $C_S = \{x \in X | h(x) \geq 0\}$ forward invariant, regardless of the disturbance input $w(t)$ chosen. For this reason, ASIF methods are naturally constructed from *robust CBFs*, as discussed in the following.

Robust CBFs extend the general barrier function theory presented in the preceding to the nondeterministic setting of (52). A continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ is a robust CBF for (52) if there exists a class κ function $\alpha : \mathbb{R} \to \mathbb{R}$ such that for all $x \in C_S := \{x \in X | h(x) \geq 0\}$, there exists a $u \in \mathbb{R}^m$ satisfying

$$BC(x, u, w) = L_f h(x) + L_{g_1} h(x)u + L_{g_2} h(x)w + \alpha(h(x)) \geq 0 \tag{53}$$

for all $w \in W$, where $L_f h(x)$, $L_{g_1} h(x)$ and $L_{g_2} h(x)$ are naturally taken to be the Lie derivatives of h along f, g_1, and g_2. The barrier constraint (53) is a linear inequality on the variable u for any $x \in \mathbb{R}^n$ and $w \in W$, and when a candidate controller $u_{des}(x)$ satisfies (53) for all $x \in C_S$ and $w \in W$, the

set C_S will be robustly forward invariant for the closed-loop dynamics of (52) under u_{des}.

When, instead, $u_{des}(x)$ does not satisfy (53), an ASIF method can be employed to assure the forward invariance of C_S. One such filter, given in the following as the RASIF-QP, is posed as a QP, where the constraints in this program are constructed from (53):

RASIF-QP

$$u_{act}(x) = \underset{u \in U}{\operatorname{argmin}} \| u - u_{des}(x) \|_2^2 \tag{54}$$

$$\text{s.t. } L_f h(x) + L_{g_1} h(x) u + L_{g_2} h(x) w + \alpha(h(x)) \geq 0 \quad \text{for all } w \in W. \tag{55}$$

When h is a robust CBF for (52), the RASIF-QP is always feasible, and the set C_S is robustly forward invariant for the closed-loop dynamics of (52) under u_{act}. However, this program contains an infinite number of linear constraints and thus is not always practically implementable. In the special instance where the disturbance set W is a polytope, the program RASIF-QP can be reduced to include only a finite number of linear constraints. Assume, for instance, that

$$W = \mathbf{Conv}\{w^1, \ldots, w^q\} \tag{56}$$

for some $w^1, \ldots, w^q \in \mathbb{R}^p$, where **Conv** denotes the convex hull operator. Then, an explicit ASIF is constructed using the RASIF-QP (polytope W):

RASIF-QP (polytope W)

$$u_{act}(x) = \underset{u \in \mathbb{R}^m}{\operatorname{argmin}} \| u - u_{des}(x) \|_2^2 \tag{57}$$

$$\text{s.t. } L_f h(x) + L_{g_1} h(x) u + L_{g_2} h(x) w + \alpha(h(x)) \geq 0 \quad \text{for all } w \in \{w^1, \ldots, w^q\} \tag{58}$$

and this program contains only q linear constraints.

Implicit Active Set Invariance for Uncertain Systems

In this section, an implicit RTA formulation is introduced for the nondeterministic setting of (52). As in the case of deterministic systems, implicit ASIF allows the system to leave the safe set C_S when safety is verified a priori via the assessment of a known safe backup control policy.

In the deterministic setting of (27), the safety of a given state $x \in X$ with respect to the backup controller u_b is assessed via a system simulation; however, in the nondeterministic setting of (52), it is necessary to assess the effects of the disturbance, perhaps through, for example, computing reachable sets [67], [140], [141]. For initial set $A \subseteq X$ and time $t \geq 0$, the time t reachable set of (52) under u_b is

$$R(t; A) := \{\Phi^{u_b}(t; x, \mathbf{w}) \in X | x \in A, \mathbf{w} : [0, t) \to \mathcal{W}\} \tag{59}$$

where $\Phi^{u_b}(t; x, \mathbf{w})$ denotes the state of (52) reached at time t when beginning at state $x \in X$ at time zero and evolving subject to the backup control input u_b and disturbance signal $\mathbf{w} : [0, t] \to \mathcal{W}$. In particular, an implicit ASIF ensures $R(T_b; x) \subseteq C_b$ for all states x along the system trajectory, where $T_b \geq 0$ maintains its definition and function as the horizon time of the backup controller.

In general, it can be difficult to compute reachable sets in closed form. However numerous efficient computational methods exist for overapproximating reachable sets, with computational speeds suitable for, for example, computing reachable sets in the control loop and enforcing safe behavior online from these predictions [140], [141], [142], [143]. Of particular interest to this work, the mixed monotonicity property of dynamical systems can be applied to compute a hyperrectangular overapproximation of $R(t;x)$ by using a single simulation of a related $2n$-dimensional deterministic embedding system, a technique that has been applied in domains such as transportation systems [144], and biological systems [145]. Further details on the mixed monotonicity property are provided in "Mixed Monotonicity for Efficient Reachability." See also [S24] and [182] for tutorials.

In the remainder of this section, we present one possible implicit ASIF construction, where overapproximations of reachable sets are computed online using the mixed monotonicity property. The basic idea is as follows: when the system is at state x, ASIF computes a hyperrectangular overapproximation of $R(T_b; x)$ by using the mixed monotonicity procedure. A barrier condition is then enforced for each vertex of the approximation to ensure $R(T_b; x) \subseteq C_b$ for all states x along the system trajectory. In this way, safety is ensured. Specific details are as follows.

For a finite set $\mathcal{S} \subset \mathbb{R}$ and some fixed parameter $p > 0$, the *log-sum-exponential* (*LSE*) of $\mathcal{S}$ is given by

$$\text{LSE}(\mathcal{S}) = -\frac{1}{p} \log \sum_{s \in \mathcal{S}} \exp(-p \cdot s). \tag{60}$$

The LSE has several useful properties: namely, $\text{LSE}(\mathcal{S}, p)$ is differentiable with respect to the elements of $\mathcal{S}$, and $\text{LSE}(\mathcal{S}, p)$ approximates $\min \mathcal{S}$; i.e.,

$$\min \mathcal{S} - \frac{n}{p} \log 2 \leq \text{LSE}(\mathcal{S}, p) < \min \mathcal{S} \tag{61}$$

for all $p > 0$, and this approximation can be made arbitrarily tight by choosing p large enough. We use the LSE, in this section, to approximate the minimum evaluation of h over a hyperrectangular subset of the state space. Define the following:

$$\text{LSE}_h(a) := \text{LSE}(\{h(z) \,|\, z \in \langle\langle\, a \,\rangle\rangle\}, p) \tag{62}$$

$$\text{LSE}_h(\Phi^E(t; x)) \tag{63}$$

$$\Psi(x) := \sup_{0 \leq \tau \leq T_b} \gamma(\tau; x) \tag{64}$$

where $\langle\langle\, x, y \,\rangle\rangle$ in (62) denotes the set of 2^n corners of a rectangle $[x,y]$ and is given by

$$\langle\langle\, x,y \,\rangle\rangle := \{z \in X | z_i \in \{x_i, y_i\}\} \tag{65}$$

and where $[\Phi^E(t;x)] \supset R(t;x)$ is the rectangular reachable set overapproximation, as defined later in (S2).

From the preceding definitions, we have that $\Psi(x) \geq 0$ when there exists a $t \in [0, T_\mathrm{b}]$ so that $[\Phi^E(t;x)] \subseteq C_\mathrm{b}$, and in this case, $R(T_\mathrm{b}; x) \subseteq C_\mathrm{b}$, as C_b is assumed to be robustly forward invariant for (52) under the backup controller u_b. Thus, an implicit ASIF for (52) is constructed using the RASIF-QP (57)—(58) with barrier function $h(x) := \Psi(x)$.

VERIFICATION OF RTA ALGORITHMS AND ARCHITECTURES

A key question in trusting RTA systems is, "Who checks the checker?" In many applications, RTA systems serve a safety-critical role and therefore must be rigorously verified. One can imagine a second RTA to check the RTA and another to

Mixed Monotonicity for Efficient Reachability

A dynamical system, possibly subject to a nondeterministic disturbance input, is mixed monotone when there exists a related decomposition function that separates the system dynamics into cooperative and competitive state interactions. Mixed monotonicity applies to continuous-time systems [S24], [S25], discrete-time systems [145], and systems with disturbances [S26], [S27], and it generalizes the *monotonicity* property of dynamical systems for which trajectories maintain a partial order over states [S28], [S29].

For an n-dimensional mixed monotone system with a disturbance input, it is possible to construct a $2n$-dimensional monotone embedding system from the decomposition function. This enables one to apply the powerful theory of monotone dynamical systems to the embedding system, and it can be used to compute useful properties of the initial system. In particular, such approaches are useful for efficiently computing reachable sets and robustly forward invariant sets for systems with disturbances, techniques that have been successfully demonstrated in online safety applications [140], [141], [142], [143]. See also [S24] and [182] for tutorials.

A dynamical system

$$\dot{x} = F(x, w) \tag{S1}$$

with state $x \in X \subseteq \mathbb{R}^n$ and disturbance $w \in \mathcal{W} \subseteq \mathbb{R}^m$, is a *mixed monotone system* when there exists a related *decomposition function* $d : X \times \mathcal{W} \times X \times \mathcal{W} \to \mathbb{R}^n$ so that for all $x, \hat{x} \in X$ and $w, \hat{w} \in \mathcal{W}$, the following hold:

- $d(x, w, x, w) = F(x, w)$.
- $\partial d_i / \partial x_j (x, w, \hat{x}, \hat{w}) \geq 0$ for all i, j, with $i \neq j$.
- $\partial d_i / \partial \hat{x}_j (x, w, \hat{x}, \hat{w}) \leq 0$ for all i, j.
- $\partial d_i / \partial w_k (x, w, \hat{x}, \hat{w}) \geq 0$ and $\partial d_i / \partial \hat{w}_k (x, w, \hat{x}, \hat{w}) \leq 0$ for all i, k.

Large classes of systems have been shown to be mixed monotone, with decomposition functions constructed from, for example, bounds on the systems' Jacobian matrix [S26] or domain-specific knowledge [S30], [S31]. In certain instances, decomposition functions can also be computed by solving an optimization problem [S32].

An important feature of mixed monotone systems is that hyperrectangular overapproximations of reachable sets can be efficiently computed using a single simulation of a related $2n$-dimensional embedding system constructed from the decomposition function. Assume that X is an extended hyperrectangle and that $\mathcal{W} := [\underline{w}, \overline{w}]$ is a hyperrectangle. When (S1) is mixed monotone with decomposition function d,

$$\begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = E(x, \hat{x}) := \begin{bmatrix} d(x, \underline{w}, \hat{x}, \overline{w}) \\ d(\hat{x}, \overline{w}, x, \underline{w}) \end{bmatrix} \tag{S2}$$

is the *embedding system* relative to d, and $\Phi^E(t; a)$ denotes the state of (S2) reached at time $t \geq 0$ when beginning at state $a \in X \times X$ at time zero. For any hyperrectangular set of states $[\underline{x}, \overline{x}] \subset X$, the following is true: if $\Phi^E(\tau; (\underline{x}, \overline{x})) \in X \times X$ for all $0 \leq \tau \leq t$,

$$R(t; [\underline{x}, \overline{x}]) \subseteq [\Phi^E_{1:n}(t; (\underline{x}, \overline{x})), \Phi^E_{n+1:2n}(t; (\underline{x}, \overline{x}))]. \tag{S3}$$

The results of (S3) provide an efficient procedure for overapproximating reachable sets for systems with disturbances; in particular, a hyperrectangular overapproximation of $R(t; [\underline{x}, \overline{x}])$ is computable from a single simulation of the embedding system (S2), where the bottom and top corners of the approximation are the first n and last n coordinates of $\Phi^E(t; (\underline{x}, \overline{x}))$. This result is demonstrated in the following numerical example.

Example

Consider the system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = F(x) = \begin{bmatrix} x_2^2 + 2 \\ x_1 \end{bmatrix} \tag{S4}$$

with state space $X = \mathbb{R}^2$. The system (S4) is mixed monotone with respect to the *decomposition function* $d : X \times X \to \mathbb{R}^2$ given by

$$\begin{aligned} d_1(x, \hat{x}) &= \begin{cases} x_2^2 + 2 & \text{if } x_2 \geq \max\{0, -\hat{x}_2\} \\ \hat{x}_2^2 + 2 & \text{if } \hat{x}_2 \leq \min\{0, -x_2\} \\ 2 & \text{if } x_2 \leq 0 \leq \hat{x}_2 \end{cases} \\ d_2(x, \hat{x}) &= x_1. \end{aligned} \tag{S5}$$

Reachable sets for (S4) are now approximated from a single simulation of the embedding system (S2), as described in

check that RTA until it is turtles all the way down. Proponents of the Simplex architecture advocate simple switching logic and preplanned backup control actions that reduce the verification burden [108]. Other approaches may rely on mathematical proofs for verification [146]. Nearly all approaches see the RTA as a modular component within the closed-loop control system that can be a verified subcomponent. It is worth noting that this verification can be done on the algorithms and actual software implementation. This section focuses on the verification of RTA algorithms and architectures in the early design phases, using formal and compositional reasoning.

Formal Specification and Analysis of RTA System Requirements, Architecture, and Design

Formal methods tools from computer science for the automated verification of hardware and software [147] can be brought to the problem of verifying RTA. In addition to the ability to verify software implementations, formal methods can be used to aid in the modeling and automated analysis of algorithm and

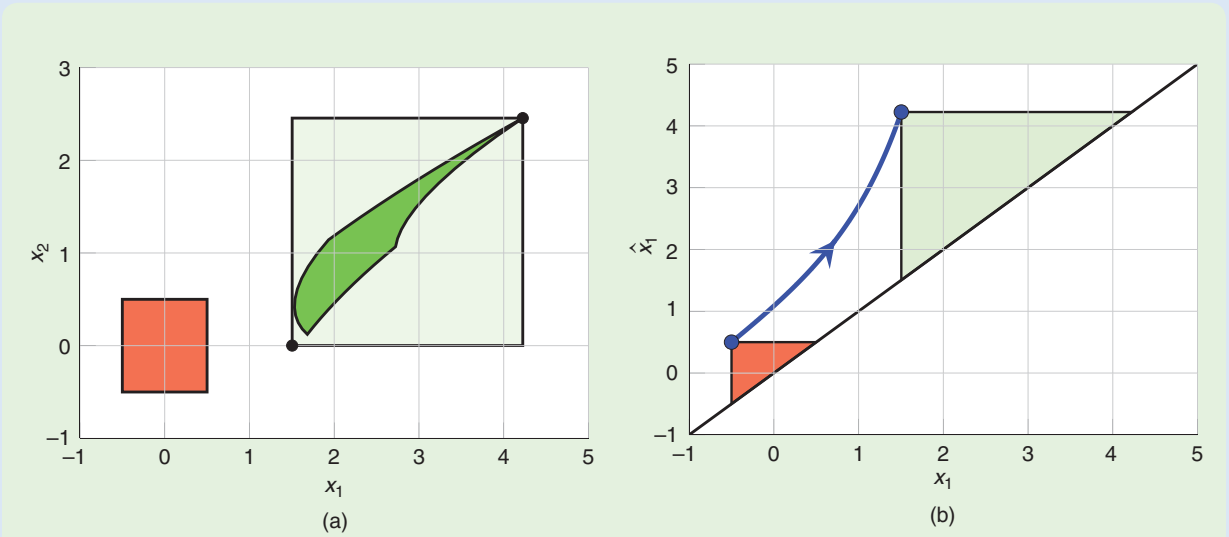

FIGURE S2 Approximating reachable sets of (S4) from the set of initial conditions $X_0 = [-1/2, 1/2] \times [-1/2, 1/2]$. (a) Here, X_0 is shown in red, and $R(1; X_0)$ is shown in green. The hyperrectangular overapproximation of $R(1; X_0)$, which is computed from the embedding system (S2), as described in (S3), is shown in light green. (b) A visualization of the bounding procedure from (S3). The trajectory of (S2) that yields Figure S2(a) is shown in blue, where Φ^E is projected to the $x_1, \hat{x}_1$-plane. The southeast cones corresponding to X_0 and the hyperrectangular overapproximation of $R(1; X_0)$ are shown in red and green, respectively.

(S3). An example is provided in Figure S2, where $R(1; X_0)$ is approximated for $X_0 = [-1/2, 1/2]^2$.

REFERENCES

[S24] S. Coogan, "Mixed monotonicity for reachability and safety in dynamical systems," in *Proc. 59th IEEE Conf. Decis. Contr. (CDC)*, 2020, pp. 5074–5085, doi: 10.1109/CDC42340.2020.9304391.

[S25] G. Enciso, H. Smith, and E. Sontag, "Nonmonotone systems decomposable into monotone systems with negative feedback," *J. Differ. Equ.*, vol. 224, no. 1, pp. 205–227, May 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002203960500210X, doi: 10.1016/j.jde.2005.05.007.

[S26] P.-J. Meyer, A. Devonport, and M. Arcak, "TIRA: Toolbox for interval reachability analysis," in *Proc. 22nd ACM Int. Conf. Hybrid Syst., Comput. Contr.*, 2019, pp. 224–229, doi: 10.1145/3302504.3311808.

[S27] S. Coogan and M. Arcak, "Efficient finite abstraction of mixed monotone systems," in *Proc. 18th Int. Conf. Hybrid Syst., Comput. Contr.*, 2015, pp. 58–67, doi: 10.1145/2728606.2728607.

[S28] H. Smith, *Monotone Dynamical Systems: An Introduction to the Theory of Competitive and Cooperative Systems: An Introduction to the Theory of Competitive and Cooperative Systems*. Providence, RI, USA: American Mathematical Society, 2008. [Online]. Available: https://books.google.com/books?id=vOfNAwAAQBAJ

[S29] D. Angeli and E. D. Sontag, "Monotone control systems," *IEEE Trans. Autom. Control*, vol. 48, no. 10, pp. 1684–1698, Oct. 2003, doi: 10.1109/TAC.2003.817920.

[S30] M. Abate and S. Coogan, "Computing robustly forward invariant sets for mixed-monotone systems," in *Proc. 59th IEEE Conf. Decis. Contr. (CDC)*, 2020, pp. 4553–4559, doi: 10.1109/CDC42340.2020.9304461.

[S31] S. Coogan and M. Arcak, "Stability of traffic flow networks with a polytree topology," *Automatica*, vol. 66, pp. 246–253, Apr. 2016, doi: 10.1016/j.automatica.2015.12.015.

[S32] M. Abate, M. Dutreix, and S. Coogan, "Tight decomposition functions for continuous-time mixed-monotone systems with disturbances," *IEEE Contr. Syst. Lett.*, vol. 5, no. 1, pp. 139–144, Jan. 2021, doi: 10.1109/LCSYS.2020.3001085.

decision logic with mathematical rigor. At one level, Lyapunov proofs can be used to guarantee the safety of control systems. Formal methods can supplement these proofs and perhaps be used where Lyapunov proofs do not exist. Three goals of formal methods are specification, analysis, and synthesis.

Formal specification facilitates a common unambiguous understanding of the system among users, designers, programmers, and testers. "Specification is difficult, unglamorous, and arguably the biggest bottleneck facing verification and validation of aerospace, and other, autonomous systems" [148]. RTA systems and formal specification and analysis have been applied successfully in a variety of applications [60], [63], [66], [97], [106], [107], [149], [150], [151], [152], [153], [154], [155], [156], [157], [158], [159], [160], [161]. Formal RTA requirements specifications have also been developed and analyzed [162], [163], [164].

Formal analysis helps to find errors and increase the reliability of the system. Formal analysis can be divided [165] into axiomatic (for example, Hoare logic [166] and theorem proving [167]), semantic approaches (for example, model checking [147]), and static analysis methods (for example, abstract interpretation [168]). Many formal analysis approaches focus on satisfiability modulo theories, where sets of variables are substituted with binary-valued functions of nonbinary values called predicates [169]. An example predicate is an inequality that is evaluated as true or false, such as $x + y \leq c$, with variables x and y and constant c. These predicates can be combined in a variety of logics, such as first-order and temporal logics, including linear temporal logic (LTL), computational tree logic (CTL), timed CTL, metric temporal logic, and signal temporal logic [147], [170], [171], [172]. The formal verification of LTL specifications has been applied to the proportional-integral-derivative attitude control of spacecraft [173], a Simplex RTA system for spacecraft attitude [160], [174], [175], switching logic for automatic maneuvering [161], LTL specification monitor automaton [67], and many other applications.

Formal synthesis integrates formal methods into the development process to convert a design to implementation with some level of automation. While many controller designs are synthesized in some fashion, formal synthesis typically involves the automatic generation of a controller and an RTA filter from formal specifications [176].

ALGORITHM 5 Implicit ASIF for nondeterministic control systems.

Input : Current state $x_{\text{curr}} \in X$
: Desired input $u_{\text{des}} \in U$
Output : Safe control input $u_{\text{act}} \in U$
Predefined: Constraint set $C_{\text{A}} \subset X$
: Invariant set $C_{\text{S}} \subset C_{\text{A}}$
: Backup control law $u_{\text{b}} : X \to U$.

function $u_{\text{act}}(x) = \text{ASIF}(x_{\text{curr}}, u_{\text{des}})$
  compute:
  $u^* = \text{argmin}_{u \in \mathbb{R}^m} \| u - u_{\text{des}} \|_2^2$
  subject to $\frac{\partial \Psi}{\partial x}(x_{\text{curr}})(f(x_{\text{curr}}) + g_1(x_{\text{curr}})u + g_2(x_{\text{curr}})w) \geq -\alpha(\Psi(x_{\text{curr}}))$
    $\forall w \in \langle\langle \underline{w}, \overline{w} \rangle\rangle$
  if Program feasible **then return** u^*
  else return $u_{\text{b}}(x_{\text{curr}})$

Compositional Verification

Compositional verification, sometimes referred to as a "divide-and-conquer" approach to verification [177], helps to verify and test portions of algorithms and systems architectures separately and then make conclusions about the system as a whole. This verification approach is helpful in modern engineering practices, where engineering tasks are divided over large teams [178]. Compositional verification includes statements such as, If property p_1 for subsystem A and property p_2 for subsystem B both hold, it implies that property p holds for the entire system. A compositional verification approach can be used to apply formal methods to verify components of an RTA system, especially in Simplex-based RTA systems, where multiple simple components (monitors, backup controllers, and switching functions) work together to ensure safety [66], [90], [160], [174], [175], [179].

Compositional verification often relies on the *assume–guarantee reasoning* of the architecture [180] and hybrid switching [181]. Assume–guarantee contracts make guarantees on the output of a component, based on assumptions about the input. Assumptions are expectations that a component has about its environment, while guarantees are behavior properties provided by the component. Verification techniques are applied to individual components with explicit formal assumptions and guarantees, as depicted in Figure 17.

CONCLUSIONS

RTA systems are growing in popularity as a way to assure safety by altering unsafe control inputs from a primary controller. The assurance mechanism of RTA systems is constructed to be agnostic to the underlying structure of the primary controller, whether the primary control comes from a human operator, an advanced control approach, or an autonomous control approach. By effectively decoupling the enforcement of safety constraints from performance-related objectives, RTA offers a number of useful advantages over traditional (offline) verification.

This article provided a tutorial on developing RTA systems, with particular emphasis on implicit and explicit safety definitions as well as Simplex architecture and ASIF RTA approaches. Implicit (or trajectory-based) safety definitions compute finite time trajectories under a backup controller online, while explicit (or region-based) approaches identify trusted regions offline via the construction of either

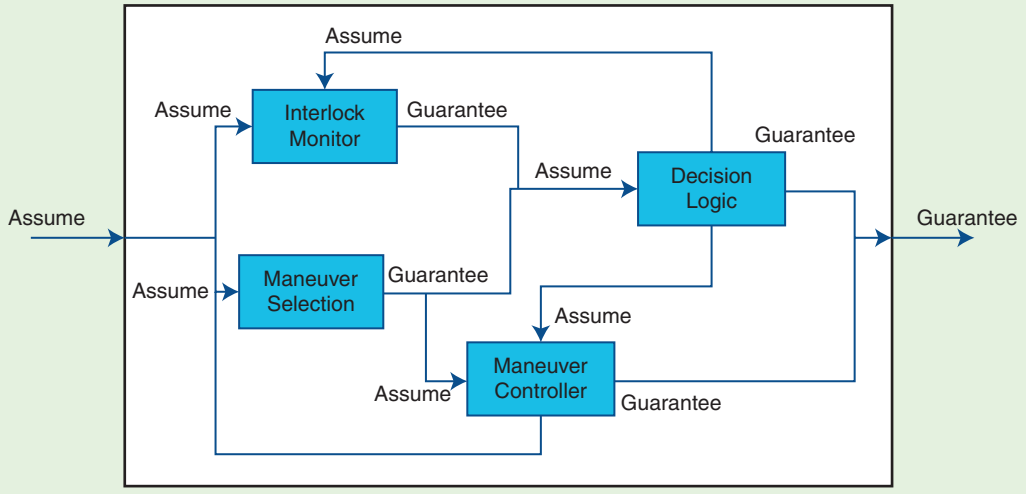

FIGURE 17 A depiction of assume–guarantee reasoning for an automatic collision avoidance system, where assumptions about properties are guaranteed by a component based on assumptions about the environment. Guarantees on the output of one component translate to assumptions on the input of another component until larger properties of the overall system are proved.

a large forward invariant or control invariant set. Simplex architecture RTA designs *monitor* for imminent violations of safety constraints and switch from the primary controller to a backup controller when an imminent unsafe condition is detected. By contrast, backup control intervenes gradually in ASIF approaches, rather than with a hard switch. Given the increasing complexity of modern control systems, RTA, in its various forms, is one tool to assure safety and a potential path to certification in safety-critical domains.

AUTHOR INFORMATION

Kerianne L. Hobbs (kerianne.hobbs@us.af.mil) is the safe autonomy lead of the Autonomy Capability Team, Air Force Research Laboratory, Wright-Patterson Air Force Base, OH 45433 USA. There, she investigates rigorous specification, analysis, and bounding techniques to enable the certification of autonomous and learning controllers for aircraft and spacecraft applications. Her previous experience includes work in automatic collision avoidance at the Air Force Research Laboratory, from 2011 to 2014, and autonomy verification and validation research, from 2012 to 2020. She has a B.S. degree in aerospace engineering from Embry–Riddle Aeronautical University, an M.S. degree in astronautical engineering from the Air Force Institute of Technology, and a Ph.D. degree in aerospace engineering from the Georgia Institute of Technology.

Mark L. Mote is a cofounder of Pytheia, Atlanta, GA, 30308 USA. He received his Ph.D. degree in robotics from the Georgia Institute of Technology in 2021 and his B.S. and M.S. degrees in aerospace engineering from the Georgia Institute of Technology in 2015 and 2018. His research interests include safe autonomy, perception, optimization, trajectory planning, and runtime assurance for safety-critical systems.

Matthew C.L. Abate is a cofounder of Pytheia, Atlanta, GA 30308 USA. He received the Ph.D. degree in robotics, the M.S. degree in mechanical engineering, and the M.S. degree in electrical and computer engineering from Georgia Institute of Technology, in 2017, 2018, and 2020, respectively. He also received the B.A. degree in engineering sciences and the B.E degree in mechanical engineering from Dartmouth College in 2017. His research interests include safety and verification for uncertain dynamical systems. He is a Student Member of IEEE.

Samuel D. Coogan is assistant professor in the School of Electrical and Computer Engineering and the School of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA. Prior to joining the Georgia Institute of Technology in 2017, he was an assistant professor in the Department of Electrical Engineering, University of California, Los Angeles. He received his B.S. degree in electrical engineering from the Georgia Institute of Technology and M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley. His research interests include dynamical systems and autonomy and focus on developing scalable tools for the verification and control of networked cyberphysical systems, with an emphasis on transportation systems. He received the 2019 Donald P. Eckman Award from the American Automatic Control Council, a 2019 Young Investigator Award from the Air Force Office of Scientific Research, a 2018 CAREER award from the National Science Foundation, and the 2017 IEEE Transactions on Control of Network Systems Outstanding Paper Award.

Eric M. Feron is professor of electrical, computer, and mechanical Engineering at King Abdullah University of Science and Technology (KAUST), Thuwal 23955, Saudi Arabia. His research interests include aerospace systems ranging from aircraft to space systems, with an emphasis on robotic

autonomy and artificial intelligence. Prior to his KAUST appointment, he was a professor of aeronautics and astronautics at the Massachusetts Institute of Technology and a professor of aerospace engineering at the Georgia Institute of Technology. He also works, on occasion, with the Ecole Nationale de l'Aviation Civile and the Institut Supérieur de l'Aéronautique et de l'Espace, France, where he was born.

REFERENCES

[1] "Auto-GCAS saves F-16." Wikimedia Commons. Accessed: Jan. 18, 2023. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/7/75/Auto-GCAS_saves_F-16.webm

[2] Advisory Circulars, "AC 25.1309-1A - System design and analysis," Federal Aviation Administration, Washington, DC, USA, Jun. 1988. [Online]. Available: https://www.faa.gov/regulations_policies/advisory_circulars/index.cfm/go/document.information/documentID/22680

[3] "Verification," Defense Acquisition Univ., Fort Belvoir, VA, USA, 2018. Accessed: Jan. 18, 2023. [Online]. Available: https://www.dau.edu/glossary/Pages/Glossary.aspx#!both|V|28759

[4] "Validation," Defense Acquisition Univ., Fort Belvoir, VA, USA, 2018. Accessed: Jan. 18, 2023. [Online]. Available: https://www.dau.edu/glossary/Pages/Glossary.aspx#!both|V|28748

[5] J. W. R. Nichols and T. Scanlon, "DoD developer's guidebook for software assurance," Carnegie Mellon Univ., Softw. Eng. Inst., Pittsburgh, PA, USA, Dec. 2018. Accessed: Jan. 18, 2023. [Online]. Available: https://resources.sei.cmu.edu/asset_files/SpecialReport/2018_003_001_538761.pdf

[6] "Standard airworthiness certification regulations: Title 14, code of federal regulations," Federal Aviation Admin., U.S. Dept. Transp., Washington, DC, USA, Oct. 2017. [Online]. Available: https://www.faa.gov/aircraft/air_cert/airworthiness_certification/std_awcert/std_awcert_regs/regs/

[7] "Airworthiness certification criteria," U.S. Dept. Defense, Arlington, VA, USA, MIL-HDBK-516C, Dec. 2014.

[8] "Standard practice: System safety," U.S. Dept. Defense, Arlington, VA, USA, MIL-STD-882E, May 2012.

[9] *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*, ARP4761, SAE International, Warrendale, PA, USA, 1996.

[10] *Guidelines for Development of Civil Aircraft and Systems*, ARP4754A, SAE International, Warrendale, PA, USA, 2010.

[11] *Software Considerations in Airborne Systems and Equipment Certification*, RTCA DO-178, Radio Technical Commission for Aeronautics, Washington, DC, USA, 2011.

[12] *Design Assurance Guidance for Airborne Electronic Hardware*, RTCA DO-254, Radio Technical Commission for Aeronautics, Washington, DC, USA, 2000.

[13] *Formal Methods Supplement to DO-178C and DO-278A*, RTCA DO-333, Radio Technical Commission for Aeronautics, Washington, DC, USA, 2011.

[14] C. Hu, Z. Wang, Y. Qin, Y. Huang, J. Wang, and R. Wang, "Lane keeping control of autonomous vehicles with prescribed performance considering the rollover prevention and input saturation," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 7, pp. 3091–3103, Jul. 2020, doi: 10.1109/TITS.2019.2924937.

[15] X. Xu, J. W. Grizzle, P. Tabuada, and A. D. Ames, "Correctness guarantees for the composition of lane keeping and adaptive cruise control," *IEEE Trans. Autom. Sci. Eng. (From July 2004)*, vol. 15, no. 3, pp. 1216–1229, Jul. 2018, doi: 10.1109/TASE.2017.2760863.

[16] T. Gurriet et al., "Towards restoring locomotion for paraplegics: Realizing dynamically stable walking on exoskeletons," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2018, pp. 2804–2811, doi: 10.1109/ICRA.2018.8460647.

[17] O. Harib et al., "Feedback control of an exoskeleton for paraplegics: Toward robustly stable, hands-free dynamic walking," *IEEE Control Syst. Mag.*, vol. 38, no. 6, pp. 61–87, Dec. 2018, doi: 10.1109/MCS.2018.2866604.

[18] O. Sanni et al., "Ariadne: A common-sense thread for enabling provable safety in air mobility systems with unreliable components," KAUST Univ. Library, Thuwal, Saudi Arabia, 2021. [Online]. Available: http://hdl.handle.net/10754/666807

[19] E. Squires, P. Pierpaoli, and M. Egerstedt, "Constructive barrier certificates with applications to fixed-wing aircraft collision avoidance," in *Proc. IEEE Conf. Contr. Technol. Appl. (CCTA)*, 2018, pp. 1656–1661, doi: 10.1109/CCTA.2018.8511342.

[20] E. Squires, P. Pierpaoli, R. Konda, S. Coogan, and M. Egerstedt, "Composition of safety constraints with applications to decentralized fixed-wing collision avoidance," 2019, *arXiv:1906.03771*.

[21] E. Squires, R. Konda, P. Pierpaoli, S. Coogan, and M. Egerstedt, "Safety with limited range sensing constraints for fixed wing aircraft," 2020, *arXiv:2010.10883*.

[22] T. Schouwenaars, M. Valenti, E. Feron, and J. How, "Implementation and flight test results of MILP-based UAV guidance," in *Proc. IEEE Aerosp. Conf.*, 2005, pp. 1–13, doi: 10.1109/AERO.2005.1559600.

[23] T. Schouwenaars, "Safe trajectory planning of autonomous vehicles," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2006.

[24] C. Hanson, "Capability description for NASA'S F/A-18 TN 853 as a testbed for the integrated resilient aircraft control project," Nat. Aeronaut. Space Admin., Washington, DC, USA, 2009. [Online]. Available: https://ntrs.nasa.gov/citations/20090022324

[25] L. Wang, A. D. Ames, and M. Egerstedt, "Safe certificate-based maneuvers for teams of quadrotors using differential flatness," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2017, pp. 3293–3298, doi: 10.1109/ICRA.2017.7989375.

[26] A. Singletary, T. Gurriet, P. Nilsson, and A. D. Ames, "Safety-critical rapid aerial exploration of unknown environments," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2020, pp. 10,270–10,276, doi: 10.1109/ICRA40945.2020.9197416.

[27] M. Mote, C. W. Hays, A. R. Collins, E. Feron, and K. L. Hobbs, "Natural motion-based trajectories for automatic spacecraft collision avoidance during proximity operations," in *Proc. IEEE Aerosp. Conf.*, 2021, pp. 1–12, doi: 10.1109/AERO50100.2021.9438434.

[28] A. Singletary, P. Nilsson, T. Gurriet, and A. D. Ames, "Online active safety for robotic manipulators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2019, pp. 173–178, doi: 10.1109/IROS40897.2019.8968231.

[29] E. Wong, J. D. Schierman, T. Schlapkohl, and A. Chicatelli, "Towards run-time assurance of advanced propulsion algorithms," in *Proc. 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conf.*, 2014, p. 3636, doi: 10.2514/6.2014-3636.

[30] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Proc. Int. Conf. Comput. Aided Verification*, Springer-Verlag, 2017, pp. 97–117.

[31] H.-D. Tran et al., "NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," in *Proc. Int. Conf. Comput. Aided Verification*, Springer-Verlag, 2020, pp. 3–17.

[32] S. Bak, H.-D. Tran, K. Hobbs, and T. T. Johnson, "Improved geometric path enumeration for verifying RELU neural networks," in *Proc. Int. Conf. Comput. Aided Verification*, Springer-Verlag, 2020, pp. 66–96.

[33] S. Gokulanathan, A. Feldsher, A. Malca, C. Barrett, and G. Katz, "Simplifying neural networks using formal verification," in *Proc. NASA Formal Methods Symp.*, Springer-Verlag, 2020, pp. 85–93.

[34] J. Garcia and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, Jan. 2015.

[35] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 2669–2678, doi: 10.1609/aaai.v32i1.11797.

[36] R. Alur, *Principles of Cyber-Physical Systems*. Cambridge, MA, USA: MIT Press, 2015.

[37] J. C. Knight, "Safety critical systems: Challenges and directions," in *Proc. 24th Int. Conf. Softw. Eng.*, 2002, pp. 547–550.

[38] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton, NJ, USA: Princeton Univ. Press, 2012.

[39] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. New York, NY, USA: Springer Science & Business Media, 2009.

[40] N. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA, USA: MIT Press, 2011.

[41] J. D. Schierman et al., "Runtime assurance framework development for highly adaptive flight control systems," Barron Associates, Inc., Charlottesville, VA, USA, Tech. Rep. AFRL-RQ-WP-TR-2016-0001, 2015. [Online]. Available: https://apps.dtic.mil/sti/pdfs/AD1010277.pdf

[42] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre, *Viability Theory: New Directions*. New York, NY, USA: Springer Science & Business Media, 2011.

[43] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Proc. Int. Workshop Hybrid Syst., Comput. Contr.*, Springer-Verlag, 2004, pp. 477–492, doi: 10.1007/978-3-540-24743-2_32.

[44] J. Anderson and A. Papachristodoulou, "Advances in computational Lyapunov analysis using sum-of-squares programming," *Discrete Continuous Dyn. Syst.-B*, vol. 20, no. 8, 2015, Art. no. 2361, doi: 10.3934/dcdsb.2015.20.2361.

[45] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE*

Trans. Autom. Control, vol. 52, no. 8, pp. 1415–1428, Aug. 2007, doi: 10.1109/TAC.2007.902736.
[46] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Trans. Autom. Control*, vol. 50, no. 7, pp. 947–957, Jul. 2005, doi: 10.1109/TAC.2005.851439.
[47] J. H. Gillula, S. Kaynama, and C. J. Tomlin, "Sampling-based approximation of the viability kernel for high-dimensional linear sampled-data systems," in *Proc. 17th Int. Conf. Hybrid Syst., Comput. Contr.*, 2014, pp. 173–182, doi: 10.1145/2562059.2562117.
[48] D. P. Bertsekas and I. B. Rhodes, "On the minimax reachability of target sets and target tubes," *Automatica*, vol. 7, no. 2, pp. 233–247, Mar. 1971, doi: 10.1016/0005-1098(71)90066-5.
[49] T. Gurriet, "Applied safety critical control," Ph.D. dissertation, California Inst. Technol., Pasadena, CA, USA, 2020.
[50] I. Mitchell, "A summary of recent progress on efficient parametric approximations of viability and discriminating kernels," in *Proc. 1st Int. Workshop Symbolic Numer. Methods Reachability Analysis SNR@CAV*, 2015, pp. 23–31.
[51] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. New York, NY, USA: Cambridge Univ. Press, 2017.
[52] M. Nagumo, "Über die lage der integralkurven gewöhnlicher differentialgleichungen," *Proc. Physico-Math. Soc. Jpn.*, vol. 24, pp. 551–559, Jan. 1942.
[53] T. Gurriet, M. Mote, A. Singletary, P. Nilsson, E. Feron, and A. D. Ames, "A scalable safety critical control framework for nonlinear systems," *IEEE Access*, vol. 8, pp. 187,249–187,275, Sep. 2020, doi: 10.1109/ACCESS.2020.3025248.
[54] T. Gurriet, M. Mote, A. Singletary, E. Feron, and A. D. Ames, "A scalable controlled set invariance framework with practical safety guarantees," in *Proc. IEEE 58th Conf. Decis. Contr. (CDC)*, 2019, pp. 2046–2053.
[55] D. Mayne, "An apologia for stabilising terminal conditions in model predictive control," *Int. J. Contr.*, vol. 86, no. 11, pp. 2090–2095, 2013, doi: 10.1080/00207179.2013.813647.
[56] B. T. Lopez and J. P. How, "Aggressive 3-D collision avoidance for high-speed navigation," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2017, pp. 5759–5765.
[57] T. Schouwenaars, J. How, and E. Feron, "Receding horizon path planning with implicit safety guarantees," in *Proc. Amer. Contr. Conf.*, 2004, vol. 6, pp. 5576–5581.
[58] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Natural motion-based trajectories for automatic spacecraft proximity operation collision avoidance," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68–73, 2015, doi: 10.1016/j.ifacol.2015.11.154.
[59] W. Clohessy and R. Wiltshire, "Terminal guidance system for satellite rendezvous," *J. Aerosp. Sci.*, vol. 27, no. 9, pp. 653–658, Sep. 1960, doi: 10.2514/8.8704.
[60] S. Bak, K. Manamcheri, S. Mitra, and M. Caccamo, "Sandboxing controllers for cyber-physical systems," in *Proc. Int. Conf. Cyber-Phys. Syst.*, 2011, pp. 3–12, doi: 10.1109/ICCPS.2011.25.
[61] D. E. Swihart et al., "Automatic ground collision avoidance system design, integration, & flight test," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 26, no. 5, pp. 4–11, May 2011, doi: 10.1109/MAES.2011.5871385.
[62] E. M. Griffin et al., "Automatic ground collision avoidance system design for pre-block 40 F-16 configurations," in *Proc. Asia-Pacific Int. Symp. Aerosp. Technol.*, 2012, pp. 1–7.
[63] M. Aiello, J. Berryman, J. Grohs, and J. Schierman, "Run-time assurance for advanced flight-critical control systems," in *Proc. AIAA Guid., Navig., Contr. Conf.*, 2010, p. 8041, doi: 10.2514/6.2010-8041.
[64] A. Burns, K. Hobbs, J. Bier, and S. Whitcomb, "Advanced capabilities for the analog flight control F-16," in *Proc. NATO Sens. Electron. Technol. Panel 168 Symp.*, 2012.
[65] N. Minoiu Enache, S. Mammar, M. Netto, and B. Lusetti, "Driver steering assistance for lane-departure avoidance based on hybrid automata and composite Lyapunov function," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 1, pp. 28–39, Mar. 2010, doi: 10.1109/TITS.2009.2026451.
[66] K. L. Hobbs, "Elicitation and formal specification of run time assurance requirements for aerospace collision avoidance systems," Ph.D. dissertation, Georgia Inst. Technol., Atlanta, GA, USA, 2020.
[67] M. Abate, E. Feron, and S. Coogan, "Monitor-based runtime assurance for temporal logic specifications," in *Proc. IEEE 58th Conf. Decis. Contr. (CDC)*, 2019, pp. 1997–2002.
[68] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *IEEE Trans. Autom. Control*, vol. 43, no. 4, pp. 509–521, Apr. 1998, doi: 10.1109/9.664154.
[69] C. J. Tomlin, J. Lygeros, and S. S. Sastry, "A game theoretic approach to controller design for hybrid systems," *Proc. IEEE*, vol. 88, no. 7, pp. 949–970, Jul. 2000, doi: 10.1109/5.871303.
[70] C. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution manoeuvres," *IEEE Trans. Intell. Transp. Syst.*, vol. 2, no. 2, pp. 110–120, Jun. 2001, doi: 10.1109/6979.928722.
[71] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2016, pp. 1484–1491, doi: 10.1109/ICRA.2016.7487284.
[72] J. H. Gillula, G. M. Hoffmann, H. Huang, M. P. Vitus, and C. J. Tomlin, "Applications of hybrid reachability analysis to robotic aerial vehicles," *Int. J. Robot. Res.*, vol. 30, no. 3, pp. 335–354, Feb. 2011, doi: 10.1177/0278364910387173.
[73] G. M. Hoffmann and C. J. Tomlin, "Decentralized cooperative collision avoidance for acceleration constrained vehicles," in *Proc. 47th IEEE Conf. Decis. Contr.*, 2008, pp. 4357–4363.
[74] C.-F. Lin, J.-C. Juang, and K.-R. Li, "Active collision avoidance system for steering control of autonomous vehicles," *IET Intell. Transp. Syst.*, vol. 8, no. 6, pp. 550–557, 2014, doi: 10.1049/iet-its.2013.0056.
[75] V. Muthukumaran, R. G. Sanfelice, and G. H. Elkaim, "A hybrid control strategy for autonomous navigation while avoiding multiple obstacles at unknown locations," in *Proc. IEEE 15th Int. Conf. Automat. Sci. Eng. (CASE)*, 2019, pp. 1042–1047, doi: 10.1109/COASE.2019.8843308.
[76] D. Phan, J. Yang, R. Grosu, S. A. Smolka, and S. D. Stoller, "Collision avoidance for mobile robots with limited sensing and limited information about moving obstacles," *Formal Methods Syst. Des.*, vol. 51, no. 1, pp. 62–86, Aug. 2017, doi: 10.1007/s10703-016-0265-4.
[77] T. Gurriet, M. Tucker, A. Duburcq, G. Boeris, and A. D. Ames, "Towards variable assistance for lower body exoskeletons," *IEEE Robot. Automat. Lett.*, vol. 5, no. 1, pp. 266–273, Jan. 2020, doi: 10.1109/LRA.2019.2955946.
[78] Q. Nguyen and K. Sreenath, "Safety-critical control for dynamical bipedal walking with precise footstep placement," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 147–154, 2015, doi: 10.1016/j.ifacol.2015.11.167.
[79] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath, "3D dynamic walking on stepping stones with control barrier functions," in *Proc. IEEE 55th Conf. Decis. Contr. (CDC)*, 2016, pp. 827–834, doi: 10.1109/CDC.2016.7798370.
[80] T. Gurriet, A. Singletary, J. Reher, L. Ciarletta, E. Feron, and A. Ames, "Towards a framework for realizable safety critical control through active set invariance," in *Proc. ACM/IEEE 9th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, 2018, pp. 98–106, doi: 10.1109/ICCPS.2018.00018.
[81] W. S. Cortez, D. Oetomo, C. Manzie, and P. Choong, "Control barrier functions for mechanical systems: Theory and application to robotic grasping," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 2, pp. 530–545, Mar. 2021, doi: 10.1109/TCST.2019.2952317.
[82] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Contr. Syst. Lett.*, vol. 5, no. 1, pp. 127–132, Jun. 2020, doi: 10.1109/LCSYS.2020.3000748.
[83] D. Pickem et al., "Safe, remote-access swarm robotics research on the Robotarium," 2016, *arXiv:1604.00640*.
[84] D. Pickem et al., "The Robotarium: A remotely accessible swarm robotics research testbed," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2017, pp. 1699–1706, doi: 10.1109/ICRA.2017.7989200.
[85] S. Wilson et al., "The Robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Syst. Mag.*, vol. 40, no. 1, pp. 26–44, Feb. 2020, doi: 10.1109/MCS.2019.2949973.
[86] S. Wilson et al., "The Robotarium: Automation of a remotely accessible, multi-robot testbed," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2922–2929, Apr. 2021, doi: 10.1109/LRA.2021.3062796.
[87] Y. Chen, M. Jankovic, M. Santillo, and A. D. Ames, "Backup control barrier functions: Formulation and comparative study," 2021, *arXiv:2104.11332*.
[88] A. Isaly, B. C. Allen, R. G. Sanfelice, and W. E. Dixon, "Zeroing control barrier functions for safe volitional pedaling in a motorized cycle," *IFAC-PapersOnLine*, vol. 53, no. 5, pp. 218–223, May 2021, doi: 10.1016/j.ifacol.2021.04.100.
[89] C. M. Holloway, "Understanding the overarching properties," Nat. Aeronaut. Space Admin., Langley Research Center, Washington, DC, USA, 2019. [Online]. Available: https://ntrs.nasa.gov/citations/20190029284
[90] K. Hobbs, J. Davis, L. Wagner, and E. Feron, "Formal specification and analysis of spacecraft collision avoidance run time assurance requirements," in *Proc. IEEE Aerosp. Conf.*, 2021, vol. 1, pp. 1–16, doi: 10.1109/AERO50100.2021.9438135.

[91] D. Swihart et al., "Design, integration and flight test of an autonomous ground collision avoidance system," *Gyroscopy Navig.*, vol. 2, no. 2, pp. 84–91, Apr. 2011, doi: 10.1134/S2075108711020088.
[92] J. B. Lyons et al., "Trust of an automatic ground collision avoidance technology: A fighter pilot perspective," *Mil. Psychol.*, vol. 28, no. 4, pp. 271–277, May 2016, doi: 10.1037/mil0000124.
[93] W. A. Olson, "Airborne collision avoidance system x," Massachusetts Inst. Technol., Lincoln Lab., Lexington, MA, USA, Tech. Rep., 2015. [Online]. Available: https://www.ll.mit.edu/r-d/projects/airborne-collision-avoidance-system-x
[94] J. D. Lee and K. A. See, "Trust in automation: Designing for appropriate reliance," *Hum. Factors*, vol. 46, no. 1, pp. 50–80, Spring 2004, doi: 10.1518/hfes.46.1.50_30392.
[95] R. Turner et al., "Automatic aircraft collision avoidance algorithm design for fighter aircraft," in *Proc. Asia-Pacific Int. Symp. Aerosp. Technol.*, 2012, pp. 1–9.
[96] M. A. Skoog, K. Prosser, and L. Hook, "Ground collision avoidance system (iGCAS)," U.S. Patent 9 633 567, Apr. 2017.
[97] P. Sorokowski, M. Skoog, S. Burrows, and S. Thomas, "Small UAV automatic ground collision avoidance system design considerations and flight test results," Nat. Aeronaut. Space Admin., Washington, DC, USA, Tech. Rep. 20150014106, 2015.
[98] J. D. Carpenter, "Simulation and piloted simulator study of an automatic ground collision avoidance system for performance limited aircraft," Air Force Inst. Technol., Wright-Patterson Air Force Base, Dayton, OH, USA, Tech. Rep., 2019. [Online]. Available: https://scholar.afit.edu/etd/2214/
[99] A. W. Suplisson, "Optimal recovery trajectories for automatic ground collision avoidance systems (Auto GCAS)," Air Force Inst. Technol., Dayton, OH, USA, Tech. Rep., 2015. [Online]. Available: https://scholar.afit.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1182&context=etd
[100] A. Burns, D. Harper, A. F. Barfield, S. Whitcomb, and B. Jurusik, "Auto GCAS for analog flight control system," in *Proc. IEEE/AIAA 30th Digit. Avionics Syst. Conf.*, 2011, pp. 8C5-1–8C5-11, doi: 10.1109/DASC.2011.6096148.
[101] P. Madhavan and D. A. Wiegmann, "Similarities and differences between human–human and human–automation trust: An integrative review," *Theor. Issues Ergonom. Sci.*, vol. 8, no. 4, pp. 277–301, May 2007, doi: 10.1080/14639220500337708.
[102] N. T. Ho, G. G. Sadler, L. C. Hoffmann, J. B. Lyons, and W. W. Johnson, "Trust of a military automated system in an operational context," *Mil. Psychol.*, vol. 29, no. 6, pp. 524–541, Jul. 2017, doi: 10.1037/mil0000189.
[103] K. A. Hoff and M. Bashir, "Trust in automation: Integrating empirical evidence on factors that influence trust," *Hum. Factors*, vol. 57, no. 3, pp. 407–434, May 2015, doi: 10.1177/0018720814547570.
[104] J. Milton and J. C. Arnold, *Introduction to Probability and Statistics*. New York, NY, USA: McGraw-Hill, 2003.
[105] L. Sha, R. Rajkumar, and M. Gagliardl, "A software architecture for dependable and evolvable industrial computing systems," Carnegie-Mellon Univ. Softw. Eng. Inst., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-95-TR-005, 1995.
[106] J. G. Rivera, A. A. Danylyszyn, C. B. Weinstock, L. R. Sha, and M. J. Gagliardi, "An architectural description of the simplex architecture," Carnegie-Mellon Univ. Softw. Eng. Inst., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-96-TR-006, 1996.
[107] C. Reis, A. Barth, and C. Pizano, "Browser security: Lessons from google chrome," *Commun. ACM*, vol. 52, no. 8, pp. 45–49, Aug. 2009, doi: 10.1145/1536616.1536634.
[108] *Standard Practice for Methods to Safely Bound Flight Behavior of Unmanned Aircraft Systems Containing Complex Functions*, ASTM F3269, ASTM International, Conshohocken, PA, USA, 2017.
[109] J. D. Schierman, M. D. DeVore, N. D. Richards, and M. A. Clark, "Runtime assurance for autonomous aerospace systems," *J. Guid., Contr., Dyn.*, vol. 43, no. 12, pp. 2205–2217, Dec. 2020, doi: 10.2514/1.G004862.
[110] S. E. Jones et al., "Automatic integrated collision avoidance system," in *Proc. 17th Australian Int. Aerosp. Congress (AIAC)*, 2017, p. 13.
[111] R. M. Turner et al., "Automatic collision avoidance technology," in *Proc. 18th Australian Int. Aerosp. Congr., HUMS-11th Defence Sci. Technol. Int. Conf. Health Usage Monitoring, ISSFD-27th Int. Symp. Space Flight Dyn. (ISSFD)*, 2019, p. 495.
[112] M. A. Skoog, L. R. Hook, and W. Ryan, "Leveraging ASTM industry standard f3269-17 for providing safe operations of a highly autonomous aircraft," in *Proc. IEEE Aerosp. Conf.*, 2020, pp. 1–7, doi: 10.1109/AERO47225.2020.9172434.
[113] B. Eller et al., "Test and evaluation of a modified F-16 analog flight control computer," in *Proc. AIAA Infotech@ Aerosp. Conf.*, 2013, p. 4726, doi: 10.2514/6.2013-4726.
[114] J. Wadley et al., "Development of an automatic aircraft collision avoidance system for fighter aircraft," in *Proc. AIAA Infotech@ Aerosp. Conf.*, 2013, p. 4727, doi: 10.2514/6.2013-4727.
[115] S. Bak and K. Hobbs, "Efficient n-to-n collision detection for space debris using 4D AABB trees," 2019, *arXiv:1901.10475*.
[116] K. Hobbs, P. Heidlauf, A. Collins, and S. Bak, "Space debris collision detection using reachability," in *Proc. 5th Int. Workshop Appl. Verification Continuous Hybrid Syst.*, 2018, pp. 218–228.
[117] L. R. Hook, M. Clark, D. Sizoo, M. A. Skoog, and J. Brady, "Certification strategies using run-time safety assurance for part 23 autopilot systems," in *Proc. IEEE Aerosp. Conf.*, 2016, pp. 1–10, doi: 10.1109/AERO.2016.7500817.
[118] *Standard Practice for Methods to Safely Bound Behavior of Aircraft Systems Containing Complex Functions Using Run-Time Assurance*, ASTM Std. F3269, ASTM International, Conshohocken, PA, USA, 2021.
[119] U. Mehmood, S. Bak, S. A. Smolka, and S. D. Stoller, "Safe CPS from unsafe controllers," 2021, *arXiv:2102.12981*.
[120] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Contr. Conf. (ECC)*, 2019, pp. 3420–3431, doi: 10.23919/ECC.2019.8796030.
[121] J. Aubin, *Viability theory (Modern Birkhäuser Classics)*. New York, NY, USA: Springer Science & Business Media, 2009.
[122] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017, doi: 10.1109/TAC.2016.2638961.
[123] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, Dec. 2015, doi: 10.1016/j.ifacol.2015.11.152.
[124] S. Prajna, "Barrier certificates for nonlinear model validation," *Automatica*, vol. 42, no. 1, pp. 117–126, Jan. 2006, doi: 10.1016/j.automatica.2005.08.007.
[125] M. Maghenem and R. G. Sanfelice, "Multiple barrier function certificates for weak forward invariance in hybrid inclusions," in *Proc. IEEE 58th Conf. Decis. Contr. (CDC)*, 2019, pp. 6319–6324, doi: 10.1109/CDC40024.2019.9029980.
[126] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proc. 53rd IEEE Conf. Decis. Contr.*, 2014, pp. 6271–6278, doi: 10.1109/CDC.2014.7040372.
[127] P. Glotfelter, J. Cortés, and M. Egerstedt, "Nonsmooth barrier functions with applications to multi-robot systems," *IEEE Contr. Syst. Lett.*, vol. 1, no. 2, pp. 310–315, Oct. 2017, doi: 10.1109/LCSYS.2017.2710943.
[128] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation," in *Proc. Robot., Sci. Syst.*, Jul. 2017, pp. 1–10, doi: 10.15607/RSS.2017.XIII.073.
[129] C. Santoyo, M. Dutreix, and S. Coogan, "A barrier function approach to finite-time stochastic system verification and control," *Automatica*, vol. 125, Jan. 2021, Art. no. 109439. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109820306415, doi: 10.1016/j.automatica.2020.109439.
[130] M. Srinivasan, S. Coogan, and M. Egerstedt, "Control of multi-agent systems with finite time control barrier certificates and temporal logic," in *Proc. IEEE Conf. Decis. Contr. (CDC)*, 2018, pp. 1991–1996, doi: 10.1109/CDC.2018.8619113.
[131] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE Contr. Syst. Lett.*, vol. 3, no. 1, pp. 96–101, Jan. 2019, doi: 10.1109/LCSYS.2018.2853182.
[132] G. Yang, C. Belta, and R. Tron, "Continuous-time signal temporal logic planning with control barrier functions," in *Proc. Amer. Contr. Conf. (ACC)*, 2020, pp. 4612–4618, doi: 10.23919/ACC45564.2020.9147387.
[133] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *Proc. Amer. Contr. Conf. (ACC)*, 2016, pp. 322–328, doi: 10.1109/ACC.2016.7524935.
[134] X. Tan, W. Shaw Cortez, and D. V. Dimarogonas, "High-order barrier functions: Robustness, safety and performance-critical control," *IEEE Trans. Autom. Control*, vol. 67, no. 6, pp. 3021–3028, Jun. 2022, doi: 10.1109/TAC.2021.3089639.
[135] W. Xiao and C. Belta, "Control barrier functions for systems with high relative degree," in *Proc. IEEE 58th Conf. Decis. Contr. (CDC)*, 2019, pp. 474–479.

[136] H. Seywald and R. Kumar, "Desensitized optimal trajectories," Analytical Mech. Assoc., Hampton, VA, USA, Rep. 3–16, 2003.
[137] M. Mote, M. Egerstedt, E. Feron, A. Bylard, and M. Pavone, "Collision-inclusive trajectory optimization for free-flying spacecraft," *J. Guid., Contr., Dyn.*, vol. 43, no. 7, pp. 1–12, Jul. 2020, doi: 10.2514/1.G004788.
[138] K. Zhou and J. C. Doyle, *Essentials of Robust Control*, vol. 104. Upper Saddle River, NJ, USA: Prentice-Hall, 1998.
[139] S. Höfer et al., "Sim2Real in robotics and automation: Applications and challenges," *IEEE Trans. Autom. Sci. Eng. (From July 2004)*, vol. 18, no. 2, pp. 398–400, Apr. 2021, doi: 10.1109/TASE.2021.3064065.
[140] M. Abate and S. Coogan, "Enforcing safety at runtime for systems with disturbances," in *Proc. IEEE 59th Conf. Decis. Contr. (CDC)*, 2020, pp. 2038–2043, doi: 10.1109/CDC42340.2020.9304203.
[141] M. Abate, M. Mote, E. Feron, and S. Coogan, "Verification and runtime assurance for dynamical systems with uncertainty," in *Proc. 24th Int. Conf. Hybrid Syst., Comput. Contr.*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 1–10, doi: 10.1145/3447928.3456656.
[142] C. Llanes, M. Abate, and S. Coogan, "Safety from in-the-loop reachability for cyber-physical systems," in *Proc. Workshop Comput.-Aware Algorithmic Des. Cyber-Phys. Syst. (CAADCPS)*, 2021, pp. 9–10, doi: 10.1145/3457335.3461706.
[143] C. Llanes, M. Abate, and S. Coogan, "Safety from fast, in-the-loop reachability with application to UAVs," in *Proc. ACM/IEEE 13th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, 2022, pp. 127–136, doi: 10.1109/ICCPS54341.2022.00018.
[144] S. Coogan, M. Arcak, and A. A. Kurzhanskiy, "Mixed monotonicity of partial first-in-first-out traffic flow models," in *Proc. IEEE 55th Conf. Decis. Contr. (CDC)*, 2016, pp. 7611–7616, doi: 10.1109/CDC.2016.7799445.
[145] H. L. Smith, "The discrete dynamics of monotonically decomposable maps," *J. Math. Biol.*, vol. 57, no. 4, pp. 309–310, Jan. 2008, doi: 10.1007/s00285-006-0004-3.
[146] C. Muñoz et al., "DAIDALUS: Detect and avoid alerting logic for unmanned systems," in *Proc. IEEE/AIAA 34th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2015, pp. 1–18, doi: 10.1109/DASC.2015.7311588.
[147] C. Baier, J. Katoen, and K. Larsen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.
[148] K. Y. Rozier, "Specification: The biggest bottleneck in formal methods and autonomy," in *Proc. Working Conf. Verified Softw., Theories, Tools, Exp.*, Springer-Verlag, 2016, pp. 8–26, doi: 10.1007/978-3-319-48869-1_2.
[149] R. Avram, X. Zhang, J. A. Muse, and M. Clark, "Nonlinear adaptive control of quadrotor UAVs with run-time safety assurance," in *Proc. AIAA Guid., Navig., Contr. Conf.*, 2017, p. 1896, doi: 10.2514/6.2017-1896.
[150] M. Bodson, J. Lehoczky, R. Rajkumar, L. Sha, and J. Stephan, "Analytic redundancy for software fault-tolerance in hard real-time systems," in *Foundations of Dependable Computing*, vol. 284, G. M. Koob and C. G. Lau, Eds. Boston, MA, USA: Springer-Verlag, 1994, pp. 183–212.
[151] L. Sha, R. Rajkumar, and M. Gagliardi, "Evolving dependable real-time systems," in *Proc. IEEE Aerosp. Appl. Conf.*, 1998, vol. 1, pp. 335–346, doi: 10.1109/AERO.1996.495894.
[152] D. Seto, B. Krogh, L. Sha, and A. Chutinan, "The Simplex architecture for safe online control system upgrades," in *Proc. Amer. Contr. Conf. ACC (IEEE Cat. No. 98CH36207)*, 1998, vol. 6, pp. 3504–3508, doi: 10.1109/ACC.1998.703255.
[153] L. Sha, "Using simplicity to control complexity," *IEEE Softw.*, vol. 18, no. 4, pp. 20–28, Jul./Aug. 2001, doi: 10.1109/MS.2001.936213.
[154] S. Bak, D. K. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha, "The system-level simplex architecture for improved real-time embedded system safety," in *Proc. 15th IEEE Real-Time Embedded Technol. Appl. Symp.*, 2009, pp. 99–107, doi: 10.1109/RTAS.2009.20.
[155] S. Bak, A. Greer, and S. Mitra, "Hybrid cyberphysical system verification with simplex using discrete abstractions," in *Proc. 16th IEEE Real-Time Embedded Technol. Appl. Symp.*, 2010, pp. 143–152, doi: 10.1109/RTAS.2010.27.
[156] R. Alur et al., "The algorithmic analysis of hybrid systems," *Theor. Comput. Sci.*, vol. 138, no. 1, pp. 3–34, Feb. 1995, doi: 10.1016/0304-3975(94)00202-T.
[157] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager, "The theory of timed I/O automata," *Synthesis Lectures Distrib. Comput. Theory*, vol. 1, no. 1, pp. 1–137, 2010.
[158] S. Mitra, "A verification framework for hybrid systems," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2007.
[159] L. R. Hook, M. Skoog, M. Garland, W. Ryan, D. Sizoo, and J. VanHoudt, "Initial considerations of a multi-layered run time assurance approach to enable unpiloted aircraft," in *Proc. IEEE Aerosp. Conf.*, 2018, pp. 1–11, doi: 10.1109/AERO.2018.8396622.
[160] K. H. Gross, "Evaluation of verification approaches applied to nonlinear system control," Master's thesis, Air Force Inst. Technol., Dayton, OH, USA, Mar. 2016.
[161] K. L. Hobbs, I. Perez, A. Fifarek, and E. M. Feron, "Formal verification of system states for spacecraft automatic maneuvering," in *Proc. AIAA Scitech Forum*, 2019, p. 1187, doi: 10.2514/6.2019-1187.
[162] I. Lee, H. Ben-Abdallah, S. Kannan, M. Kim, O. Sokolsky, and M. Viswanathan, "A monitoring and checking framework for run-time correctness assurance," 1998. [Online]. Available: https://repository.upenn.edu/cgi/viewcontent.cgi?article=1313&context=cis_papers
[163] I. Lee, S. Kannan, M. Kim, O. Sokolsky, and M. Viswanathan, "Runtime assurance based on formal specifications," Dept. Papers (CIS), Univ. Pennsylvania, Philadelphia, PA, USA, Jul. 1999. [Online]. Available: https://repository.upenn.edu/cgi/viewcontent.cgi?article=1311&context=cis_papers
[164] C. Torens and F.-M. Adolf, "Formal requirements and model-checking for V&V automation of a RPAS mission management system," in *Proc. AIAA Infotech @ Aerosp.*, Kissimmee, FL, USA, Jan. 2015, pp. 1–13.
[165] M. Wooldridge, "Agents and software engineering," *AI* IA Notizie*, vol. 11, no. 3, pp. 31–37, 1998.
[166] C. A. R. Hoare, "An axiomatic basis for computer programming," *Commun. ACM*, vol. 12, no. 10, pp. 576–580, Oct. 1969, doi: 10.1145/363235.363259.
[167] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem-proving," *Commun. ACM*, vol. 5, no. 7, pp. 394–397, Jul. 1962, doi: 10.1145/368273.368557.
[168] P. Cousot and R. Cousot, "Abstract interpretation frameworks," *J. Logic Comput.*, vol. 2, no. 4, pp. 511–547, Aug. 1992, doi: 10.1093/logcom/2.4.511.
[169] L. De Moura and N. Bjørner, "Satisfiability modulo theories: Introduction and applications," *Commun. ACM*, vol. 54, no. 9, pp. 69–77, Sep. 2011, doi: 10.1145/1995376.1995394.
[170] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Workshop on Logic of Programs*, D. Kozen, Ed. Berlin, Germany: Springer-Verlag, 1981, pp. 52–71.
[171] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Syst.*, vol. 2, no. 4, pp. 255–299, Nov. 1990, doi: 10.1007/BF01995674.
[172] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Y. Lakhnech and S. Yovine, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 152–166.
[173] K. Gross et al., "Evaluation of formal methods tools applied to a 6U cubesat attitude control system," in *Proc. AIAA SPACE Conf. Expo.*, 2015, p. 4529.
[174] K. H. Gross, M. A. Clark, J. A. Hoffman, E. D. Swenson, and A. W. Fifarek, "Run-time assurance and formal methods analysis applied to nonlinear system control," *J. Aerosp. Inform. Syst.*, vol. 14, no. 4, pp. 232–246, Apr. 2017, doi: 10.2514/1.I010471.
[175] K. H. Gross et al., "Formally verified run time assurance architecture of a 6U cubesat attitude control system," in *Proc. AIAA Infotech@ Aerosp.*, 2016, p. 0222, doi: 10.2514/6.2016-0222.
[176] B. Könighofer et al., "Shield synthesis," *Formal Methods Syst. Des.*, vol. 51, no. 2, pp. 332–361, Sep. 2017, doi: 10.1007/s10703-017-0276-9.
[177] C. S. Păsăreanu, D. Giannakopoulou, M. G. Bobaru, J. M. Cobleigh, and H. Barringer, "Learning to divide and conquer: Applying the l* algorithm to automate assume-guarantee reasoning," *Formal Methods Syst. Des.*, vol. 32, no. 3, pp. 175–205, Jan. 2008, doi: 10.1007/s10703-008-0049-6.
[178] D. A. Peled, *Software Reliability Methods*. New York, NY, USA: Springer Science & Business Media, 2013.
[179] K. H. Gross, A. W. Fifarek, and J. A. Hoffman, "Incremental formal methods based design approach demonstrated on a coupled tanks control system," in *Proc. IEEE 17th Int. Symp. High Assurance Syst. Eng. (HASE)*, 2016, pp. 181–188, doi: 10.1109/HASE.2016.16.
[180] M. W. Whalen, A. Gacek, D. Cofer, A. Murugesan, M. P. Heimdahl, and S. Rayadurgam, "'Your 'what' is my 'how': Iteration and hierarchy in system design," *IEEE Softw.*, vol. 30, no. 2, pp. 54–60, Mar. 2013, doi: 10.1109/MS.2012.173.
[181] T. A. Henzinger, M. Minea, and V. Prabhu, "Assume-guarantee reasoning for hierarchical hybrid systems," in *Proc. Int. Workshop Hybrid Syst., Comput. Contr.*, M. D. Di Benedetto and A. Sangiovanni-Vincentelli, Eds. Berlin, Germany: Springer-Verlag, 2001, pp. 275–290.
[182] M. C. L. Abate, "Mixed monotonicity for efficient reachability with applications to robust safe autonomy," Ph.D. dissertation, Georgia Inst. Technol., Atlanta, GA, USA, 2022.