From Formal Requirements to Highly Assured Software for Unmanned Aircraft Systems

César Muñoz, Anthony Narkawicz, and Aaron Dutle

NASA Langley Research Center, Hampton, Virginia 23681-2199
{cesar.a.munoz,anthony.narkawicz,aaron.m.dutle}@nasa.gov

Abstract. Operational requirements of safety-critical systems are often written in restricted specification logics. These restricted logics are amenable to automated analysis techniques such as model-checking, but are not rich enough to express complex requirements of unmanned systems. This short paper advocates for the use of expressive logics, such as higher-order logic, to specify the complex operational requirements and safety properties of unmanned systems. These rich logics are less amenable to automation and, hence, require the use of interactive theorem proving techniques. However, these logics support the formal verification of complex requirements such as those involving the physical environment. Moreover, these logics enable validation techniques that increase confidence in the correctness of numerically intensive software. These features result in highly-assured software that may be easier to certify. The feasibility of this approach is illustrated with examples drawn for NASA's unmanned aircraft systems.

1 Introduction

Recent advances in theorem proving technology have prompted the development of environments such as ASSERT (Analysis of Semantic Specifications and Efficient generation of Requirements-based Test) [17] and SpeAR (Specification and Analysis of Requirements) [6] that provide English-like, but semantically rigorous, languages to capture requirements. These requirements are formally analyzed for consistency using automated theorem proving tools. Environments such as ASSERT and SpeAR are examples of the state-of-the-art in formal requirements design. Yet, the kinds of requirements that can be analyzed using these environments are those supported by automated verification techniques, which are typically limited to finite state machines and decidable theories supported by SMT solvers. These formalisms are not rich enough to allow for the specification of complex requirements of cyber-physical systems. This short paper reports work by the Formal Methods (FM) Team at NASA Langley Research Center (LaRC) on the use of higher-order logic and interactive theorem proving for the specification, analysis, and implementation of operational and functional requirements of unmanned aircraft systems (UAS).

2 UAS Detect and Avoid

In 2011, the UAS Sense and Avoid Science and Research Panel (SARP) was tasked with making a recommendation to the FAA for a quantitative definition of a concept for UAS called *well clear*. The origin of this concept is the see-and-avoid principle in manned aircraft operations that states that on-board pilots have, in part, the responsibility for not "operating an aircraft so close to another aircraft as to create a collision hazard", "to see and avoid other aircraft", and when complying with the particular rules addressing right-of-way, on-board pilots "may not pass over, under, or ahead [of the right-of-way aircraft] unless well clear" [7]. The lack of a similar principle for UAS was identified by the FAA as one of the main obstacles in the integration of UAS in the National Airspace System. Hence, the final report of the Federal Aviation Administration (FAA) Sense and Avoid (SAA) Workshop [5] defined the concept of *sense and avoid*, also called *detect and avoid* (DAA), as "the capability of a UAS to remain well clear from and avoid collisions with other airborne traffic."

Consiglio et al. proposed the following guiding principles for the definition of well clear and DAA requirements: (a) The well-clear concept should be geometrically represented by a time and distance volume in the airspace, (b) DAA should interoperate with existing collision avoidance systems, (c) DAA should avoid undue concern for traffic aircraft, and (d) DAA should enable self-separation capabilities [1]. Based on these guidelines, a family of well-clear volumes was formally specified in the Program Verification System (PVS) [14]. This family is defined by a Boolean predicate, representing a volume in the airspace, that depends on the position and velocity of the ownship and intruder aircraft at the current time. It was formally verified that volumes in this family satisfy several properties such as inclusion, symmetry, extensibility, local convexity, and convergence [8,10]. These properties were used by the UAS SARP to discard competing proposals for the well-clear volume. The volume ultimately recommended by the UAS SARP [3] is based on distance and time functions used in the detection logic of the second generation of the Traffic Alerting and Collision Avoidance System (TCAS II) Resolution Advisory (RA) detection logic [16]. Since this volume is a member of the family specified in PVS, it inherits the family's formally verified properties. For example, it has been formally verified that for a choice of threshold values, the well-clear volume is larger than the TCAS II RA volume [10] (inclusion) and that in pairwise encounter both aircraft simultaneously compute the same well-clear status (symmetry). The use of higher-order logic enabled the definition of this family of volumes that is not only parametric with respect to distance and time thresholds, but also with respect to continuous functions on positions and velocities.

The standards organization RTCA established Special Committee 228 (SC-228) to provide technical guidance to the FAA for defining minimum operational performance standards for a DAA concept based on the definition of well-clear recommended by the UAS SARP. This concept consists of three functional capabilities: detection logic, alerting logic, and maneuver guidance logic. The detection logic specifies a time interval where a well-clear violation occurs, within a lookhead time interval, assuming non-accelerating aircraft trajectories. A parametric algorithm that computes this time interval, for an arbitrary choice of the threshold values used in the definition of the well-clear volume, has been formally verified in PVS [9]. This parametric algorithm is key to the definition of the alerting logic, which is specified by a series of thresholds that yield volumes of decreasing size. Depending on the time to violation of these volumes, the alerting logic returns a numerical value representing the severity of a predicted conflict. The smaller the volume and the shorter the time to violation, the greater the severity. It has been formally verified that the alerting logic satisfies the following operational properties (assuming non-accelerating aircraft trajectories) [8,10]: (extensibility) alerts progress according to the severity level; (local convexity) once an alert is issued, it is continuously issued until threat disappears; and (convergence) once an alert is issued, it does not disappear before time of closest point of approach. Finally, the manuever guidance logic specifies ranges of one-dimensional maneuvers, i.e., change of horizontal direction, change of horizontal speed, change of vertical speed, or change of altitude, that lead to a well-clear violation within a lookahead time interval. In the case of a well-clear violation, the maneuver guidance logic specifies ranges of maneuvers that recover well-clear status. Assuming a kinematic model of the ownship trajectories, an algorithm that computes maneuver guidance for each dimension has been formally proved to be correct within a user specified granularity. This algorithm is parametric with respect to a detection algorithm for an arbitrary definition of the well-clear volume. These algorithms are collectively called DAIDALUS (Detect and Avoid Alerting Logic for Unmanned Systems) [11] and they are included in RTCA DO-365 [15].

3 From DAIDALUS to ICAROUS

Software implementations of DAIDALUS are available in Java and C++ and they are distributed under NASA's Open Source Agreement.[1] The PVS specifications and proofs are also available as part of the distribution. Except for language idiosyncrasies both implementations are identical and they closely follow the PVS algorithms. A formal verification of the software implementations is a major endeavor that has not been attempted. In particular, DAIDALUS algorithms are formally verified in PVS assuming real-number arithmetic, while the software implementations of DAIDALUS use floating-point arithmetic. However, the software implementations of DAIDALUS have been validated against the PVS algorithms using model animation [4] on a set of stressing cases. This validation improves the assurance that the hand translation from formal models to code is faithful and that floating point errors do not greatly affect correctness and safety properties that are formally verified in PVS.

The approach used in the development of DAIDALUS, from formal requirements to highly assured software, is called MINERVA [13] (Mirrored Implementation Numerically Evaluated against Rigorously Verified Algorithms). The

[1] `https://github.com/nasa/wellclear`.

MINERVA approach has been used in the development of other UAS applications. PolyCARP, a collection of algorithms for weather avoidance and geofencing has been formally developed in PVS [12]. Implementations of PolyCARP in Java, C++, and Python have been validated using model validation. This development, including software, specifications, and proofs, is available under NASA's Open Source Agreement.[2] DAIDALUS and PolyCARP are two of the algorithms included in ICAROUS (Independent Configurable Architecture for Reliable Operations of Unmanned Systems) [2]. ICAROUS is an open software architecture composed of mission specific software modules and highly assured core algorithms for building autonomous unmanned aircraft applications.[3]

4 Conclusion

The examples presented in this paper show that the use of expressive formalisms, such as PVS, for writing requirements and formally analyzing them is not only feasible but effective in the development of safety-critical systems. Higher-order logic enables, for example, the specification and formal analysis of generic models that can be instantiated and reused in multiple ways. DAIDALUS algorithms, for example, can be instantiated with different set of thresholds, different notions of well-clear, and different aircraft performance characteristics. The default configuration of DAIDALUS defined in DO-365 is appropriate for large fixed wing UAS. In ICAROUS, however, DAIDALUS is instantiated with the performance of a small rotorcraft and a smaller set of thresholds that define a cylindrical well-clear volume. The formal models are correct for both of any instantiation due to the parametric nature of the models.

However, rich formalisms such as higher-order logic are not the silver bullet. These expressive logics are typically undecidable and, in the case of PVS, even type-checking is undecidable. Interactive theorem proving is a human intensive activity and the tools are still difficult to use by system developpers. Applications such as DAIDALUS, PolyCARP, and ICAROUS are possible because of years of fundamental developments in interactive theorem proving technology including formal libraries and proof strategies. The call in this paper is for the integration of this infrastructure, which is also available in other proofs assistants, in modern requirement engineering tools like ASSERT and SPEAR.

References

1. María Consiglio, James Chamberlain, César Muñoz, and Keith Hoffler. Concept of integration for UAS operations in the NAS. In *Proceedings of 28th International Congress of the Aeronautical Sciences, ICAS 2012*, Brisbane, Australia, 2012.
2. María Consiglio, César Muñoz, George Hagen, Anthony Narkawicz, and Swee Balachandran. ICAROUS: Integrated Configurable Algorithms for Reliable Operations of Unmanned Systems. In *Proceedings of the 35th Digital Avionics Systems Conference (DASC 2016)*, Sacramento, California, US, September 2016.

[2] `https://github.com/nasa/PolyCARP`.

[3] `https://github.com/nasa/ICAROUS`.

3. Stephen P. Cook, Dallas Brooks, Rodney Cole, Davis Hackenberg, and Vincent Raska. Defining well clear for unmanned aircraft systems. In *Proceedings of the 2015 AIAA Infotech @ Aerospace Conference*, number AIAA-2015-0481, Kissimmee, Florida, January 2015.
4. Aaron Dutle, César Muñoz, Anthony Narkawicz, and Ricky Butler. Software validation via model animation. In Jasmin Blanchette and Nikolai Kosmatov, editors, *Proceedings of the 9th International Conference on Tests & Proofs (TAP 2015)*, volume 9154 of *Lecture Notes in Computer Science*, pages 92–108, L'Aquila, Italy, July 2015. Springer.
5. FAA Sponsored Sense and Avoid Workshop. Sense and avoid (SAA) for Unmanned Aircraft Systems (UAS), October 2009.
6. Aaron W. Fifarek, Lucas G. Wagner, Jonathan A. Hoffman, Benjamin D. Rodes, M. Anthony Aiello, and Jennifer A. Davis. SpeAR v2.0: Formalized past LTL specification and analysis of requirements. In Clark Barrett, Misty Davies, and Temesghen Kahsai, editors, *NASA Formal Methods*, pages 420–426, Cham, 2017. Springer International Publishing.
7. International Civil Aviation Organization (ICAO). Annex 2 to the Convention on International Civil Aviation, July 2005.
8. César Muñoz and Anthony Narkawicz. Formal analysis of extended well-clear boundaries for unmanned aircraft. In Sanjai Rayadurgam and Oksana Tkachuk, editors, *Proceedings of the 8th NASA Formal Methods Symposium (NFM 2016)*, volume 9690 of *Lecture Notes in Computer Science*, pages 221–226, Minneapolis, MN, June 2016. Springer.
9. César Muñoz, Anthony Narkawicz, and James Chamberlain. A TCAS-II resolution advisory detection algorithm. In *Proceedings of the AIAA Guidance Navigation, and Control Conference and Exhibit 2013*, number AIAA-2013-4622, Boston, Massachusetts, August 2013.
10. César Muñoz, Anthony Narkawicz, James Chamberlain, María Consiglio, and Jason Upchurch. A family of well-clear boundary models for the integration of UAS in the NAS. In *Proceedings of the 14th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, number AIAA-2014-2412, Georgia, Atlanta, USA, June 2014.
11. César Muñoz, Anthony Narkawicz, George Hagen, Jason Upchurch, Aaron Dutle, and María Consiglio. DAIDALUS: Detect and Avoid Alerting Logic for Unmanned Systems. In *Proceedings of the 34th Digital Avionics Systems Conference (DASC 2015)*, Prague, Czech Republic, September 2015.
12. Anthony Narkawicz and George Hagen. Algorithms for collision detection between a point and a moving polygon, with applications to aircraft weather avoidance. In *16th AIAA Aviation Technology, Integration, and Operations Conference, AIAA AVIATION Forum*, number AIAA-2016-3598, Washington, DC, USA, June 2016.
13. Anthony Narkawicz, César Muñoz, and Aaron Dutle. The MINERVA software development process. In *Proceedings of the Workshop on Automated Formal Methods 2017 (AFM 2017)*, Meno Park, California, USA, 2017.
14. Sam Owre, John Rushby, and Natarajan Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752. Springer-Verlag, June 1992.
15. RTCA SC-1228. RTCA-DO-365, Minimum Operational Performance Standards for Detect and Avoid (DAA) Systems, May 2017.
16. RTCA SC-147. RTCA-DO-185B, Minimum Operational Performance Standards for Traffic alert and Collision Avoidance System II (TCAS II), July 2009.

17. Kit Siu, Abha Moitra, Michael Durling, Andy Crapo, Meng Li, Han Yu, Heber Herencia-Zapana, Mauricio Castillo-Effen, Shiraj Sen, Craig McMillan, Daniel Russell, Sundeep Roy, and Panagiotis Manolios. Flight critical software and systems development using ASSERT. In *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, number 978-1-5386-0365-9/17, pages 1–10, Sept 2017.