A Formal Approach to Multi-UAV Route Validation[⋆]

Toby Wilkinson[⋆⋆], Michael Butler, Martin Paxton, and Xanthippe Waldron

University of Southampton, UK
{stw08r, mjb}@ecs.soton.ac.uk
TEKEVER Ltd, UK
{martin.paxton, xanthippe.waldron}@tekever.com

Abstract. We present ongoing work to apply Event-B to the validation of routes for an Unmanned Aircraft System consisting of a Ground Control Station and two or more UAVs. We extend the mathematical language of Event-B to include a theory of continuous paths in 3-D Euclidean space that allows important safety properties describing the safe separation of UAVs to be formalised in a natural manner. Refinement of the model allows a mathematically verified route validator algorithm to be systematically derived.

Keywords: Event-B, Autonomous Systems, Verification.

1 Introduction

The capabilities of Unmanned Aircraft Systems (UAS) are rapidly increasing, but this makes the task of demonstrating that such systems are safe increasingly difficult. This difficulty manifests itself in two ways; the inherent complexity of the systems themselves, and the compressed timescales under which these systems are developed.

To address these issues engineers need better ways to capture and analyse the system requirements, and better ways for building modular systems, in which the verification evidence for individual components can be combined to provide verification evidence for the system as a whole. DO-178C [7] opens the door to the use of formal methods in the certification of civil aircraft, and this approach is also applicable to the development of Unmanned Aircraft Systems.

An emerging scenario is for a single UAS to have multiple UAVs that collaborate to perform a collection of tasks. For such a system, even if it is deployed in

[⋆] This document is an overview of MOD sponsored research and is released to inform projects that include safety-critical or safety-related software. The information contained in this document should not be interpreted as representing the views of the MOD, nor should it be assumed that it reflects any current or future MOD policy. The information cannot supersede any statutory or contractual requirements or liabilities and is offered without prejudice or commitment.

[⋆⋆] Supported by the ASUR Programme project 1014_C6_PH1_104.

segregated airspace, there is still a safety requirement that at all times the UAVs remain a safe distance apart. The routes that are assigned to the UAVs must therefore be checked to determine that they are safe. One possible architecture for such a UAS is to have a separate route validator function that checks the output of the planning function that generates the routes. The idea being that the route validator can be made simpler than the route generator, and therefore its correctness more easily shown.

In this paper we present ongoing work to demonstrate how a route validator function can be systematically derived from a natural mathematical formulation of the properties required of a set of routes (one per UAV) to be safe. We do this by extending the mathematical language of the formal method Event-B with a mathematical theory of continuous paths in 3-D Euclidean space. This allows safety properties like minimum safe separation to be naturally and clearly formalised. Then through systematic refinement the model can be refined to one yielding a mathematically verified algorithm for route validation. A key step in the refinement chain is the data refinement of the continuous path associated with each UAV with a corresponding list of waypoints. The gluing invariant then reconstructs each UAV's route as the piecewise linear path between the waypoints. Proof obligations are generated to ensure that these paths are indeed continuous.

In a final step we aim to (manually at this stage) produce code that implements the verified algorithm.

2 Event-B

Event-B [2] is a development of the B-Method [1] that extends the scope of formal modelling to allow for reasoning at the system level. This distinguishes Event-B from traditional program verification methods like Hoare logic [5] that only allow for reasoning about the software parts of a system.

Event-B uses set theory and first-order logic as a modelling notation, and one of the key features of Event-B is the support for refinement. Through refinement a system can be represented at different levels of abstraction, and mathematical proof is used to verify the consistency between refinement levels.

The modelling approach is to construct an initial abstract model that describes the main purpose of the system, and successively refine this model to layer in other features. Each refinement step modifies or extends the model in order to introduce new functionality, or add further details of how some goal is achieved. The resulting chain of refinements facilitates abstraction since it allows us to postpone treatment of some system features to later refinement steps. Abstraction and refinement together allow us to manage system complexity.

System properties that must always remain true are defined in a model as *invariants* on the model variables, and Event-B defines proof obligations to ensure that the invariants are preserved by all events. The refinement of a model (usually referred to as the abstract model) by another model (referred to as the concrete model) also generates proof obligations, and these must be discharged in order to show that any behaviour of the concrete model satisfies the abstract behaviour. *Gluing invariants* are used to specify the relationship between the variables of the abstract and concrete models, and give rise to proof obligations for pairs of abstract and corresponding concrete events.

The Rodin Platform is an Eclipse-based open source IDE for Event-B that provides effective support for refinement and mathematical proof [3]. Rodin includes automatic tools to generate the proof obligations associated with a model, and provers that attempt to automatically discharge these obligations.

Rodin can also be extended with plug-ins that provide additional functionality, and these include:

1. ProB a model checker and animator [6].
2. The UML-B plug-in which supports modelling and refinement of class diagrams and state machines, and translates models into Event-B [8].
3. The Theory plug-in which supports extension of the mathematical modelling language of Event-B, and associated proof rules for the Rodin provers [4].

3 Case Study: Multi-UAV Route Validation

Our case study is route validation for a UAS (Unmanned Aircraft System) consisting of a Ground Control Station (GCS) and two or more UAVs. The case study itself is synthetic, but has been developed in conjunction with TEKEVER Ltd [1], and is therefore representative of a real system.

In this hypothetical system, the GCS (in conjunction with the operator) generates a route for each of the UAVs (possibly generated by some planning algorithm based upon a set of tasks), and these are then checked by a route validator to determine that they are safe.

The route validator must ensure that a set of routes (one per UAV) satisfy the following safety properties:

1. Mutual deconfliction (safe separation between UAVs).
2. That the UAVs stay within airspace constraints (the segregated airspace assigned to the UAS and any no-fly areas).
3. That the UAVs avoid terrain and other obstacles.

The main focus in the case study will be on the first of these, the need to maintain safe separation between UAVs.

The output from the case study will be a (manually generated) implementation of the mathematically verified route validator. TEKEVER aim to test this implementation in one of their systems, however a complete solution will require further development. For example, one question that would need to be addressed is how do the new routes sit with respect to a UAV's current position, capabilities, and status?

[1] TEKEVER is a company headquartered in Portugal, with an Aerospace, Security, Defence and Space division focusing on the development of unmanned systems, including UAVs. The company has a number of different UAV platform types ranging from smaller hand-launched aircraft through to >4m wingspan multi-sensor platforms that are being developed in part for maritime search and patrolling missions.

4 Our Approach

The idea behind our approach, is that through the development of domain specific mathematical theories, it is be possible to provide an environment in which complex safety properties can be easily and clearly stated. Moreover, such an environment can facilitate experimentation with different formulations of system safety properties during the requirements gathering and analysis phases of system development.

For example, in our case study two possible formal statements of the deconfliction safety property may be:

$$\forall a, b \in U \;.\; \forall t \in \mathbb{R} \;.\; a \neq b \Rightarrow \mathit{min\_sep} \leq d(r_a(t), r_b(t)),$$

or

$$\forall a, b \in U \;.\; \forall t \in \mathbb{R} \;.\; a \neq b \Rightarrow \forall t_a, t_b \in (t - \Delta t, t + \Delta t) \;.\; \mathit{min\_sep} \leq d(r_a(t_a), r_b(t_b)).$$

Here U is a set of UAVs, and the first statement says that for any pair a and b of distinct UAVs, and any time t, then the distance between the position $r_a(t)$ of a at time t, and the position $r_b(t)$ of b, is at least $\mathit{min\_sep}$.

The second statement is similar, but now allows for the fact that the UAVs may not keep to the time schedule associated with their routes, and instead may arrive early or late at any point. Here Δt is some fixed bound on how early or late we believe a UAV could be.

In both formulations, $r_a \colon \mathbb{R} \to \mathbb{R}^3$ is a continuous path in 3-D Euclidean space, and $d \colon \mathbb{R}^3 \times \mathbb{R}^3 \to [0, \infty)$ is the usual Euclidean metric. Thus in our formalisation of the safety properties we are working with real numbers (defined axiomatically and including the least upper bound axiom) and not floating point approximations, and moreover we potentially have access to the full expressive power of continuous mathematics and the differential and integral calculus. [2]

To allow the above safety properties to be directly stated in a model of the route validator we are using the Theory plug-in to extend the mathematical language of Event-B to include a theory of 3-D Euclidean space (defined as a normed real vector space with an inner product and the usual metric).

At the abstract level the routes passed to the route validator are simply arbitrary continuous paths, and safety invariants can be expressed directly as above. Then using data refinement the route associated with each UAV is replaced by a list of waypoints, pairs of position and time, where each waypoint represents the location of the UAV at the specified time. The gluing invariant for the data refinement defines how the route for each UAV can be reconstructed as a piecewise linear path between the waypoints.

We have developed within the Theory plug-in a mathematical theory of continuous paths and piecewise linear paths, and defined a mathematical function

[2] The Rodin theory of real numbers currently has a definition of continuous functions, but not differentiation or integration. Adding the basic definitions would be easy (within the Theory plug-in), but to be useable all the basic theorems of a first course in real analysis would need to be included too.

that generates the piecewise linear path associated with a list of waypoints. Proof obligations were generated to show that the piecewise linear path produced by this function was indeed a continuous path, and this then established that the data refinement from routes to waypoint lists was sound.

Work is ongoing to refine the model by splitting the events into more fine-grained steps until a verified algorithm for the route validator is produced. The aim is to do this in a modular fashion so that different formulations of the safety properties can be tried, with the corresponding changes to the algorithm localised, and the re-verification (proof) effort minimised.

During the project we also aim to produce code (at this stage manually) that implements the verified algorithms, however, the modelled algorithms will be expressed in terms of genuine real variables, whereas any implementation in software will necessarily use floating point variables. There is therefore the possibility that the coded implementations could deviate from the behaviour of the mathematically verified algorithms. We hope to address this issue in future work, but it is out of scope for this project.

5 Conclusion and Future Work

The approach outlined above shows much promise for the formal development of multi-UAV route planning systems. Future work could include development of more complete specifications and algorithms for the route validator function, or the development using the Theory plug-in of a theory of floating point numbers, and further data refinement of the model from real to floating point variables.

References

1. Abrial, J.-R.: The B-Book: Assigning Programs to Meanings. Cambridge University Press (1996)
2. Abrial, J.-R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press (2010)
3. Abrial, J.-R., Butler, M., Hallerstede, S., Hoang, T.S., Mehta, F., Voisin, L.: Rodin: an open toolset for modelling and reasoning in Event-B. International Journal on Software Tools for Technology Transfer 12(6), 447–466 (2010)
4. Butler, M., Maamria, I.:Practical Theory Extension in Event-B. In: Liu, Z. and Woodcock, J. and Zhu, H. (eds.) Theories of Programming and Formal Methods – Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday, LNCS, vol. 8051, pp. 67–81. Springer (2013)
5. Hoare, C.A.R.: An axiomatic basis for computer programming. Commun. ACM 12(10), 576–580 (1969)
6. Leuschel, M., Butler, M.: ProB: an automated analysis toolset for the B method. International Journal on Software Tools for Technology Transfer 10(2), 185–203 (2008)
7. RTCA, Inc.: DO-178C, Software Considerations in Airborne Systems and Equipment Certification (December 2011)
8. Snook, C., Butler, M.: UML-B: Formal modeling and design aided by UML. ACM Transactions on Software Engineering and Methodology 15(1), 92–122 (2006)