A FRAMEWORK FOR PROBABILISTIC EVALUATION OF INTERVAL MANAGEMENT TOLERENCE IN THE TERMINAL RADAR CONTROL AREA

Heber Herencia-Zapana, National Institute of Aerospace, Hampton, VA

George Hagen, NASA Langley Research Center, Hampton, VA

Natasha Neogi, National Institute of Aerospace, Hampton, VA

Abstract

Projections of future traffic in the national airspace show that most of the hub airports and their attendant airspace will need to undergo significant redevelopment and redesign in order to accommodate any significant increase in traffic volume. Even though closely spaced parallel approaches increase throughput into a given airport, controller workload in oversubscribed metroplexes is further taxed by these approaches that require stringent monitoring in a saturated environment. The interval management (IM) concept in the TRACON area is designed to shift some of the operational burden from the control tower to the flight deck, placing the flight crew in charge of implementing the required speed changes to maintain a relative spacing interval. The interval management tolerance is a measure of the allowable deviation from the desired spacing interval for the IM aircraft (and its target aircraft). For this complex task, Formal Methods can help to ensure better design and system implementation.

In this paper, we propose a probabilistic framework to quantify the uncertainty and performance associated with the major components of the IM tolerance. The analytical basis for this framework may be used to formalize both correctness and probabilistic system safety claims in a modular fashion at the algorithmic level in a way compatible with several Formal Methods tools.

Introduction

The transportation of people and goods through the air is a critical part of our country's infrastructure and economy. The Next Generation Air Transportation System (NextGen) seeks to transform the current centrally-controlled, voice-communication-based air transportation system into an information-rich, highly automated, and agile system that is safer, more environmentally acceptable, and sufficiently scalable and adaptable to allow for large increases in air traffic and system disruptions.

NextGen will require an evolutionary plan to provide automation tools to support controllers and pilots in flexible, collaborative decision making as well as to assure necessary emergent properties (such as system safety) over a heterogeneous mix of equipage, algorithms and operational procedures (see Figure 1). System wide fault tolerance is necessary. This is a distinct property from the fault-tolerance of individual components of the system. Taken in the context of software agents interacting, even if an algorithm is provably correct, a component may fail, and an algorithm's implementation may fail due to faults arising from such problems as unexpected latency in communication, cumulative sensor errors, garbled messages, computational errors between algorithm variants, or even malicious attacks. These faults can then propagate in unexpected ways to other components of the overall system, and means must be in place to mitigate such occurrences.

One new concept for NextGen is **interval management** (IM), which relies on the notion of trajectory based operations in concert with improved capabilities in computer technology and a move from ground-based navigation aids to satellite-based navigation systems to increase capacity. At the heart of the concept is a shift in flight planning and separation responsibility away from ground-based ATC to the flight deck, where pilots will be able to make decisions on the routes, altitudes, and speeds of their aircraft, both tactically and strategically based on their intent. Decentralized decisions, however, can impact global optimality and performance.

We present a modular framework for assessing the safety of IM algorithms by developing formal specifications of the components, their interactions and the necessary set of safety properties. This

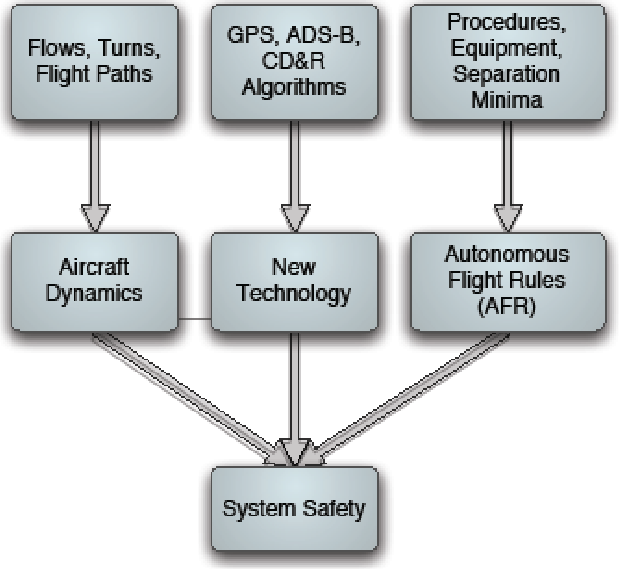
Flows, Turns, Flight Paths
GPS, ADS-B, CD&R Algorithms
Procedures, Equipment, Separation Minima
Aircraft Dynamics
New Technology
Autonomous Flight Rules (AFR)
System Safety

Figure 1. System safety

framework allows a close and direct linkage between probabilistic analysis and the algorithm implementation.

Probabilistic analysis is typically performed using a model of the system. We are seeking here to instead link the probabilistic analysis more directly to the system at the code level. We hope this rigorous and systematic examination of the system will eliminate overlooked special cases of the system's algorithms and allow for a more complete study of the implementation's behavior.

This paper is divided as follows. The first section presents the interval management application, where we describe the relation between the IM concept and an algorithm that could be used to implement it. The second section describes the components of the speed control algorithm. The third section shows the correctness of the speed control algorithm. The fourth section illustrates how it is possible to do probabilistic analysis at the algorithm level. Finally the last section gives the conclusions and directions for future research.

Interval Management

The airspace involves interactions of numerous entities: aircraft with continuous dynamics, control algorithms (both onboard and in control towers) with discrete logic, human decision makers (both onboard aircraft and in control towers), sensors and actuators, and communications channels, as well as the flight rules that govern operational procedures.

Algorithms are an important part of the NextGen concept. Different aspects of NextGen may involve numerous different proposed algorithms. For example, an overview of aircraft separation algorithms can be found in [1]. This paper focuses on an algorithm performing the interval management operation, incorporating probability analysis to improve its design.

The main actors of the IM operation are the IM aircraft, with a position and air speed denoted by $\mathbf{x_{IM}(t)}$, $\mathbf{v_{IM}(t)}$ and the target aircraft, with position and air speed denoted by $\mathbf{x_T(t)}$, $\mathbf{v_T(t)}$.

Interval management operations can be divided in three phases: negotiation, where aircraft on two different trajectories determine how they will both follow a common trajectory (the order of merging), action, where they merge onto a common trajectory, and the terminal approach phase, where one aircraft follows the descent path of the other [2]. This work focuses on the third phase, terminal approach. In this phase aircraft have already merged and are flying on a single leg in a straight path to the terminal. The dynamic of the IM and target aircraft are described by the following differential equations:

$$\frac{\mathbf{dx}}{\mathbf{dt}} = \mathbf{v}\cos\boldsymbol{\gamma}\sin\boldsymbol{\Psi}$$

$$\frac{\mathbf{dv}}{\mathbf{dt}} = \frac{\boldsymbol{T}\boldsymbol{cos}\boldsymbol{\alpha} - \mathbf{D}}{\boldsymbol{m}} - \mathbf{g}\sin\boldsymbol{\gamma}$$

Where $\boldsymbol{\gamma}$ is the flight path, $\mathbf{v}$ true air speed, $\boldsymbol{\Psi}$ heading angle, $\boldsymbol{\alpha}$ angle of attack, $\mathbf{T}$ thrust, $\mathbf{D}$ drag, $\mathbf{g}$ gravity and $\mathbf{m}$ mass.

The actual position of the aircraft performing the interval management (the “own ship”) along its trajectory path is denoted by $\mathbf{x_{IM}(t)}$, and the actual position of the target aircraft along its trajectory path is denoted by $\mathbf{x_T(t)}$. The measured positions and velocities of the aircraft are sampled with some fixed frequency (in the case of ADS-B messages, every 1 second). The positions and velocities are based on GPS data and are sent via ADS-B communication channels, all subject to errors.

There are several potential means of determining the proper spacing of aircraft on approach. While in all cases the ultimate goal is to maintain a certain physical separation distance **D**, in practice this may also be maintained through temporal spacing, which simulations have suggested produces better performance [3,4,5]. If the target aircraft passes through a certain point at time **t**, then the IM aircraft is required to pass through the same point at a later time $\mathbf{t+\tau}$, with a small amount of leeway.

While under interval management, if two aircraft are separated by less than the specified minimum distance **D** at a given instant in time, then the amount of overlap is called the spacing error. When keeping aircraft separated temporally, the range error is used instead. The **range error** is the distance between the along-path position of the IM aircraft at time **t** and the along-path position of the target aircraft at time $\mathbf{t-\tau}$, where $\boldsymbol{\tau}$ is the desired time spacing interval between the aircraft. The range error is calculated by $\mathbf{e(t)=x_{IM}(t)-x_T(t-\tau)}$ [6].

The speed command applied to the IM aircraft is a function of the range error and the speed of the target aircraft $\mathbf{v^c} = \mathbf{F(e(t),V_T(t))}$. There are several ways to calculate the speed command [5-8].

We model the IM aircraft in such a way that we assume that the thrust and drag are the primary means of changing the IM aircraft dynamics. Therefore altering the values of these variables represents the application of the speed command [9].

The IM speed control algorithm relies upon ADS-B position and velocity measurements broadcast by the target aircraft, where the position and velocity accuracy depends on the accuracy of the GPS readings. Let $\boldsymbol{\varepsilon_s}$ be a predicate that is true if the measured position and velocity values are within expected tolerances.

Speed Control Algorithm

The desired speed command, $\mathbf{v^c}$, is a speed that will preserve the IM spacing that is defined by the range error being less than some distance **D**, as follows $\mathbf{|e(t)|=|x_{IM}(t)-x_{Tar}(t-\tau)|<D}$, and **conflict** means that there exists a **t** such that $\mathbf{|e(t)|\geq D}$.

The implementation of the speed command to the IM aircraft would be an algorithm with the following imperative instructions and special actions:

- `read`: read the position and speed of IM and target aircraft and store them
- `cd` is the conflict detection algorithm, where given the position and speed it calculates if there is a conflict in the time window between two subsequent messages. If a conflict is detected, then it returns a true value, denoted by `cd`=T, and `cd`=F otherwise.
- `comp_maneuver`: compute the desired speed, the thrust, and drag and store them
- `¬act`: null action. The algorithm does not change the aircraft dynamic.
- `act`: The algorithm alters the aircraft thrust and drag in order to get the desired speed command
- `comp_time`: estimate the time to transition the desired speed command and the time it takes to perform these calculations and store this value
- `alarm`: issue an alarm. The algorithm cannot solve the conflict
- `cdw`**:** returns `cdw`=T if the conflict detection algorithm determines there will be a conflict in the time window stored by the `comp_time` command. If there is no projected conflict, `cdw`=F

The speed control algorithm (with desired pre- and post conditions) is as follows:

```
read
IF cd=F THEN
    ¬act
ELSE
    comp_maneuver
    comp_time
    IF cdw=F THEN
        act
    ELSE
        alarm
    ENDIF
```

```
ENDIF
```

If no conflict is detected, then no corrective action is taken. If a conflict is detected, however, then a corrective maneuver will be calculated. If the maneuver is projected to be able to resolve the conflict, then it will be enacted, otherwise an alarm will be raised.

Our analysis is primarily concerned with missing an actual conflict. The noted precondition is that the sensor values must be within acceptable tolerances and there is an actual conflict. That means there is the possibility that the conflict is not detected (possibly due to measurement errors), as well as the possibility of failure in other steps of the algorithm. The end result of executing the algorithm should be that, if a conflict exists, it is either resolved or else an alert is issued.

Speed Control Algorithm Correctness

The correctness of an algorithm is related to the goals of the process to be carried out. Informally, the set of beliefs concerning the purpose of the algorithm is referred to as its specification. We can then say that the algorithm is correct with respect to its specifications if, for the valid range of input data accepted by the algorithm, the result produced by the algorithm is both predicted and repeatable. An algorithm that always produces the expected answer(s) if it terminates is said to be partially correct. An algorithm that is always guaranteed to terminate, given the resource bounds detailed in the specifications, is regarded as being *feasible* and *correct*. Note that if the resource bounds are not met, the algorithm may not terminate.

Program verification based on deductive methods uses either automatic decision procedures or proof assistants to ensure the validity of user-annotated code. These annotations often express domain-specific properties of the code. However, formulating annotations correctly (i.e., as precisely as the domain expert really intends) is nontrivial in practice. The challenges of producing domain specific code annotation arise along two directions. First, the domain knowledge has its own inherent complexity. In this interval management application, for example, the annotations are required to capture the expression of system-wide safety properties. Second, the code annotations are required to be stated in a manner that can be interpreted by some theorem proving software. The logical language supported by a particular verification tool may be too weak to express the desired user defined and domain specific code annotations. Many automatic decision procedures, for example, are limited to bounded integer arithmetic o, at most, rational linear algebra.

In order to solve these two challenges this paper proposes to use **Hoare logic** [10]. Hoare logic is a formal system with a set of logical rules for reasoning about the correctness of a program. The central feature of Hoare logic is the **Hoare triple**. A Hoare triple describes how the execution of a piece of program changes the state of the computation. The triple has the form **{P}S{Q}** where **P** and **Q** are assertions indicating pre-conditions and post-conditions and **S** is a command. The basic idea is that, given some **P,** after executing **S**, **Q** will hold. It is possible to annotate the algorithm as follows (with ∧ being the logical *and* symbol and ∨ being the symbol for logical *or*, and the predicate **do(**X**)** indicate the command X is to be performed):

$\{\varepsilon_s \wedge \textbf{conflict}\}$
IF cd=F **THEN** ¬act **ELSE s' ENDIF**
$\{\neg\textbf{conflict} \vee \textbf{do(}\mathtt{alarm}\textbf{)}\}$

Where **s'** =

```
comp_maneuver
comp_time
IF cdw=F THEN
    act
ELSE
    alarm
ENDIF
```

This annotation indicates the desired result: given the state information is within acceptable bounds and there is a conflict, once the speed control algorithm is executed then either the conflict will be resolved or an alarm will be issued.

Using the Hoare logic the speed control algorithm is equivalent to:

$$\{\varepsilon_s \wedge \textbf{conflict} \wedge \mathtt{cd}{=}\mathrm{F}\}\neg\mathtt{act}\ \{\neg\textbf{conflict}\}$$

or

$$\{\varepsilon_s \wedge \textbf{conflict} \wedge \mathtt{cd}{=}\mathrm{T}\}\textbf{s'}\{\neg\textbf{conflict} \vee \textbf{do(}\mathtt{alarm}\textbf{)}\}$$

We propose that, using Hoare logic with annotations, it is possible to prove the correctness of this sort of interval management algorithm. One way to prove the correctness would be to use certain program analysis tools such as Frama-C[11,12].[1]

The algorithm correctness is with respect to its annotations, but there are still uncertainties inherent in the information provided by the annotations. The next section focuses in the analysis of these uncertainties.

Speed Control Algorithm Uncertainty

The **IM tolerance |e(t)|<D** is a measure of the allowable deviation from the desired spacing interval for the IM aircraft (and its target aircraft) during the execution of the algorithm. The IM tolerance represents the bounds on the fault free spacing precision that must be achieved and maintained by an IM aircraft implementing the flight deck based speed control algorithm; it is usually quantified in a probabilistic fashion as **Pr[conflict] ≤p** [13,14]**.**

The IM tolerance is directly affected by the quality of the state data, attained through GPS and other sensors and ADS-B with probability **Pr[$\mathcal{E}_s$].** Here we will look at two cases: missed alerts and false alerts.

Missed Alerts

The main idea is to analyze how the state data uncertainty in the measurements can affect the IM tolerance. The goal is to never reach **{conflict}** after applying the speed control algorithm no matter whether the condition **cd** is true or false. In the event that a conflict is unavoidable, the algorithm should raise an alert.

Let recall our basic speed control algorithm:

IF `cd`=F **THEN** ¬`act` **ELSE s' ENDIF**

Using the annotations, we wish to prove that the probability of the following happening is acceptably low:

$\{\mathcal{E}_s \wedge \textbf{conflict}\}$

IF `cd`=F **THEN** ¬`act` **ELSE s' ENDIF**

$\{\textbf{conflict} \wedge \neg\textbf{do}(\texttt{alarm})\}$

The precondition indicates that the sensor values are within the expected range with some probability and that there exists an actual conflict. The post-condition of this undesirable situation is that after the algorithm executes there still a conflict and no alarm has been indicated.

The IM tolerance **Pr[conflict]≤p** and the probability of the sensor error **Pr[$\mathcal{E}_s$]** could be obtained from aviation standards such as [16]. The question to be answered is what would be the probabilistic requirements for the `cd` and **s'** components of the algorithms such that the IM tolerance is held with the required probability.

The speed control algorithm is required to hold the IM tolerence with a certain probability **p**:

$$\mathbf{Pr}\begin{bmatrix}\{\mathcal{E}_s \wedge \textbf{conflict} \wedge \texttt{cd} = \mathbf{F}\}\neg\texttt{act}\{\textbf{conflict}\} \\ \{\mathcal{E}_s \wedge \textbf{conflict} \wedge \texttt{cd} = \mathbf{T}\}\mathbf{s'} \\ \qquad\qquad \{\textbf{conflict} \wedge \neg\textbf{do}(\texttt{alarm})\}\end{bmatrix} \le \mathbf{p}$$

And this is equal to:

$$\mathbf{Pr}[\{\mathcal{E}_s \wedge \textbf{conflict} \wedge \texttt{cd} = \mathbf{F}\}\neg\texttt{act}\{\textbf{conflict}\}] + \mathbf{Pr}\begin{bmatrix}\{\mathcal{E}_s \wedge \textbf{conflict} \wedge \texttt{cd} = \mathbf{T}\}\mathbf{s'} \\ \qquad\qquad \{\textbf{conflict} \wedge \neg\textbf{do}(\texttt{alarm})\}\end{bmatrix} \le \mathbf{p}$$

For how this if-construct affects the probability, see [17].

This inequality is called the uncertainty budget. The uncertainty budget analysis consists of checking the necessary conditions for each module of the control speed algorithm in order to preserve the uncertainty budget. The following is one possible, though conservative, way of dividing the uncertainty budget among subcomponents.

We know that if both of the left-hand side terms of the uncertainty budget are less than **p/2** then the IM tolerance will hold.

Let us first focus on the first element of the uncertainty budget, which is equal to

$$\mathbf{Pr}\left[\frac{\neg\text{act}\,\{\textbf{conflict}\}}{\{\boldsymbol{\varepsilon}_s \wedge \textbf{conflict} \wedge \text{cd} = \text{F}\}}\right] \times \mathbf{Pr}[\{\boldsymbol{\varepsilon}_s \wedge \textbf{conflict} \wedge \texttt{cd} = \mathbf{F}\}] \le \frac{\mathbf{P}}{\mathbf{2}}$$

[1] These specific tools do not require an annotation at each line as proposed by Hoare. Instead they rely on the Dijkstra-style weakest precondition calculus to compute the backward semantics of the function code **S** to the post-condition **Q** and generate the weakest pre-condition **wp(S;Q)** that is guaranteed to obtain **Q** after executing **S**. What actually needs to be proved is that this weakest pre-condition holds [15].

Because we assume a conflict actually exists, outside of any external forces acting on the aircraft, the first probability is equal to one. Therefore,

$$\mathbf{Pr}[\boldsymbol{\varepsilon}_s \wedge \mathbf{conflict} \wedge \texttt{cd=F}\}] \leq \frac{\mathbf{p}}{\mathbf{2}}$$

and this is equal to

$$\mathbf{Pr}\left[\frac{\mathbf{conflict} \wedge \mathrm{cd=F}}{\boldsymbol{\varepsilon}\mathbf{s}}\right] \times \mathbf{Pr}[\boldsymbol{\varepsilon}_s] \leq \frac{\mathbf{p}}{\mathbf{2}}$$

The first element of this inequality is the probability of missing the alert due to sensor error; such an analysis can be seen in [18,19]. We then need to guarantee that the product of the sensor error probability and missed alert probability needs to be less than **p**/2. One way to hold the inequality is if the two expressions are

[c1] $$\mathbf{Pr}\left[\frac{\mathbf{conflict} \wedge \mathrm{cd=F}}{\boldsymbol{\varepsilon}\mathbf{s}}\right] \leq \sqrt{\frac{\mathbf{p}}{\mathbf{2}}}$$

and

[c2] $$\mathbf{Pr}[\boldsymbol{\varepsilon}_s] \leq \sqrt{\frac{\mathbf{p}}{\mathbf{2}}}$$

This mean that we need to guarantee that both the probability of missing the alert because of sensor error and that the probability of the sensor error needs to be less than $\sqrt{\mathbf{p}/2}$**.**

Now we perform the same analysis for the second term of the uncertainty budget.

$$\mathbf{Pr}\left[\frac{\mathbf{s'}\{\mathbf{conflict} \ \wedge \ \neg\mathbf{do}(\mathrm{alarm})\}}{\{\boldsymbol{\varepsilon}\mathbf{s} \ \wedge \ \mathbf{conflict} \ \wedge \ \mathrm{cd} = \mathrm{T}\}}\right] \times \mathbf{Pr}[\boldsymbol{\varepsilon}_s \wedge \ \mathbf{conflict} \ \wedge \ \mathbf{cd} = \mathrm{T}\}] \leq \ \mathbf{p/2}$$

Then the required probability for the subalgorithm **s'** could be

[c3] $$\mathbf{Pr}\left[\frac{\mathrm{s'}\{\mathbf{conflict} \wedge \neg\mathbf{do}(\mathrm{alarm})\}}{\{\boldsymbol{\varepsilon}\mathbf{s} \wedge \mathbf{conflict} \wedge \mathrm{cd=T}\}}\right] \leq \sqrt{\frac{\mathbf{p}}{\mathbf{2}}}$$

And a conservative value for the second component is

[c4] $$\mathbf{Pr}\left[\frac{\mathbf{conflict} \wedge \mathrm{cd=T}}{\boldsymbol{\varepsilon}\mathbf{s}}\right] \times \mathbf{Pr}[\boldsymbol{\varepsilon}_s] \leq \sqrt{\frac{\mathbf{p}}{\mathbf{2}}}$$

Now having these inequalities, one possible condition to preserve the IM tolerance is as follows:

Lemma: if the inequalities [c1,..,c4] hold then the uncertainty budget holds.

The basic idea for the proof is to use basic inequalities properties such as if $a \leq c/2$ and $b \leq c/2$ then $a + b \leq c$, and if $a \leq \sqrt{c}$ and $b \leq \sqrt{c}$ then $a \times b \leq c$, where a,b,c are positive real numbers.

The uncertainty budget analysis allows us to see how, given data for the uncertainty of the sensor measurement and IM tolerance it is possible to put bounds to the probability of missing conflict alerts, as well as other pieces of code such as **s'.**

False Alerts

The other main case of concern is a false positive in the initial detection, which will either induce an unnecessary maneuver (which itself could potentially result in a new conflict) or signal a false alarm. This can be captured with the pre- and post conditions:

{ε$_s$ ∧ ¬**conflict}**
IF `cd=F` **THEN** ¬`act` **ELSE s' ENDIF**
{do(`act)` ∨ **do**(`alarm)`**}**

In this situation, the speed control algorithm should do nothing with a certain probability **p'**:

$$\mathbf{Pr}\begin{bmatrix}\{\mathcal{E}_s \wedge \neg\mathbf{conflict} \wedge \texttt{cd} = \mathbf{F}\}\neg\texttt{act} \\ \{\mathbf{do}(\mathrm{act}) \vee \mathbf{do}(\mathrm{alarm})\} \\ \{\mathcal{E}_s \wedge \neg\mathbf{conflict} \wedge \texttt{cd} = \mathbf{T}\}\mathbf{s'} \\ \{\mathbf{do}(\mathrm{act}) \vee \mathbf{do}(\mathrm{alarm})\}\}\end{bmatrix} \leq \mathbf{p'}$$

And this is equal to:

$$\mathbf{Pr}\begin{bmatrix}\{\mathcal{E}_s \wedge \neg\mathbf{conflict} \wedge \texttt{cd} = \mathbf{F}\}\neg\texttt{act} \\ \{\mathbf{do}(\mathrm{act}) \vee \mathbf{do}(\mathrm{alarm})\}\end{bmatrix} + \mathbf{Pr}\begin{bmatrix}\{\mathcal{E}_s \wedge \neg\mathbf{conflict} \wedge \texttt{cd} = \mathbf{T}\}\mathbf{s'} \\ \{\mathbf{do}(\mathrm{act}) \vee \mathbf{do}(\mathrm{alarm})\}\}\end{bmatrix} \leq \mathbf{p'}$$

From the algorithm, the probability of the first term is equal to zero, so the probability of a false alert is found by calculating:

$$\mathbf{Pr}[\boldsymbol{\varepsilon}_s \wedge \neg\mathbf{conflict} \wedge \texttt{cd=T}\}] \leq \mathbf{p'}$$

Conclusion

The safety argument for NextGen interval management concepts relies on the ability to reliably detect conflicts. Because of unavoidable errors in sensor and transmission data, we cannot guarantee that the probability of a missed alert is zero. We can, however, provide arguments that a given algorithm will only produce missed alerts with a probability within a certain tolerance. We have proposed a framework that links the probabilities associated with the subcomponents of an algorithm through annotations in the code, and have provided a simple interval management algorithm along with its annotations. The annotations used in this framework are compatible with several Formal Methods tools; these can be used to demonstrate the (partial) correctness of an algorithm. In addition, the annotations used to perform a probabilistic analysis of the algorithm in the form of an uncertainty budget, and we sketch this process for the interval management algorithm.

As future research we would like to formalize the annotations in an appropriate machine-readable format (such as in ACSL), which would allow us to automatically verify the logical partial correctness of the algorithm in a tool set such as Frama-C. While such an automatic tool is unlikely to be able to perform the full analysis, it should be able to treat the probabilistic terms as uninterpreted and create proof obligations that could then be checked in an interactive theorem prover such as PVS [20]. Additional research will also be necessary to find reasonable means of determining the appropriate probabilities and bounds for projected sensor data, such as is used in s'.

References

[1] Kuchar, J. k, L. C. Yang, 2000, A review of conflict detection and resolution modeling methods. IEEE Transactions on Intelligent Transportation Systems, 1:179-189

[2] Twu, P.; Chipalkatty, R.; Rahmani, A.; Egerstedt, M.; Young, R, 2010, Air traffic maximization for the terminal phase of flight under FAA's NextGen framework. Digital Avionics Systems Conference (DASC), IEEE/AIAA 29th

[3] Ivanescu, D, C. Shaw, E. Hoffman and K. Zeghal, 2006, Towards Performance Requirements for Airborne Spacing a Sensitivity Analysis of Spacing Accuracy. 6th AIAA Aviation Technology, integration and Operations Conference (ATIO)

[4] Ivanescu, D, D. Powell, C. Shaw, E. Hoffman and K. Zeghal, 2004, Effect of aircraft self-merging in sequence on an airborne collision avoidance system. AIAA Guidance. Navigation and Control Conference and Exhibit

[5] Vinken, P., Hoffman, E., and Zeghal, K 2000, Influence of Speed and Altitude Profile on the Dynamics of the In-Trail Following Aircraft. American Institute of Aeronautics and Astronautics, Inc

[6] Weitz, L, 2011, Investigating string stability of a time-history control law for Interval Management, Transportation Research Part C: Emerging Technologies

[7] Hoffman, E, Dan Ivanescu, Chris Shaw, Karim Zeghal, 2002, Analysis of Spacing Guidance for Sequencing Aircraft on Merging Trajectories. 21st Digital Avionics Systems Conference, Irvine, California, October 2002

[8] Itoh, E, M. Everdij, B. Bakker and H. Blom, 2009, Speed Control for Airborne Separation Assistance in Continuous Descent Arrivals. 9th AIAA Aviation Technology, Integration, and Operations Conference.

[9] Nuic, A, C. Poinsot, M. Iagaru, E. Gallo, F. A. Navarro, C. Querejeta, 2005, Advanced Aircraft Performance Modeling for ATM: Enhancements to the BADA Model. 24th Digital Avionics System Conference, Washington D.C.

[10] Hoare, C.A.R. 1969, An axiomatic basis for computer programming. Comm. ACM 12, 576-580

[11] Correnson, L., Cuoq, P., Puccetti, A., Signoles, J.: Frama-C user manual

[12] Burghardt, J., Gerlach, J., Hartig, K, 2010 ACSL by example towards a verified C standard library version 4.2.0 for Frama-C beryllium 2

[13] Levitt, I, L.A. Weitz, 2011, Towards Defining Required Interval Management Performance, Ninth USA/Europe Air Traffic Management Research and Development Seminar (ATM)

[14] Mohleji, S, G. Wang, 2010, Modeling ADS-B Position and Velocity Errors for Airborne Merging

and Spacing in Interval Management Application. The MITRE Corporation, technical report

[15] Dijkstra, E, 1976, A Discipline of Programming. Prentice-Hall (1976)

[16] Minimum aviation system performance standards for automatic dependent surveillance broadcast (ADS-B). DO-242A, RTCA (June 2002), section 2.1.2.12-2.1.2.15

[17] Cousot, P, M, Monerau, 2012, Probabilistic Abstract Interpretation. In Proceedings of the 22nd European Symposium on Programming (ESOP 2012). Lecture Notes in Computer Science, vol. 7211, pages 166--190, Springer-Verlag, Heidelberg

[18] Narkawicz, A, C. Muñoz and H. Herencia-Zapana and G. Hagen. Formal Verification of Lateral and Temporal Safety Buffers for State-Based Conflict Detection, Institution of Mechanical Engineers, Part G, Journal of Aerospace Engineering, 2012

[19] Herencia-Zapana, H, G. Hagen, A. Narkawicz. Formalizing Probabilistic Safety Claims, NASA Formal Methods, Lecture Notes in Computer Science Springer Berlin, 2011

[20] Owre, S., Shankar, N., Rushby, J.M., J.Stringer-Calvert, D.W,1999, PVS Language Reference. Computer Science Laboratory, SRI International

Acknowledgements

This work is supported in part by the National Aeronautics and Space Administration under NASA Cooperative Agreement NNL09AA00A, activity 2736.