How Formal Methods Impels Discovery: A Short History of an Air Traffic Management Project

Ricky W. Butler, George Hagen, Jeffrey M. Maddalon, César A. Muñoz, Anthony Narkawicz
NASA Langley Research Center
Hampton, VA 23681, USA
{r.w.butler,george.hagen,j.m.maddalon,c.a.munoz,anthony.narkawicz}@nasa.gov
Gilles Dowek
École polytechnique and INRIA
LIX, École polytechnique
91128 Palaiseau Cedex, France
gilles.dowek@polytechnique.fr

Abstract

In this paper we describe a process of algorithmic discovery that was driven by our goal of achieving a complete, mechanically verified algorithm for conflict prevention bands for use in enroute air traffic management. The algorithms were originally defined in the PVS specification language and subsequently have been implemented in Java and C++. We do not present the proofs in this paper. Instead, we describe the process of discovery and the key ideas that enabled the final formal proof of correctness to be completed.

1 Introduction

The formal methods team at NASA Langley has developed air traffic separation algorithms for the last 10 years. Given the safety-critical nature of these algorithms, we have emphasized formal verification of the correct operation of these algorithms. In February of 2008 Ricky Butler and Jeffrey Maddalon started a project to develop and formally verify algorithms that compute conflict prevention bands for en-route aircraft.

In air traffic management systems, a conflict prevention system surveys nearby aircraft and provides ranges of maneuvers that avoid conflicts with these aircraft The maneuvers are typically constrained to ones where only one parameter is varied at a time: track angles, vertical speeds, or ground speeds. Such maneuvers are easy for pilots to fly and they also have the advantage that they can be presented in terms of prevention bands. Prevention bands display the maneuvers that result in conflict within a specified lookahead time as a function of one parameter. Without conflict prevention information, a pilot might enter into a secondary conflict while seeking to solve a primary conflict or otherwise changing his flight plan.

Avoiding potential conflicts involves analyzing possible maneuvers of the aircraft. These ranges of guidance maneuvers are often referred to as *conflict-prevention information*. The National Aerospace Laboratory (NLR) refers to their conflict prevention capability as Predictive Airborne Separation Assurance System or Predictive ASAS [3]. The NLR approach provides two sets of bands: near-term conflicts (within 3 minutes) are shown in red, while intermediate-term conflicts (within 5 minutes) are shown in amber as illustrated in Figure 1. We did not directly analyze the NLR system because the algorithms were not available to us.

When we began this project, we had no idea that this project would take almost two years to complete and that four additional formal methods researchers would join our effort before we were done. This project has been one of the most interesting and enjoyable projects that we have worked on in our careers. The reason for this is manifold, but the following aspects of this project contributed to this interest and

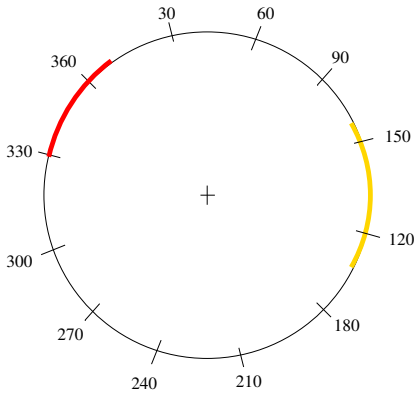

Figure 1: Compass Rose with Conflict Prevention Bands

enjoyment: (1) the work resulted in a very elegant algorithm that is implemented in Java and C++, (2) the final algorithm was very different from our first ideas, (3) there were many, many discoveries that were surprising. At some points in the project we were having daily insights that improved the algorithm or a proof, (4) on the surface the problem looks simple, but looks can be deceiving and the problem is actually very subtle with many special cases. After studying the problem for over a year, we developed an algorithm and rigorously documented this algorithm in a NASA Technical Memorandum [8]. We also formalized much of the mathematical development in the paper. We planned to follow up this paper with another paper that documented a complete formal proof of this algorithm. Much to our surprise this final formal proof step found some deficiencies in our algorithm. These deficiencies were repaired and a formal proof in the Prototype Verification System (PVS) [9] was finally completed in November 2009.

In this paper we will present a brief history of this project and highlight how the goal of formally verifying the algorithm in the PVS theorem prover pushed us to new discoveries. We are sure that many of the discoveries would not have been made if we had taken a more traditional approach of constructing algorithms and testing them until they *worked*.

2 Notation

We consider two aircraft, the *ownship* and the *intruder* aircraft, that are potentially in conflict in a 3-dimensional airspace. The prevention bands algorithm discussed here is intended to be used as a *tactical* conflict prevention system. Tactical refers to the use of state-based information, e.g,. initial position and velocity, straight line trajectories, e.g., constant velocity vectors in an Euclidean airspace, and short lookahead time (typically 5 to 10 minutes).

We use the following notations:

$\mathbf{s}_o$	3D vector	Initial position of the ownship aircraft
$\mathbf{v}_o$	3D vector	Initial velocity of the ownship aircraft
$\mathbf{s}_i$	3D vector	Initial position of the traffic aircraft
$\mathbf{v}_i$	3D vector	Initial velocity of the traffic aircraft

The components of each vector are scalar values, so they are represented without the bold-face font, for example $\mathbf{s}_o = (s_{ox}, s_{oy}, s_{oz})$. As a simplifying assumption, we regard the position and velocity vectors as accurate and without error. For notational convenience, we use $\mathbf{v}^2 = \mathbf{v} \cdot \mathbf{v}$ and we denote by $\mathrm{gs}(\mathbf{v})$ the

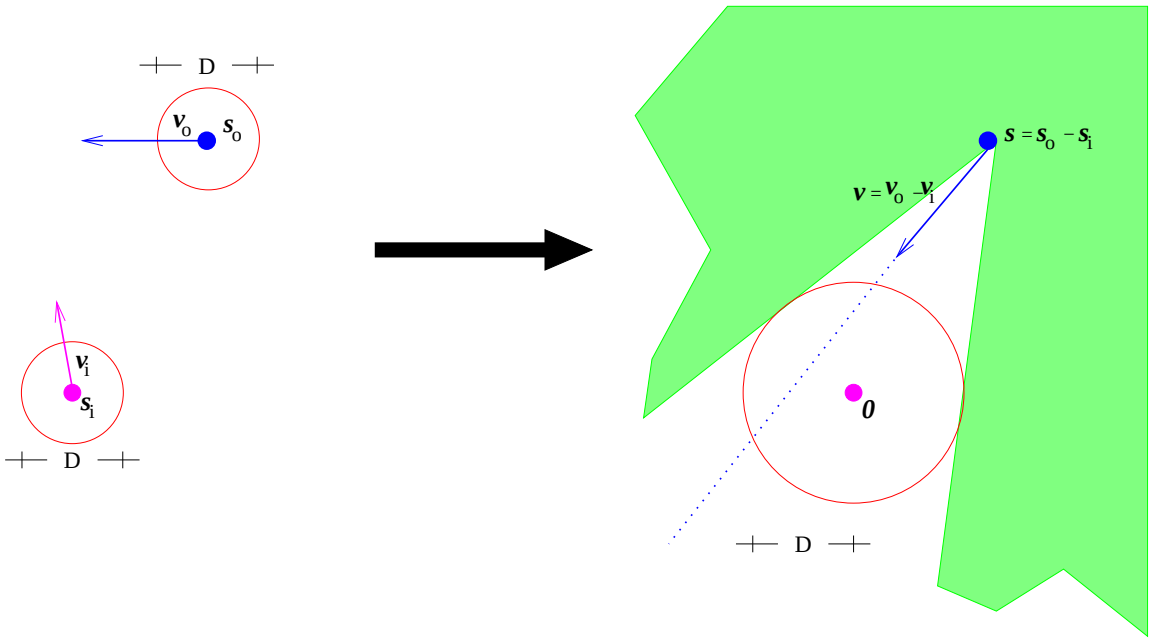

Figure 2: Translated Coordinate System

ground speed of $\mathbf{v}$, i.e., the norm of the 2-dimensional projection of $\mathbf{v}$:

$$\mathrm{gs}(\mathbf{v}) \quad = \quad \sqrt{v_x^2 + v_y^2}. \tag{1}$$

In the airspace system, the separation criteria are specified as a minimum horizontal separation D and a minimum vertical separation H (typically, D is 5 nautical miles and H is 1000 feet). It is convenient to develop the theory using a translated coordinate system. The relative position $\mathbf{s}$ is defined to be $\mathbf{s} = \mathbf{s}_o - \mathbf{s}_i$ and relative velocity of the ownship with respect to the traffic aircraft is denoted by $\mathbf{v}$. With these vectors the traffic aircraft is at the center of the coordinate system and does not move. For example in the left side of Figure 2, the blue (upper) point represents the ownship with its velocity vector and its avoidance area (circle of diameter D around the aircraft). The magenta (lower) point represents the intruder aircraft. The right side represents the same information in the translated coordinate system. In a 3-dimensional airspace, the separation criteria defines a cylinder of radius D and half-height H around the traffic aircraft. This cylinder is called the *protected zone*.

In Figure 2, the two aircraft are potentially in conflict because the half-line defined by the relative velocity vector $\mathbf{v}$ intersects the protected area, meaning that in some future time the ownship will enter the protected zone around the traffic. If this future time is within the lookahead time T, the two aircraft are said to be in *conflict*. Each point in the green (dashed) area represents a possible relative vector that avoids the protected area. Each one of these vectors yields a new velocity vector for the ownship that resolves the potential conflict.

3 The Start of the Project

We began our project by first surveying the literature for previous solutions. Hoekstra [4] describes in his PhD dissertation the NLR algorithms with some diagrams [3], but unfortunately did not provide much detail about how the algorithms actually worked. We decided to develop our own track angle, ground

speed, and vertical speed algorithms. In this paper, we will only present the track bands. These are the most challenging and interesting of the three and the other bands are computed and verified using analogous methods. We adopted the NLR idea of introducing two parameters, T_{red} (typically 3 minutes) and T_{amber} (typically five minutes), which divide the set of conflicts based on their nearness (in time) to a loss of separation. If a loss of separation will occur within T_{red}, then the region is colored red. On the other hand, if a loss of separation will occur after T_{red}, but before T_{amber}, then the region is colored amber, otherwise it is painted green.

We first recognized that each aircraft's contribution to the prevention band is independent of all other aircraft; thus, the problem neatly decomposes into two steps:

1. Solve the bands problem for the ownship relative to each other aircraft separately.

2. Merge all of the pairwise regions.

We also quickly realized that an iterative solution was possible for the first step. We already had a formally proven, efficient algorithm available to us named `CD3D` that decides if a conflict occurs for specific values of $\mathbf{s}_o$, $\mathbf{v}_o$, $\mathbf{s}_i$, and $\mathbf{v}_i$, and parameters D, H, and T. More formally, `CD3D` determines whether there exists a future time t where the aircraft positions $\mathbf{s}_o + t\mathbf{v}_o$ and $\mathbf{s}_i + t\mathbf{v}_i$ are within a horizontal distance D of each other *and* where the aircraft are within vertical distance H of each other. In other words there is a predicted loss of separation within the lookahead time. Therefore, one needs only to execute `CD3D` iteratively, varying the track angle from 0 to 360^o per traffic aircraft. By running different scenarios, we determined that a step size of 0.1^owould be adequate for ranges of up to 200 nautical miles. This iterative approach may not scale well in airborne systems, where tactical separation assurance algorithms will be typically run at frequencies of about 1Hz. Furthermore, this brute-force approach unnecessarily consumes the much needed on-board computational resources .

4 Search For an Analytical Solution

To solve the prevention bands problem in an analytical way, it is useful to define separate horizontal and vertical notions of conflict. In the relative coordinate system, we define

An *horizontal conflict* occurs if there exists a future time t within the lookahead time T where the aircraft are within horizontal distance D of each other, i.e.,

$$(s_x + tv_x)^2 + (s_y + tv_y)^2 \quad < \quad D^2. \tag{2}$$

where $\mathbf{s}$ and $\mathbf{v}$ are, respectively, the relative position and the relative velocity of the ownship with respect to the intruder aircraft.

A *vertical conflict* occurs if there exists a future time t within the lookahead time T where the aircraft are within horizontal distance H of each other, i.e.,

$$|s_z + tv_z| \quad < \quad H. \tag{3}$$

We say that two aircraft are in *conflict* if there is a time t where they are in horizontal and vertical conflict. Formally, we define the predicate `conflict?` as follows

$$\texttt{conflict?}(\mathbf{s},\mathbf{v}) \quad \equiv \quad \exists\, 0 \le t \le T : (s_x + tv_x)^2 + (s_y + tv_y)^2 < D^2 \text{ and } |s_z + tv_z| < H. \tag{4}$$

4.1 Track Only Geometric Solution

For given vectors $\mathbf{v}_o$ and $\mathbf{v}_i$, we need to find the track angles α such that the relative vector

$$\mathbf{v}_\alpha = (\mathrm{gs}(\mathbf{v}_o)\,\mathtt{cos}\,\alpha - v_{ix},\ \mathrm{gs}(\mathbf{v}_o)\,\mathtt{sin}\,\alpha - v_{iy},\ v_{oz} - v_{iz}), \tag{5}$$

is not in conflict.

For simplicity, we decided to first solve the track bands problem without consideration of the finite lookahead time. We also decided to ignore vertical speed considerations and look at the horizontal plane only. The problem thus reduced to finding the tangent lines to the horizontal protection zone (in the relative frame of reference) as a function of α. We begin with the observation that in order for a vector to be tangent it must intersect the circle of the protection zone. In other words, we need solutions of

$$\mathbf{s} + t\mathbf{v}_\alpha = D \tag{6}$$

or equivalently

$$(\mathbf{s} + t\mathbf{v}_\alpha)^2 = D^2 \tag{7}$$

Expanding we obtain a quadratic equation $at^2 + bt + c = 0$ with

$$\begin{aligned} a &= {V_\alpha}^2 \\ b &= 2\mathbf{s}\cdot V_\alpha \\ c &= \mathbf{s}^2 - D^2 \end{aligned}$$

The tangent lines are precisely those where the discriminant of this equation is zero. In other words, where $b^2 - 4ac = 0$. But, expanding the dot products yield:

$$\begin{aligned} b^2 &= 4[s_x(\omega\,\mathtt{cos}\,\alpha - v_{ix}) + s_y(\,\omega\,\mathtt{sin}\,\alpha - v_{iy})]^2 \\ 4ac &= 4(\omega^2 - 2\omega(v_{ix}\,\mathtt{cos}\,\alpha + v_{iy}\mathtt{sin}\,\alpha) + v^2)(\mathbf{s}\cdot\mathbf{s} - D^2) \end{aligned}$$

The discriminant finally expands into a complex second-order polynomial in $\mathtt{sin}\alpha$ and $\mathtt{cos}\alpha$. But to solve for α, we need to eliminate the $\mathtt{cos}\,\alpha$ using the equation

$$\mathtt{cos}\alpha = \sqrt{1 - \mathtt{sin}^2\alpha}$$

The net result is an unbelievably complex fourth order polynomial in $\mathtt{sin}\,\alpha$. Solving for α analytically would require the use of the quartic formulas. Although these formulas are complicated, such a program could probably be written in a day or two. But, how would we verify these solutions? After all, the quartic equations involve the use of complex analysis. Therefore, we begin to look for simplifications.

After seeking simplifications of the formula above, we found a simplification of the discriminant that had been used in the verification of the KB3D algorithm [6]:

$$(\mathbf{s}\cdot\mathbf{v})^2 - \mathbf{v}^2(\mathbf{s}^2 - D^2) = 0 \text{ if and only if } \mathbf{s}\cdot\mathbf{v} = R\varepsilon\,\mathtt{det}(\mathbf{s},\mathbf{v}), \tag{8}$$

where $\varepsilon \in \{-1,+1\}$, $\mathtt{det}(\mathbf{s},\mathbf{v}) \equiv \mathbf{s}^\perp\cdot\mathbf{v}$, $\mathbf{s}^\perp = (-s_y, s_x)$, and $R = \frac{\sqrt{\mathbf{s}^2 - D^2}}{D}$. The beauty of the final form is that the equation is linear on $\mathbf{v}$. The two solutions are captured in the two values of ε. When we instantiate $\mathbf{v}_\alpha$ in this formula, we end up with a quadratic equation in $\mathtt{sin}\,\alpha$.

Using this approach, we were able to derive the following solutions for α. If $\frac{|G|}{\sqrt{E^2+F^2}} \leq 1$ then in

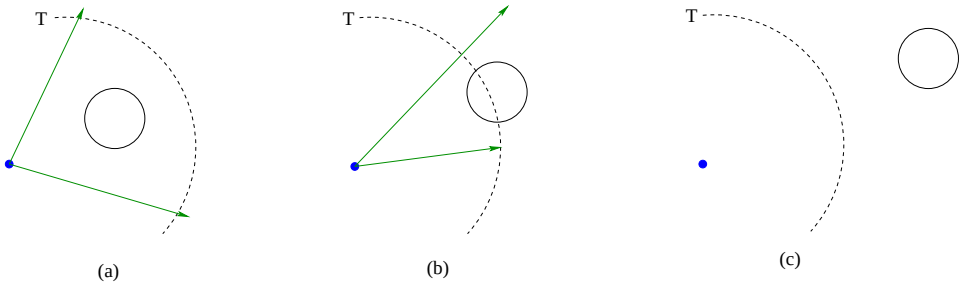

Figure 3: Relationship of Encounter Geometry and Lookahead Time

some 2π range, we have

$$\alpha_1 = \texttt{asin}\left(\frac{G}{\sqrt{E^2+F^2}}\right) - \texttt{atan}(E,F),$$
$$\alpha_2 = \pi - \texttt{asin}\left(\frac{G}{\sqrt{E^2+F^2}}\right) - \texttt{atan}(E,F),$$

where

$$E = \omega(R\varepsilon s_x - s_y),\ \ F = -\omega(R\varepsilon s_y + s_x),\ \ G = \mathbf{v}_i \cdot (R\varepsilon\,\mathbf{s}^{\perp} - \mathbf{s}),$$

Since E, F, and G are all functions of ε, we have two pairs of α_1 and α_2 or a total of four total angles. These angles represent the angles where the track prevention band changes color, assuming no lookahead time. In many cases, we have fewer unique angles, for instance when $\frac{|G|}{\sqrt{E^2+F^2}} > 1$ or when the regions are adjacent (i.e., two α's are equal). This result was formalized in the PVS theorem prover and we have written a Java implementation of the algorithm.

4.2 Solution with Lookahead Time

The solution presented so far only considers the 2-dimensional case with no lookahead time. Figure 3 illustrates three distinct cases that appear when lookahead is considered. Cases (a) and (c) were easy to handle, but we realized case (b) was going to take some additional analysis. But we were quite pleased with our new result and decided to present the result to our branch head and research director. During the presentation a member of the team announced, "I think you can solve this problem without trigonometry," and he urged us to defer the use of trigonometry until the last possible moment. In other words, he suggested that we solve for (v_α) without expanding its components. Only after the appropriate abstract solution vector is found, should the conversion to a track α be made. This was a key idea that had been used in the development of the KB3D algorithms, which resulted in very efficient and elegant algebraic solutions [1]. Indeed, we realized that the infinite lookahead problem was solvable by a particular kind of KB3D resolutions called *track lines* and, for the case of track solutions, computed by the function presented in Figure 4.

The function `trk_line` returns the vector $\mathbf{0}$ when all track angles for the ownship yield a potential conflict. Otherwise, the vector returned by this function is a velocity vector for the ownship that is tangent to the 2-dimensional protected zone. Since $\varepsilon$and $\iota$are ± 1, for given $\mathbf{s}$, $\mathbf{v}_o$, and $\mathbf{v}_i$ there are four possible track line solutions.

The key to solve track bands with finite lookahead is to find where the *projected* lookahead time intersects the protected zone. That is, plot where the relative position of the aircraft will be after T time units in every possible direction given an unchanged ground speed. And find the intersection points with

```
trk_line(s,v_o,v_i,ε,ι) : Vect2 =
  LET u = tangent_line(s,ε),
      a = u^2,
      b = u·v_i,
      c = v_i^2 - v_o^2 IN
  IF discriminant(a,2*b,c) ≥ 0 THEN
    LET k = root(a,2*b,c,ι) IN
    IF k ≥ 0 THEN
      ku+v_i
    ELSE
      0
    ENDIF
  ELSE
    0
  ENDIF
```

Figure 4: Track Line Solutions

the protection zone. The function `trk_circle`, also available in KB3D, provides these solutions, which are called *track circles*.

The function `trk_circle` returns the vector **0** when there are no track circle solutions, i.e., when the lookahead time boundary t and the protected zone do not intersect, or when there are an infinite number of solutions. Otherwise, the vector returned by this function is a velocity vector for the ownship that intersect the 2-dimensional protected zone at a time later than t. Since $\varepsilon$and $\iota$are ± 1, for given $\mathbf{s}$, $\mathbf{v}_o$, and $\mathbf{v}_i$ there are four possible track circle solutions. The `trk_line` and `trk_circle` functions and all of their subfunctions are fully developed and defined in [8].

We believe that the finite lookahead problem would have been intractable in the trigonometric approach pursued at first. This switch to a pure algebraic approach was fundamental to achieving the final 3D proof of the bands algorithm.

4.3 The Track Bands Algorithm

The idea of the algorithm is to first find the critical track vectors using our `track_line` and `track_circle` functions. These critical vectors are

$$\begin{aligned}
\mathbf{R}_{mm} &= \texttt{track_line}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, -1, -1),\\
\mathbf{R}_{mp} &= \texttt{track_line}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, -1, +1),\\
\mathbf{R}_{pm} &= \texttt{track_line}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, +1, -1),\\
\mathbf{R}_{pp} &= \texttt{track_line}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, +1, +1),\\
\mathbf{C}_{rm} &= \texttt{track_circle}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, T_{red}, -1),\\
\mathbf{C}_{rp} &= \texttt{track_circle}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, T_{red}, +1),\\
\mathbf{C}_{am} &= \texttt{track_circle}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, T_{amber}, -1),\\
\mathbf{C}_{ap} &= \texttt{track_circle}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, T_{amber}, +1),\\
\mathbf{C}_{em} &= \texttt{track_circle}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, t_{entry}, -1),\\
\mathbf{C}_{ep} &= \texttt{track_circle}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, t_{entry}, +1),\\
\mathbf{C}_{xm} &= \texttt{track_circle}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, t_{exit}, -1),\\
\mathbf{C}_{xp} &= \texttt{track_circle}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, t_{exit}, +1).
\end{aligned}$$

Some of these vectors may be zero vectors in which case they are ignored. Next, find the corresponding track angles (using `atan`) and sort this list of angles. To provide appropriate bounding, the angles 0 and 2π are added. Finally, use the conflict probe (such as CD3D) at an angle between each of the critical angles to characterize the whole region (i.e., determine which color the region should be painted: green, amber, or red). This procedure is iterated between the ownship and all traffic aircraft. Finally, the resulting bands are merged to get the display in Figure 1.

5 Formal Verification of Pairwise Prevention Bands Algorithms

The algorithms `trk_line` and `trk_circle` discussed in Section 4.2 have been verified *correct* for conflict resolution, i.e., they compute vectors that yield conflict free trajectories for the ownship, and *complete* for track prevention bands, i.e., they compute all critical vectors were the track bands change colors. These algorithms are slightly different from the original ones presented in [8]. Indeed, the algorithms presented in that report, while still correct for conflict resolution, failed to compute all the critical vectors. Therefore, those algorithms, which had been tested over 10,000 test cases, were incorrect. The missing vectors were found during the formal verification process.

The general idea of the correctness proof of the prevention bands algorithms is simple. For a given parameter of the ownship, e.g., track angle, we define a function $\Omega_{\text{trk}} : \mathbb{R} \to \mathbb{R}$, parametrized by $\mathbf{s}$, $\mathbf{v}_o$, and $\mathbf{v}_o$, that characterizes conflicts in the following way: $\Omega_{\text{trk}}(\alpha) < 0$ if and only if $\texttt{conflict?}(\mathbf{s}, \mathbf{v}_\alpha)$, where $\mathbf{v}_\alpha$ is defined as in Formula 5.

Then, we prove that the critical vectors computed in Section 4.3 are *complete*, i.e., they are exactly the zeros of the function Ω_{trk}. Next, we prove that the function Ω_{trk} is continuous. Therefore, by the Intermediate Value theorem, we can deduce that any point in an open bands region, e.g., the mid-point, determines the color of the whole band. This last step requires the existence of a conflict probe algorithm that is correct and complete, which we have already developed and verified.

Since the functions that compute the critical vectors for ground speed are much simpler that for track angle, we decided to start the formalization of that proof sketch with the case of 2-dimensional ground speed bands with no lookahead time. In this case, the function Ω_{gs} that we needed to construct had to

characterize `conflict?`$(\mathbf{s},\mathbf{v}_k)$, where

$$\mathbf{v}_k = \frac{k}{\mathrm{gs}(\mathbf{v})}\mathbf{v}. \tag{9}$$

The following formula provided the needed relationship between horizontal conflict that does not include a quantification over time:

$$\texttt{horizontal\_conflict?}(\mathbf{s},\mathbf{v}) \iff \mathbf{s}\cdot\mathbf{v} < R\,\texttt{det}(\mathbf{v},\mathbf{s}) < -\mathbf{s}\cdot\mathbf{v}, \tag{10}$$

where, R is defined as in Formula 8.

The function Ω_{gs} was constructed based on this theorem. The resulting function required the use of if-then-else logic so the proof that it was continuous was tedious. After much effort, we were able to prove in PVS that the ground-speed algorithms `gs_line` and `gs_circle`, which are analogous to `trk_line` and `trk_circle`, were complete assuming no look-ahead time.

The lesson learned from this first attempt was that we needed a more abstract way of defining the functions Ω_{gs} and Ω_{trk} such that the complexity of the continuity proofs can be untangled from the subtleties of the track and ground speed resolutions. With this in mind, we define a function $\Omega\colon \mathbb{R}^n \to \mathbb{R}$, parametrized by $\mathbf{s}$, $\mathbf{v}_o$, and $\mathbf{v}_o$, such that $\Omega_{\mathrm{trk}} = \Omega \circ \mathbf{v}_\alpha$ and $\Omega_{\mathrm{gs}} = \Omega \circ \mathbf{v}_k$. The continuity of Ω_{trk} and Ω_{gs} is a consequence of the continuity of Ω, which can be proved once and for all for all kinds of bands, and the continuity of $\mathbf{v}_\alpha$ and $\mathbf{v}_k$. All this seems straightforward except that there are several technical difficulties.

The function Ω is closely related to the function that computes the minimum distance between two aircraft. That function is, in general, noncontinuous for an infinite lookahead time. Interestingly, it is continuous when a lookahead time is considered, but the general proof of this fact requires the use of vector variant of the Heine-Cantor Theorem, i.e., if M is a compact metric space, then every continuous function $f : M \to N$, where N is a metric space, is uniformly continuous, which was not available in the PVS NASA libraries. Furthermore, the minimum distance function may have flat areas. Therefore, special attention has to be paid to the definition of Ω to guarantee that the set of critical points is finite. Otherwise, it cannot be proven that the critical vector algorithms are complete.

The next sections discuss the formal verification of the prevention bands algorithms with lookahead time for both the 2-D and 3-D cases.

5.1 Verification of 2D Prevention Bands

In the 2-dimensional case, a direct definition of Ω is possible by using τ of minimal horizontal separation between two aircraft:

$$\tau(\mathbf{s},\mathbf{v}) = -\frac{\mathbf{s}\cdot\mathbf{v}}{\mathbf{v}^2}. \tag{11}$$

From τ, we can define Ω as follows:

$$\Omega(\mathbf{v}) = (\mathbf{s} + \min(\max(0,\tau(\mathbf{s},\mathbf{v})),T)\mathbf{v})^2 - D^2, \tag{12}$$

where $\mathbf{s}$ is the relative distance between the ownship and the intruder aircraft.

The use of square distances in Formula 12 avoids the use of the square root function. Since the minimum and maximum of a continuous function is continuous, the use min and max is easier to handle than the if-then-logic used our first attempt.

The function Ω is not defined when $\mathbf{v}$ is $\mathbf{0}$. Therefore, rather than using Ω directly, we use the

function $\mathbf{v} \mapsto \mathbf{v}^2\Omega(\mathbf{v})$, which is defined everywhere, and we prove that it is continuous and that it correctly characterizes conflicts, i.e., `conflict?`$(\mathbf{s},\mathbf{v})$ if and only if $\mathbf{v}^2\Omega(\mathbf{v}) < 0$.

The function $\mathbf{v}^2\Omega(\mathbf{v})$ has an infinite number of zeroes in some special cases, e.g., when $\mathbf{s}$ is at the border of the protected zone, i.e, when $\mathbf{s}^2 = D^2$. In those, special cases, we use an alternative characterization of conflicts that has the required properties. In August 2009, we completed the proof of the 2-dimensional track and ground speed bands with finite lookahead time. For additional technical details on this formal development, we refer the reader to [7].

5.2 Verification of 3D Prevention Bands

The verification of the 3D conflict prevention bands algorithm is similar to that of the 2D algorithm. Indeed, many of the geometrical concepts critical to the verification in the 2D case can be generalized to the 3D case. However, these generalizations are typically nontrivial. The reason for this is that, geometrically speaking, a circle (a 2D protected zone) is much easier to work with than a cylinder (a 3D protected zone). The Ω function used in the verification of the 2D algorithm uses the horizontal time of minimum separation τ, which is easy to compute analytically. In contrast, the fact that a cylinder is not a smooth surface indicates that a 3D generalization of the Ω function will not be as simply defined.

Despite these geometric challenges, a concept was discovered that can be used to simplify geometry problems involving distance on cylinders. This concept is the notion of a *normalized cylindrical length* [2]:

$$||\mathbf{u}||_{\text{cyl}} = \max\left(\frac{\sqrt{u_x^2+u_y^2}}{D}, \frac{|u_z|}{H}\right). \tag{13}$$

This metric nicely reduces horizontal and vertical loss of separation into a single value. Indeed, if $\mathbf{s}$ is the relative position vector of two aircraft, then $||\mathbf{s}||_{\text{cyl}} < 1$ if and only if the aircraft are in 3D loss of separation.

Using the cylindrical distance metric, the Ω function can be defined in the 3D case as follows.

$$\Omega_{3D}(\mathbf{v}) = \min_{t\in[0,T]} ||\mathbf{s}+t\cdot\mathbf{v}||_{cyl} - 1, \tag{14}$$

where S is the relative distance between the ownship and intruder aircraft. An immediate consequence of this definition is that two aircraft are in conflict if and only if $\Omega_{3D}(\mathbf{v}) < 0$.

The correctness of the prevention bands algorithms relies on the fact that Ω_{3D} is a continuous function of $\mathbf{v}$, that the set of critical vectors, i.e., the zeroes of the function is finite, and that the critical vector algorithms are complete.

For many functions, a proof of continuity follows immediately from definitions. In this case, function Ω_{3D} is a minimum over the closed interval $[0,T]$. While standard methods from differentiable calculus are often employed in similar problems, this function is a minimum of a non-differentiable function, namely the cylindrical length. Its closed form involves several if-else statements and it would be difficult to use directly in a proof of continuity. Thus, somewhat more abstract results from real analysis were needed to be extended to vector analysis, e.g., the notion of limits, continuity, compactness, and finally the Heine-Cantor Theorem.

As in the 2-dimensional case, the function Ω_{3D} may have flat areas and, consequently, in some special cases, may have an infinite number of critical zeros. We carefully identified these special cases and then used an alternative definition of Ω_{3D}. These special cases are extremely rare, indeed all the missing critical vectors in the original algorithms presented in [8] were due to these special cases. Although they are rare, dealing with them is necessary for the correctness of the algorithms. If one critical vector is

missing, the coloring of the bands will be potentially switch from red to green.

Finally, the PVS proof that the algorithms find all of the critical points is less abstract but more tedious than the proof of continuity. It required the development of several PVS theories on the Ω_{3D} function, which are general enough to be used in other state-based separation assurance algorithms. The proof of the correctness of the 3-dimensional algorithms for track, ground speed, and vertical speed with finite lookahead time was complete in December 2009.

6 Verification of the Merge Algorithm

Soon after the extension from purely geometric solutions to solutions that incorporate a lookahead time, we realized that standard set operations (set union, set difference, etc.) could be used to implement both the lookahead time and the merging of bands from a pairwise solution to a 1-to-n analysis.

Suppose we had a way to determine the set track angles that have a loss of separation within time T, denoted $\mathscr{G}_{<T}$. Then since $T_{red} < T_{amber}$, we may define the colored bands of track angles in terms of this new set:

$$\begin{aligned} \mathscr{G}_{red} &= \mathscr{G}_{<T_{red}} \\ \mathscr{G}_{amber} &= \mathscr{G}_{<T_{amber}} - \mathscr{G}_{<T_{red}} \\ \mathscr{G}_{green} &= \{\alpha \,|\, 0^\circ \leq \alpha < 360^\circ\} - \mathscr{G}_{<T_{amber}} \end{aligned}$$

This observation simplifies the analysis, because now we only need to analyze one set, $\mathscr{G}_{<T}$. This operation only uses the set difference operation.

Next we observed that each aircraft's contribution to the set $\mathscr{G}_{<T}$ is independent of all other traffic; thus, the problem neatly divides into a series of aircraft pairs: the ownship and each traffic aircraft. If we use $\mathscr{G}^{o,i}_{<T}$ to represent the set of track angles which cause a loss of separation within time T between traffic aircraft i and the ownship o, then the set of track angles for all traffic is then be formed by

$$\mathscr{G}_{<T} = \bigcup_{i \in traffic} \mathscr{G}^{o,i}_{<T}$$

This observation simplifies the analysis again, because now we only need to find the track angles which cause a conflict between two aircraft, denoted by the set $\mathscr{G}^{o,i}_{<T}$. Before we examine the set $\mathscr{G}^{o,i}_{<T}$ in detail, we introduce some mathematical modeling concepts.

At this point we realized that our solution would rely on a Java or C++ implementation of sets of floating point numbers with their associated set operations. Common implement ions of sets in programming languages do not include efficient ways to deal with ranges of floating point number; therefore, we chose to implement our own. However, we realized that these algorithms would be non-trivial and would require verification. Thus we performed a code-level verification of the algorithm to merge and subtract bands.

Each band is represented by an interval describing its minimal and maximal values, with the set of all bands of one color being an interval set. These interval sets were internally represented by arrays of (ordered) intervals. Necessary properties for the implementation would be that the data structures representing the bands both remained ordered and preserved the proper value ranges within a set of bands.

For multiple aircraft, red bands for each ownship/intruder pair are merged together and the ownship green bands are calculated as the difference of the red bands from a single all-inclusive band. Merging combines overlapping bands as appropriate, with subtraction breaking larger bands into smaller ones. The main complications in the proof resulted from boundary conditions and an ordering calculation

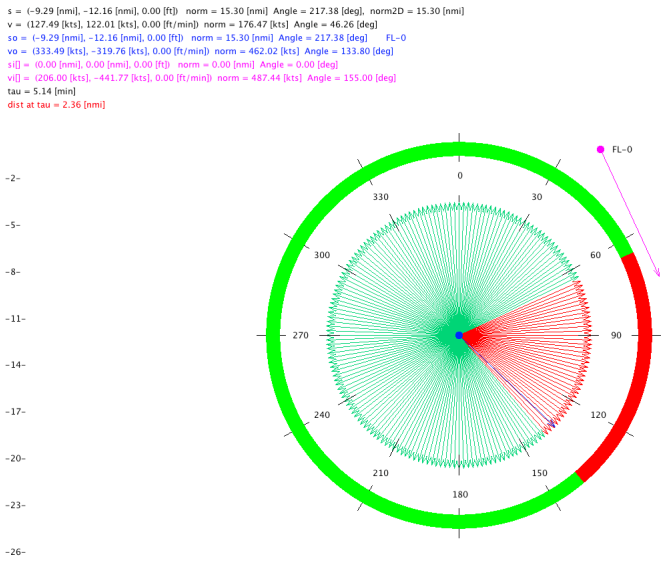

Figure 5: Screenshot of Track Prevention Band Display

where zero or a positive value represents an actual position in the array of intervals, but a negative value represents a point between (or beyond) the intervals currently in the set.

One aspect the formal verification brought to the fore was the interpretation of intervals. It was not possible to exclusively use closed or open intervals for both merging and difference operations. Removing a closed interval, which includes the endpoints, leaves us with open intervals — everything up to, but not including, these end points, and possible inconsistencies if we mixed the two. Removing two adjacent open intervals could also lead to a left over point between them.

In the interest of having a consistent interface and eliminating redundant code, we decided to have the results of both merge and difference be closed intervals. The inputs for union would likewise be closed, and also the original interval set for difference. The interval set to be subtracted, however, would be interpreted as open intervals. As mentioned above, this did lead to the possibility of introducing artifacts of *singleton* intervals, where both endpoints have the same value. After consideration, however, we realized these could be safely eliminated, as they are equivalent to a critical point at a local minimum or maximum. Green singletons could be eliminated without introducing additional danger, and a red singleton would represent a brush against (but not a cross into) the intruder's protected zone.

7 Java and C++ Implementations

Ultimately these algorithms will be brought into large simulation environments where they will be evaluated for performance benefits (improvements to airspace capacity or aircraft efficiency). Some of these simulation environments are in Java and some are in C++. Therefore a requirement of this project is not only to develop an algorithm and verify it, but to also produce Java and C++ implementations of the algorithm. The initial Java version of the algorithm was available in December 2008, see Figure 5. This version successfully passed the limited test suite we developed. By the summer of 2009 we had a C++ version and the testing apparatus to verify exact agreement between the Java and C++ versions, along with a regression suite of 100 test scenarios.

Since PVS is a specification language and contains non-computable functions, we deliberately restricted our use of PVS to only the computable elements. In this way there is a direct translation of

PVS into Java or C++. We are currently developing a tool to automatically convert PVS specifications into Java [5], but the tool is not mature enough to handle the specification of these kinds of algorithms in PVS. Even by converting by hand, we ran into certain problems. PVS libraries contain all the appropriate vector operations (addition, dot-product, etc.). These libraries do not exist in standard Java or C++. First we pursued the approach of finding a third-party library to offer these functions. However, we found certain quirks in their implementation. One vector library took full advantage of the imperative nature of the Java language, implementing functions on vectors which would change the parameters to the vector. This results in efficient code, because object creation is not necessary, but does not closely relate to the functional style of PVS. Because of these incompatibilities, we chose to implement our own vector libraries. In a similar way, we developed our own set operations (union and intersection).

However even with this hand translation, we still do not have an behavioral replica of the PVS in Java or C++. The most glaring difference is that Java and C++ use floating point numbers and PVS uses actual real numbers. All of our verifications in PVS are accomplished with vectors defined over the real numbers. This can be thought of as computation using infinite precision arithmetic. Clearly, our Java and C++ implementations execute on less powerful machines than this. There are several places where we must be especially careful:

- Calculation of quadratic discriminants. Since we are often computing tangents, the theoretical value is zero, but the floating point answer can easily be small negative number near zero. We would then miss a critical point.
- The possibility of the mid-point of a region being very close to zero.

Finally, another aspect related to this issue is that the data input into the algorithm is not precise. The general rule-of-thumb is that the error in the input data will overwhelm any error introduced by floating point computations. However, we would like to make a formal statement that includes both data and computational errors.

8 Conclusions

In this paper, we have presented a short history of the development and formal verification of prevention bands algorithms. The resulting track-angle, ground speed, and vertical speed bands algorithms are far more simple than our earlier versions. The goal of completing a formal proof forced us to search for simplifications in the algorithms and in the underlying mathematical theories. A key insight that enabled the completion of this work, is that trigonometric analysis should be deferred until the latest possible time. Although, the project took far longer than we expected, we are very pleased with the elegance and efficiencies of the discovered algorithms.

References

[1] G. Dowek, A. Geser, and C. Muñoz. Tactical conflict detection and resolution in a 3-D airspace. In *Proceedings of the 4th USA/Europe Air Traffic Management R&DSeminar, ATM 2001*, Santa Fe, New Mexico, 2001. A long version appears as report NASA/CR-2001-210853 ICASE Report No. 2001-7.

[2] Gilles Dowek and C. Muñoz. Conflict detection and resolution for 1,2,...,N aircraft. In *Proceedings of the 7th AIAA Aviation, Technology, Integration, and Operations Conference, AIAA-2007-7737*, Belfast, Northern Ireland, 2007.

[3] J. Hoekstra, R. Ruigrok, R. van Gent, J. Visser, B. Gijsbers, M. Valenti, W. Heesbeen, B. Hilburn, J. Groeneweg, and F. Bussink. Overview of NLR free flight project 1997-1999. Technical Report NLR-CR-2000-227, National Aerospace Laboratory (NLR), May 2000.

[4] J. M. Hoekstra. Designing for safety: The free flight air traffic management concept. Technical Report 90-806343-2-8, Technische Universiteir Delft, November 2001.

[5] Leonard Lensink, César Muñoz, and Alwyn Goodloe. From verified models to verifiable code. Technical Memorandum NASA/TM-2009-215943, NASA, Langley Research Center, Hampton VA 23681-2199, USA, June 2009.

[6] Jeffrey Maddalon, Ricky Butler, Alfons Geser, and César Muñoz. Formal verification of a conflict resolution and recovery algorithm. Technical Report NASA/TP-2004-213015, NASA/Langley Research Center, Hampton VA 23681-2199, USA, April 2004.

[7] Jeffrey Maddalon, Ricky Butler, César Muñoz, and Gilles Dowek. A mathematical analysis of conflict prevention information. In *Proceedings of the AIAA 9th Aviation, Technology, Integration, and Operations Conference, AIAA-2009-6907*, Hilton Head, South Carolina, USA, September 2009.

[8] Jeffrey Maddalon, Ricky Butler, César Muñoz, and Gilles Dowek. A mathematical basis for the safety analysis of conflict prevention algorithms. Technical Report TM-2009-215768, NASA Langley, June 2009.

[9] S. Owre, J. Rushby, and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *Proc. 11th Int. Conf. on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752. Springer-Verlag, June 1992.