Provably Correct Conflict Prevention Bands Algorithms

Anthony Narkawicz[a], César Muñoz[a], Gilles Dowek[b]

[a] *NASA Langley Research Center, Hampton, VA*
[b] *École Polytechnique, France*

Abstract

In air traffic management, a pairwise conflict is a predicted loss of separation between two aircraft, referred to as the ownship and the intruder. A conflict prevention bands system displays ranges of maneuvers for the ownship that characterize regions in the airspace that are either conflict-free or "don't go" zones that the ownship has to avoid. Errors in the calculation of prevention bands may result in incorrect separation assurance information being displayed to pilots or air traffic controllers. Algorithms that compute conflict prevention bands are surprisingly difficult to formalize and verify. This paper presents a method for the analysis and verification of prevention bands algorithms. The method, which has been implemented in the Prototype Verification System (PVS), is illustrated with a provably correct 3-dimensional prevention bands algorithm for track angle maneuvers.

Keywords: formal verification, theorem proving, air traffic management

1. Introduction

The next generation of air traffic management systems may enable modes of operations where aircraft take a primary responsibility in the management of air traffic separation. These modes of operations are supported by advances in hardware and software technologies. For example, global navigation satellite systems, such as Global Positioning System (GPS), provide accurate surveillance information, which is then broadcast to traffic aircraft and ground elements by systems such as Automatic Dependent Surveillance-Broadcast (ADS-B). This information is then used by separation assurance system to advise aircraft crew and air traffic controllers about air traffic conflicts.

In air traffic management, a *(pairwise) conflict* is a predicted loss of separation between two aircraft within a lookahead time. One of the aircraft

is called the *ownship* and the other aircraft, which represents an arbitrary traffic aircraft, is called the *intruder*. A *conflict prevention* system consists of algorithms that sense traffic aircraft and characterize ranges of maneuvers for the ownship that are either conflict-free or that lead to conflict. The maneuvers are typically constrained to those where only one parameter of the ownship's velocity is varied at a time. Examples of such maneuvers are those that modify either the track angle, vertical speed, or ground speed of the aircraft.

A *(pairwise) prevention bands* algorithm, for a given parameter such as track angle, has as input the state information of the ownship and intruder aircraft, i.e., their 3-dimensional position and velocity vectors, and returns a list of regions, called *bands*, consisting of values for the specified parameter. There is a natural way to associate a color, either red or green, to each band. *Red bands* specify "don't go" zones, i.e., parameter values that the ownship has to avoid because they lead to conflict. Conversely, the *green bands* specify parameter values for the ownship that yield conflict-free maneuvers.

A pairwise prevention bands algorithm is *correct* if every possible value for the chosen parameter is either contained in a band or is a boundary point of one of the bands, and if the colors of the bands characterize conflict as follows. For all bands B and parameter values $x \in B$, the ownship's maneuver corresponding to the value x is in conflict with the intruder aircraft if and only if the color of B is red. Equivalently, the ownship's maneuver corresponding to x is not in conflict if and only if the color of B is green.

There are serious safety implications if a preventions band algorithm is incorrect, since a pilot may assume that certain maneuvers are safe when they are not. Thus, formal verification of such algorithms ensures reliability and hence safety of the airspace system. Surprisingly, mathematically precise conflict prevention bands algorithms are difficult to analyse and verify [1]. The formal verification of a prevention bands algorithm for horizontal conflicts was described in [2]. Three-dimensional prevention bands algorithms were presented, without correctness proofs, in [3]. The 3-dimensional algorithms presented in that paper compute incorrect bands for some special cases. Hoekstra [4] graphically describes some algorithms developed in the National Aerospace Laboratory (NLR) in the Netherlands [5], but he does not provide much detail about how the algorithms actually work.

If a verifiable conflict detection algorithm is available, an iterative approximation of preventions bands is possible. For instance, approximate colored bands for track angle maneuvers can be computed by varying the ownship's track angle by some small value and checking, using the conflict detection probe, whether the angle variations result in a conflict or not.

However, such an iterative approach would consume more computational resources than an analytical one where the edges of the bands are computed directly. Furthermore, an iterative approach may not scale well where such separation assurance algorithms must be executed for many different traffic aircraft every second.

This paper presents a method for the analysis and verification of prevention bands algorithms. The method is illustrated with a corrected version of a 3-dimensional prevention bands algorithm for track angle maneuvers originally proposed in [3]. Corrected versions of 3-dimensional ground speed and vertical speed prevention bands algorithms have been also developed and are presented in a companion technical report [6].

The mathematical development presented in this paper has been specified and formally verified in the Prototype Verification System (PVS) [7]. PVS is a proof assistant that consists of a specification language, based on classical higher-order logic, and a mechanical theorem prover for this logic. The PVS specification language allows for the precise definition of mathematical objects such as *functions* and *relations*, and the precise statement of logical formulas such as *lemmas* and *theorems*. Proofs of logical formulas can be mechanically checked using the PVS theorem prover, which guarantees that every proof step is correct and that all possible cases of a proof are covered. All lemmas and theorems presented in this paper have been mechanically checked in PVS. For the sake of simplicity, only proof sketches of the main results are presented in the paper. A self-contained development that includes definitions and formal proofs and all required libraries is available in a compressed file at `http://shemesh.larc.nasa.gov/people/cam/ACCoRD/PVS-Feb-23-10.tgz`. Once this file is uncompressed, a README file provides instructions for rebuilding the development using a standard version of PVS 4.2 (`http://http://pvs.csl.sri.com/download.shtml`). A summary of that development is presented in the appendix of this paper.

Notation

The use of a formal language, e.g., in this case the specification language of PVS, enforces rigorous definitions of mathematical objects, where all dependencies are clearly specified. This level of rigor guarantees a very high confidence on the correctness of the results presented in this paper. However, this also makes the notation heavy and difficult to read for the non-expert reader. For this reason, the work presented here uses standard mathematical notation and does not assume that the reader is familiar with the syntax or semantics of the PVS language. In particular, the following conventions are

used by the authors to make this development more accessible to the casual reader:

- The PVS specification language is *strongly typed*, i.e., all declarations are explicitly typed. This feature guarantees that all PVS functions are total and well-defined. For instance, a mathematical formula that includes a division needs to make explicit the fact that the divisor is different from zero, otherwise the expression would be undefined. In PVS, these conditions are handled by a type system, which is enforced by the PVS type-checker. Since PVS type annotations tend to be verbose, formulas in this paper appear untyped. When necessary, the type domain of variables is made explicit in the context where the formula appears.

- PVS is based on higher-order logic, so it supports the definition of functions that return functions or that have functions as arguments. In this paper, arguments of a higher-order function are called *parameters* and those parameters are implicitly defined in the text. For example, a function $f : \mathbb{R} \mapsto \mathbb{R}$ with implicit parameters $\mathbf{s}$ and $\mathbf{v}$ is indeed defined in PVS as a higher-order function f that given $\mathbf{s}$ and $\mathbf{v}$ returns a function of type $\mathbb{R} \mapsto \mathbb{R}$.

- The PVS notation is declarative, i.e., there is not a notion of memory state as in imperative programming languages. In this paper, algorithms are represented by functions. By convention, names of functions that are intended to have a logical meaning are written in *italics*. Functions that represent algorithms to be implemented in a programming language are written in `typewriter` font.

The following mathematical notations are used in this paper. Vector variables are written in **boldface** and can denoted by their components. For example, if $\mathbf{w} \in \mathbb{R}^3$ and $\mathbf{u} \in \mathbb{R}^2$, then $\mathbf{w} = (\mathbf{w}_x, \mathbf{w}_y, \mathbf{w}_z)$ and $\mathbf{u} = (\mathbf{u}_x, \mathbf{u}_y)$. The notation $\mathbf{w}_{(x,y)}$ denotes the projection of $\mathbf{w}$ in the horizontal plane, i.e.,[1]

$$\mathbf{w}_{(x,y)} \equiv (\mathbf{w}_x, \mathbf{w}_y),$$

and the notation $\mathbf{u}$ `with` $[z \leftarrow r]$ denotes the 3-dimensional vector whose projection to $\mathbb{R}^2$ is $\mathbf{u}$ and whose z-coefficient is $r \in \mathbb{R}$, i.e.,

$$\mathbf{u} \texttt{ with } [z \leftarrow r] \equiv (\mathbf{u}_x, \mathbf{u}_y, r).$$

[1]The symbol $\equiv$ is used in this paper to introduce mathematical definitions.

The notation $\|\mathbf{w}\|$ refers to the norm of the vector $\mathbf{w}$ and the notation $\mathbf{w} \cdot \mathbf{w}'$ refers to the dot product of the vectors $\mathbf{w}$ and $\mathbf{w}'$. The expression $\mathbf{0}$ represents the zero vector, e.g., the vector whose components are 0.

If $\mathbf{u} \in \mathbb{R}^2$, then $\mathbf{u}^\perp$ denotes the (right) perpendicular vector:

$$\mathbf{u}^\perp \equiv (\mathbf{u}_y, -\mathbf{u}_x).$$

The function sign: $\mathbb{R} \mapsto \{-1, 1\}$ is defined such that $\text{sign}(x) = 1$ if $x \geq 0$ and $\text{sign}(x) = -1$ otherwise. The expression $\iota = \pm 1$ denotes the fact that an integer ι belongs to the set $\{-1, 1\}$. Moreover, $\neg$, $\implies$, $\iff$ denote logical negation, implication, and equivalence, respectively.

2. Statement of the Problem

The prevention bands algorithms discussed here only use the state-based information of the ownship and intruder aircraft, i.e., constant position and velocity vectors that are elements of the 3-dimensional Euclidean space $\mathbb{R}^3$. Aircraft dynamics are represented by a point moving at constant linear speed. The current position and velocity vectors of the ownship are denoted $\mathbf{s}_o$ and $\mathbf{v}_o$, respectively, while the vectors $\mathbf{s}_i$ and $\mathbf{v}_i$ denote the current state of the intruder aircraft/

In the airspace system, the separation requirement for two aircraft is specified as a minimum horizontal separation D and a minimum vertical separation H. A conflict between the ownship and the intruder occurs when there is a time in the future, within a lookahead time T, such that the horizontal distance between the aircraft is less than D, and the vertical distance is less than H. Typically, D is 5 nautical miles, H is 1000 feet, and T is 5 minutes.

For the remainder of the paper, it is assumed that the ground speeds of the ownship and intruder aircraft are not zero, i.e., $\|\mathbf{v}_{o(x,y)}\| \neq 0$ and $\|\mathbf{v}_{i(x,y)}\| \neq 0$, and that the aircraft are not in loss of separation, i.e., either $\|\mathbf{s}_{o(x,y)} - \mathbf{s}_{i(x,y)}\| \geq D$ or $|\mathbf{s}_{oz} - \mathbf{s}_{iz}| \geq H$. Therefore, $\mathbf{v}_{o(x,y)} \neq \mathbf{0}$, $\mathbf{v}_{i(x,y)} \neq \mathbf{0}$, and $\mathbf{s}_o - \mathbf{s}_i \neq \mathbf{0}$.

2.1. Conflicts

The ownship and the intruder aircraft are in *conflict* if there exists $t \in [0, T]$ such that, at time t, vertical separation is lost, i.e,

$$|((\mathbf{s}_o + t\,\mathbf{v}_o) - (\mathbf{s}_i + t\,\mathbf{v}_i))_z| < H,$$

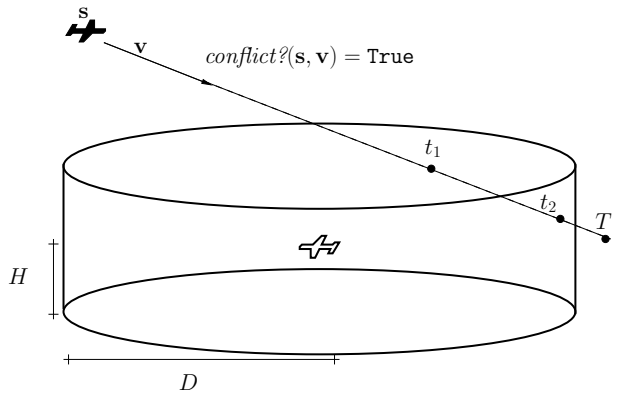

Figure 1: Conflict scenario

and horizontal separation is lost, i.e.,

$$\|(\mathbf{s}_o + t\,\mathbf{v}_o)_{(x,y)} - (\mathbf{s}_i + t\,\mathbf{v}_i)_{(x,y)}\| < D.$$

Since $(\mathbf{s}_o + t\,\mathbf{v}_o) - (\mathbf{s}_i + t\,\mathbf{v}_i) = (\mathbf{s}_o - \mathbf{s}_i) + t\,(\mathbf{v}_o - \mathbf{v}_i)$, the predicate that characterizes conflict can be defined on $\mathbf{s} = \mathbf{s}_o - \mathbf{s}_i$ and $\mathbf{v} = \mathbf{v}_o - \mathbf{v}_i$, i.e., the relative position and velocity vector, respectively, of the ownship with respect to the intruder. By notational convenience, conflict is defined by a predicate of the two relative vectors $\mathbf{s}$ and $\mathbf{v}$ rather than a predicate of four vectors $\mathbf{s}_o$, $\mathbf{v}_o$, $\mathbf{s}_i$, and $\mathbf{v}_i$.

$$\begin{aligned} \textit{conflict?}(\mathbf{s}, \mathbf{v}) \equiv \exists t \in [0, T] : |(\mathbf{s} + t\,\mathbf{v})_z| < H \text{ and} \\ \|\mathbf{s}_{(x,y)} + t\,\mathbf{v}_{(x,y)}\| < D. \end{aligned} \tag{1}$$

For the remainder of this paper, the relative position and velocity vectors, $\mathbf{s}$ and $\mathbf{v}$, will be used in place of $\mathbf{s}_o - \mathbf{s}_i$ and $\mathbf{v}_o - \mathbf{v}_i$, respectively.

The separation requirement can be understood as an imaginary horizontal cylinder, called *protected zone*, of height $2H$ and radius D around the intruder aircraft. A conflict between the ownship and the intruder aircraft occurs when there exists a time $t \in [0, T]$ at which the ownship is in the interior of the intruder's protected zone. Figure 1 illustrates the protected zone around the intruder aircraft and a conflict scenario with a loss of separation during the time interval (t_1, t_2).

2.2. Ownship Maneuvers

A *maneuver* is a new velocity vector for the ownship that is assumed to be implemented in zero time. The set of maneuvers that are relevant to

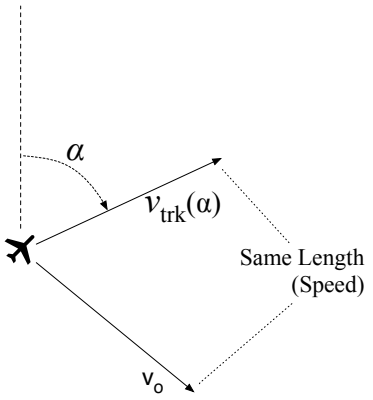

Figure 2: Track angle maneuver for the ownship

prevention bands are those generated by a function $\nu : \mathbb{R} \mapsto \mathbb{R}^3$, implicitly parametrized by $\mathbf{v}_o$, that given a value r returns a new velocity vector $\nu(r)$. For instance, track angle maneuvers are characterized by the function ν_{trk}, which is defined as follows:

$$\nu_{\text{trk}}(\alpha) \equiv (\|\mathbf{v}_{o(x,y)}\| \sin\alpha, \|\mathbf{v}_{o(x,y)}\| \cos\alpha, \mathbf{v}_{oz}), \tag{2}$$

where $\alpha \in \mathbb{R}$ is a track angle. In this case, there exists a function $\texttt{track}\colon \mathbb{R}^3 \mapsto \mathbb{R}$ that satisfies

$$\texttt{track}(\nu_{\text{trk}}(\alpha)) = \alpha. \tag{3}$$

Track angle maneuvers satisfy the following properties:

$$\|\nu_{\text{trk}}(\alpha)_{(x,y)}\| = \|\mathbf{v}_{o(x,y)}\| \tag{4}$$

$$\nu_{\text{trk}}(\alpha)_z = \mathbf{v}_{oz} \tag{5}$$

The track angle maneuver for the ownship that is given by $\nu_{\text{trk}}(\alpha)$ is illustrated by Figure 2.

Other functions $\nu\colon \mathbb{R} \to \mathbb{R}^3$, such as those that characterize ground speed maneuvers and vertical speed maneuvers, can be similarly defined. For such a function ν, an argument x of ν can be viewed as a parameter of the ownship's velocity, and $\nu(x)$ as the corresponding velocity vector.

2.3. Conflict Detection Algorithms

A *conflict detection* algorithm `cd` is a function that takes as parameters the relative position `s` of the aircraft and the velocity vectors $\mathbf{v}_o$, $\mathbf{v}_i$, and returns a Boolean value, i.e., `True` or `False`.

Definition 1. *The algorithm* `cd` *is* correct *if it holds that*

$$\textit{conflict?}(\mathbf{s}, \mathbf{v}_o - \mathbf{v}_i) \implies \texttt{cd}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i).$$

The algorithm `cd` *is* complete *if it holds that*

$$\texttt{cd}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i) \implies \textit{conflict?}(\mathbf{s}, \mathbf{v}_o - \mathbf{v}_i).$$

In other words, a conflict detection algorithm is correct if it does not have missed alerts, i.e., it detects all conflicts, and it is complete if it does not have false alerts, i.e., it only detects actual conflicts. Note that a conflict detection algorithm that always returns `True` is correct but not complete and an algorithm that always returns `False` is complete but not correct. An example of a correct *and* complete conflict detection algorithm is `cd3d` (see Appendix in [3]).

2.4. Prevention Bands Algorithms

Given a function $\nu\colon \mathbb{R} \mapsto \mathbb{R}^3$ that is implicitly parameterized by $\mathbf{v}_o$ as above, and a closed interval I of real numbers, a *prevention bands algorithm* for ν over I is a function with parameters $\mathbf{s}$, $\mathbf{v}_o$, and $\mathbf{v}_i$ that returns a finite, ordered sequence L_ν of elements of I, such that the upper and lower bounds of I are in L_ν. The lower and upper bounds of the interval I are minimum and maximum values for the argument of ν. For $\nu = \nu_{\text{trk}}$, the closed interval I is defined as $[0, 2\pi]$. For $\nu = \nu_{\text{vs}}$, the lower and upper bounds of I are typically the minimum and maximum vertical speeds for the ownship, respectively.

Each consecutive pair A and B of entries in L_ν determines an open interval (A, B), which is called a *band* (for the parameter represented by ν). By abuse of notation, the syntax $(A, B) \in L_\nu$ will denote that (A, B) is a band in L_ν, i.e., A and B are consecutive entries in L_ν.

To each band (A, B) in L_ν, a Boolean value is associated as follows

$$\texttt{conflict\_band}(\mathbf{s}, \mathbf{v}_i, A, B) \equiv \texttt{cd}(\mathbf{s}, \nu(\frac{A+B}{2}), \mathbf{v}_i), \tag{6}$$

where `cd` is any correct conflict detection algorithm, such as `cd3d`. The algorithm `conflict_band` tests whether there is a conflict for the ownship maneuver that is given by evaluating ν on the midpoint of the interval (A, B).

Definition 2. *Given a function* $\nu\colon \mathbb{R} \mapsto \mathbb{R}^3$ *and a closed interval* $I \subseteq \mathbb{R}$*, a prevention bands algorithm for* ν *is correct if and only if for any band* (A, B) *in* L_ν*,*

$$\texttt{conflict\_band}(\mathbf{s}, \mathbf{v}_i, A, B) \iff \forall y \in (A, B) : \textit{conflict?}(\mathbf{s}, \nu(y) - \mathbf{v}_i) \tag{7}$$

The definition above states that all the points in a band computed by a correct prevention bands algorithm have the same conflict property, i.e., either all the points yield conflict-free maneuvers or all the points yield maneuvers that lead to conflict. It is important to note that for a correct preventions band algorithm, the midpoint in Equation (6) can be replaced by any other point in the band (A, B), since all the points have the same conflict property.

A preventions band algorithm L can be represented graphically by assigning a color, either red or green, to each band $(A, B) \in L_\nu$. The associated color is red if $\texttt{conflict\_band}(\mathbf{s}, \mathbf{v}_i, A, B) = \texttt{True}$, and it is green if $\texttt{conflict\_band}(\mathbf{s}, \mathbf{v}_i, A, B) = \texttt{False}$, then the corresponding color is green. This is illustrated in Figure 3 for the track angle case, i.e., $\nu = \nu_{\text{trk}}$ and $I = [0, 2\pi]$.

A prevention bands algorithm for track angle maneuvers will return a finite, ordered sequence $L_{\nu_{\text{trk}}}$ of track angles in the interval $[0, 2\pi]$. This sequence will contain both of the angles 0 and 2π. If the algorithm is correct, then each consecutive pair, α and β, of track angles in this sequence defines a band, i.e., an open interval (α, β), with the property that either

1. all track angles between α and β result in conflict, or
2. all track angles between α and β do not result in conflict.

If the track angles between α and β all result in conflict, the region between α and β is colored red. Otherwise, this region is colored green.

2.5. Proving Correctness of a Prevention Bands Algorithm

This section provides a general strategy that can be followed to formally verify that a given prevention bands algorithm is correct. Subsequent sections will describe the use of this strategy in the formal verification of a track angle prevention bands algorithm.

Recall that a prevention bands algorithm depends on a function $\nu \colon \mathbb{R} \mapsto \mathbb{R}^3$, implicitly parametrized by $\mathbf{v}_o$, and a closed interval $I \subseteq \mathbb{R}$. The real-valued argument of the function ν determines a new velocity vector for the ownship. For instance, if $\nu = \nu_{\text{trk}}$ and $I = [0, 2\pi]$, the domain of ν_{trk} are track angles such that for any $\alpha \in I$, $\nu_{\text{trk}}(\alpha)$ is a new velocity vector for the ownship that has the same ground speed as $\mathbf{v}_o$.

Theorem 1 can be used to verify the correctness of a prevention bands algorithm for ν over I that computes a finite sequence L_ν. It requires the construction of a particular continuous function, called classification function, and a completeness property on L_ν.

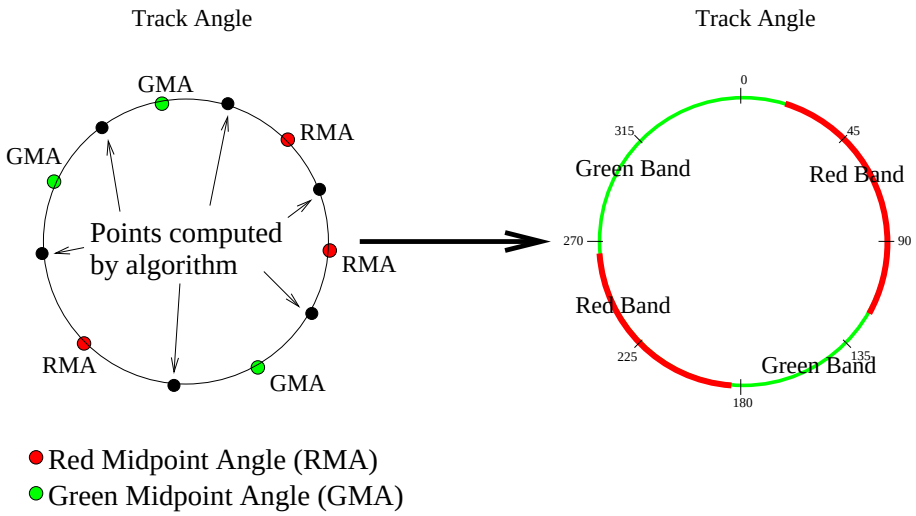

Figure 3: Relation between track angle prevention bands algorithm and graphical display

Definition 3. *A* classification function *function for* ν, $\Omega_\nu \colon \mathbb{R} \mapsto \mathbb{R}$, *is a continuous function, implicitly parameterized by* $\mathbf{s}$ *and* $\mathbf{v}_i$, *that characterizes conflict? in the following way:*

$$\Omega_\nu(x) < 1 \iff \textit{conflict?}(\mathbf{s}, \nu(x) - \mathbf{v}_i).$$

Definition 4. *Let* $\Omega_\nu \colon \mathbb{R} \mapsto \mathbb{R}$ *be a function, implicitly parameterized by* $\mathbf{s}$ *and* $\mathbf{v}_i$. *A finite sequence* L_ν *of real numbers in a closed interval* I *is* Ω_ν-complete *if for all* $x \in I$,

$$\Omega_\nu(x) = 1 \implies x \in L_\nu,$$

Theorem 1. *A prevention bands algorithm for* ν *over* I *that computes a finite sequence* L_ν *is correct if there exists a classification function* Ω_ν *such that* L_ν *is* Ω_ν*-complete.*

Proof. Assume that there exists a classification function Ω_ν such that L_ν is Ω_ν-complete. Let (A, B) be a band in L_ν.

- Assume that `conflict_band`$(\mathbf{s}, \mathbf{v}_i, A, B)$ holds. Let y be a real number in the open interval (A, B). Suppose, by reduction to absurdity, that $\neg \textit{conflict?}(\mathbf{s}, \nu(y) - \mathbf{v}_i)$. Since Ω_ν is a classification function, $\Omega_\nu(y) \geq 1$.

By hypothesis, L_ν is Ω_ν-complete. Thus, since $(A, B) \in L_\nu$ and y is equal to neither A nor B, it follows that $\Omega_\nu(y) > 1$. By the definition of the function `conflict_band` given in Equation (6), it holds that *conflict?*$(\mathbf{s}, \nu(x) - \mathbf{v}_i)$, where $x = \frac{A+B}{2}$. Since Ω_ν is a classification function, $\Omega_\nu(x) < 1$. By definition, Ω_ν is continuous. Thus, the intermediate value theorem implies that there exists some z between x and y such that $\Omega_\nu(z) = 1$. Since z is therefore in the interval (A, B), A and B are consecutive in L_ν, and the algorithm computes all points where Ω_ν realizes a value of 1, this is a contradiction.

- Similar reasoning is used to show that if $\neg$`conflict_band`$(\mathbf{s}, \mathbf{v}_i, A, B)$, then any y in (A, B) satisfies $\neg$*conflict?*$(\mathbf{s}, \nu(y) - \mathbf{v}_i)$.

□

Using Theorem 1 to verify that a prevention bands algorithm that computes L_ν is correct requires construction of a classification function Ω_ν such that L_ν is Ω_ν-complete. Section 3 proposes the definition of a generic function Ω that can be used to construct classification functions for a given function $\nu\colon \mathbb{R} \mapsto \mathbb{R}^3$. Section 4 presents a theorem that can be used to prove Ω_ν-completeness for a given sequence L_ν.

3. The Function Ω

Let $\Omega\colon \mathbb{R}^3 \mapsto \mathbb{R}^3$ be a continuous function, implicitly parametrized by $\mathbf{s}$ $(= \mathbf{s}_o - \mathbf{s}_i)$, that characterizes *conflict?* in the following way:

$$\Omega(\mathbf{v}) < 1 \iff \textit{conflict?}(\mathbf{s}, \mathbf{v}). \tag{8}$$

For any continuous function ν, a classification function Ω_ν can be constructed as follows.

$$\Omega_\nu(x) \equiv \Omega(\nu(x) - \mathbf{v}_i). \tag{9}$$

Given such a function Ω, the verification of correctness of a prevention bands algorithm over an interval I is reduced to proving that L_ν is Ω_ν-complete, i.e., the sequence returned by each algorithm contains all $x \in I$ where the function Ω_ν attains a value of 1. Since the algorithm corresponding to ν will compute a sequence of values in a distinct way, a special proof of Ω_ν-completeness will be required for each function $\nu\colon \mathbb{R} \to \mathbb{R}^3$. The function Ω will be of use in this step as well. Indeed, the function Ω will be defined such that vectors $\mathbf{v}$ where $\Omega(\mathbf{v}) = 1$ have particular forms. The proof that L_ν is Ω_ν-complete can then be completed by by proving that $x \in L_\nu$ if and

only if the vector $\nu(x)$ has one of these forms. An example of a successful application of this strategy can be found in Section 5, where $\nu = \nu_{\text{trk}}$.

The rest of this section concerns the definition of such a function Ω.

3.1. Cylindrical Distance

Recall from Section 2.1 that the protected zone is a horizontal cylinder around the intruder aircraft that has half-height H and radius D. In order to define the function Ω that satisfies Equation (8), a notion of cylindrical distance is needed.

Definition 5. *The* cylindrical length *of a vector* $\mathbf{w} \in \mathbb{R}^3$ *is the quantity*

$$\|\mathbf{w}\|_{cyl} \equiv \max\left(\frac{\|\mathbf{w}_{(x,y)}\|}{D}, \frac{|\mathbf{w}_z|}{H}\right).$$

Definition 6. *The* cylindrical distance *between two vectors,* $\mathbf{w}_1$ *and* $\mathbf{w}_2$*, is the quantity* $\|\mathbf{w}_1 - \mathbf{w}_2\|_{cyl}$.

Cylindrical distance is a metric on $\mathbb{R}^3$, in the sense of real analysis [8], and $\mathbb{R}^3$ is a metric space with this metric. In particular, this means that the triangle inequality holds for any $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^3$:

$$\|\mathbf{w}_0 - \mathbf{w}_2\|_{\text{cyl}} \leq \|\mathbf{w}_0 - \mathbf{w}_1\|_{\text{cyl}} + \|\mathbf{w}_1 - \mathbf{w}_2\|_{\text{cyl}}. \tag{10}$$

The key property of cylindrical distance, as it relates to loss of separation of aircraft, is stated in the following theorem.

Theorem 2. *Two aircraft are in loss of separation if and only if* $\|\mathbf{s}\|_{cyl} < 1$*, where, as in Section 1,* $\mathbf{s} = \mathbf{s}_o - \mathbf{s}_i$ *is the relative position vector of the aircraft.*

3.2. The Definition of Ω

By Theorem 2, the ownship and the intruder aircraft are in conflict if and only if there exists some time $t \in [0, T]$ such that $\|\mathbf{s} + t\,\mathbf{v}\|_{\text{cyl}} < 1$. Thus, for $\mathbf{s}$ such that $\|\mathbf{s}\|_{\text{cyl}} \neq 1$, i.e., for $\mathbf{s}$ not on the boundary of the protected zone, the function $\Omega(\mathbf{v})$ is defined as

$$\Omega(\mathbf{v}) \equiv \min_{t \in [0,T]} \|\mathbf{s} + t\,\mathbf{v}\|_{\text{cyl}}. \tag{11}$$

Two important remarks on the definition of the function Ω given by Formula (11) are in order. First, the function Ω is well-defined since the quantity $\|\mathbf{s} + t\,\mathbf{v}\|_{\text{cyl}}$ actually attains a minimum as t ranges over the interval

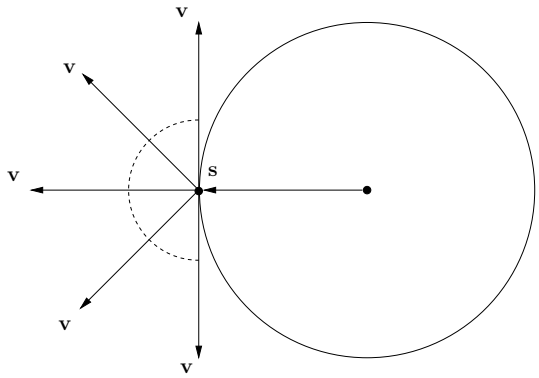

Figure 4: Infinite many places where $\min_{t\in[0,T]} \|\mathbf{s}+t\,\mathbf{v}\|_{\text{cyl}} = 1$

$[0,T]$. That is, there exists some $\tau \in [0,T]$ such that $\|\mathbf{s}+\tau\,\mathbf{v}\|_{\text{cyl}} \leq \|\mathbf{s}+t\,\mathbf{v}\|_{\text{cyl}}$ for all $t \in [0,T]$. Indeed, when the vectors $\mathbf{s}$ and $\mathbf{v}$ are fixed, the function $d_{cyl}\colon [0,T] \mapsto \mathbb{R}$ defined by $d_{cyl}(t) = \|\mathbf{s}+t\,\mathbf{v}\|_{\text{cyl}}$ is continuous, and every continuous function on a closed interval attains a minimum on that interval. The function d_{cyl} is continuous because it is the maximum of two functions, d_{horiz} and d_{vert}, defined by

$$d_{horiz}(t) \equiv \frac{\|(\mathbf{s}+t\,\mathbf{v})_{(x,y)}\|}{D},$$
$$d_{vert}(t) \equiv \frac{|(\mathbf{s}+t\,\mathbf{v})_z|}{H},$$

both of which are continuous.

Second, Formula (11) does not define Ω when $\|\mathbf{s}\|_{\text{cyl}} = 1$. If $\|\mathbf{s}\|_{\text{cyl}} = 1$, in which case $\mathbf{s}$ is on the boundary of the cylinder, then any $\mathbf{v}$ which points outward from the cylinder will satisfy $\min_{t\in[0,T]} \|\mathbf{s}+t\,\mathbf{v}\|_{\text{cyl}} = 1$. This is because the minimum is attained at $t = 0$ for any such $\mathbf{v}$. This is illustrated in Figure 4 in the case where $\|\mathbf{s}_{(x,y)}\| = D$ and $|\mathbf{s}_z| < H$.

Therefore, if $\|\mathbf{s}\|_{\text{cyl}} = 1$, there is an infinite number of vectors $\mathbf{v}$ such that $\min_{t\in[0,T]} \|\mathbf{s}+t\,\mathbf{v}\|_{\text{cyl}} = 1$. Defining Ω in this case using Formula (11) would make L_ν Ω_ν-incomplete, as by definition the sequence L_ν is *finite*.

This shows that some care is needed when defining Ω on the boundary of the cylinder. Formula (12) presents a definition of Ω that is suitable for

showing that a sequence L_ν is Ω_ν-complete.

$$\Omega(\mathbf{v}) \equiv \begin{cases} \mathbf{s}_{(x,y)} \cdot \mathbf{v}_{(x,y)} & \text{if } \|\mathbf{s}_{(x,y)}\| = D \text{ and } |\mathbf{s}_z| < H \\ \mathbf{s}_z \mathbf{v}_z & \text{if } \|\mathbf{s}_{(x,y)}\| < D \text{ and } |\mathbf{s}_z| = H \\ \max(\mathbf{s}_{(x,y)} \cdot \mathbf{v}_{(x,y)}, \mathbf{s}_z \mathbf{v}_z) & \text{if } \|\mathbf{s}_{(x,y)}\| = D \text{ and } |\mathbf{s}_z| = H \\ \min_{t \in [0,T]} \|\mathbf{s} + t\,\mathbf{v}\|_{\text{cyl}} & \text{otherwise, i.e., if } \|\mathbf{s}\|_{\text{cyl}} \neq 1 \end{cases} \tag{12}$$

The following theorem states that Formula (12) defines a function Ω that satisfies Equation (8). The proof of this theorem is a basic exercise in vector algebra.

Theorem 3. *conflict?*$(\mathbf{s}, \mathbf{v}) \iff \Omega(\mathbf{v}) < 1$.

The formal proof that Ω is continuous requires more work and it is explained in the rest of this section. Section 4 provides a classification theorem for Ω, which is used in section 5 to show that a particular prevention bands algorithm, e.g., $\nu = \nu_{\text{trk}}$, is Ω_ν-complete.

3.3. Continuity of Ω

Since the if-statements in the definition of Ω do not depend on $\mathbf{v}$, Ω is continuous if and only if each of the quantities $\mathbf{s}_{(x,y)} \cdot \mathbf{v}_{(x,y)}$, $\mathbf{s}_z \mathbf{v}_z$, $\max(\mathbf{s}_{(x,y)} \cdot \mathbf{v}_{(x,y)}, \mathbf{s}_z \mathbf{v}_z)$, and $\min_{t \in [0,T]} \|\mathbf{s} + t\,\mathbf{v}\|_{\text{cyl}}$ are continuous functions of $\mathbf{v}$. Only one of these four statements is nontrivial, that the minimum $\min_{t \in [0,T]} \|\mathbf{s} + t\,\mathbf{v}\|_{\text{cyl}}$ is continuous in $\mathbf{v}$. This is proved using standard techniques from real analysis [8]. In fact, it follows from a generalization of the Heine-Cantor theorem, which says that a continuous function on a closed interval is uniformly continuous.

Theorem 4. *If A and B are real numbers with $A < B$ and $f\colon [A, B] \times \mathbb{R}^n \mapsto \mathbb{R}$ is continuous, then the function $g\colon \mathbb{R}^n \mapsto \mathbb{R}$ defined by $g(\mathbf{v}) \equiv \min_{t \in [A,B]} f(t, \mathbf{v})$ is continuous.*

The formal proof of this theorem required the development of a vector analysis library in PVS, which is now part of the PVS NASA Libraries.[2]

The continuity of Ω is a direct consequence of Theorem 4, when $A = 0$, $B = T$, and $f(t, \mathbf{v}) = \|\mathbf{s} + t\,\mathbf{v}\|_{\text{cyl}}$.

Theorem 5. *The function Ω is continuous.*

[2]The PVS NASA Libraries are available from `http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html`.

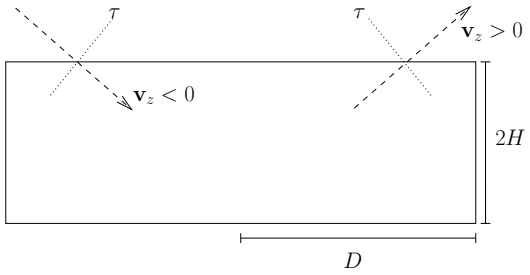

Figure 5: Case $\mathbf{v}_z \neq 0$, $0 < \tau < T$, $|\mathbf{s}_z + \tau\,\mathbf{v}_z| = H$, and $\|(\mathbf{s} + \tau\,\mathbf{v})_{(x,y)}\| < D$

The purpose for constructing the function Ω was to provide a definition for $\Omega_\nu \colon \mathbb{R} \mapsto \mathbb{R}$ for every function $\nu \colon \mathbb{R} \mapsto \mathbb{R}^3$. The following corollaries follow directly from theorems 3 and 5.

Corollary 6. *For any $\nu \colon \mathbb{R} \mapsto \mathbb{R}^3$, the function Ω_ν, defined in Equation (9), satisfies $\Omega_\nu(x) < 1$ if and only if conflict?$(\mathbf{s}, \nu(x) - \mathbf{v}_i)$.*

Corollary 7. *If $\nu \colon \mathbb{R} \mapsto \mathbb{R}^3$ is continuous, then the function Ω_ν is continuous.*

4. Classification of Critical Vectors

To verify the correctness of a prevention bands algorithm for ν over a closed interval I, it must be shown that the computed sequence L_ν is finite and includes all points $x \in I$ such that $\Omega(\nu(x) - \mathbf{v}_i) = 1$. Vectors $\mathbf{v}$ that satisfy $\Omega(\mathbf{v}) = 1$ are called *critical vectors*. This section shows that critical vectors can be analytically classified in a finite way. Since critical vectors correspond to maneuvers that are *tangent* to the protected zone, algorithms for finding maneuvers that result in critical vectors are useful for conflict resolution. Indeed, the classification of critical vectors was originally used for conflict resolution algorithms [9].

Consider a relative position vector $\mathbf{s}$ that satisfies $\|\mathbf{s}\|_{\text{cyl}} \neq 1$ and a critical vector $\mathbf{v}$. Since $\Omega(\mathbf{v}) = 1$, it holds that $\min_{t\in[0,T]} \|\mathbf{s} + t\,\mathbf{v}\|_{\text{cyl}} = 1$. This minimum is attained at a real number $\tau \in [0, T]$. Since $\|\mathbf{s}\|_{\text{cyl}} \neq 1$, it follows that $\tau \neq 0$. Thus, either $\tau = T$ or $0 < \tau < T$. If it holds that $\mathbf{v}_z \neq 0$, $0 < \tau < T$, $|\mathbf{s}_z + \tau\,\mathbf{v}_z| = H$, and $\|(\mathbf{s} + \tau\mathbf{v})_{(x,y)}\| < D$, then it can be shown that $\min_{t\in[0,T]} \|\mathbf{s} + t\,\mathbf{v}\|_{\text{cyl}} < 1$. That is, there is a time near τ where the aircraft will be in loss of separation. This is illustrated in Figure 5.

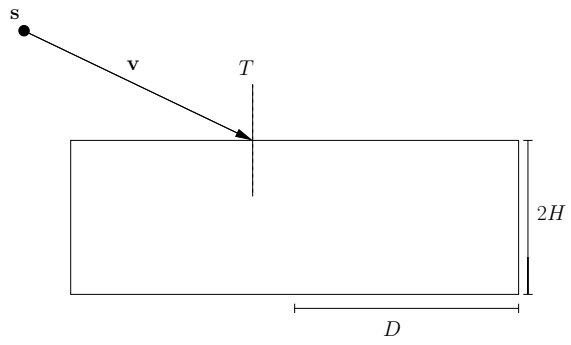

Figure 6: Case $\tau = T$, $|\mathbf{s}_z + T\,\mathbf{v}_z| = H$, and $\|(\mathbf{s} + T\,\mathbf{v})_{(x,y)}\| < D$

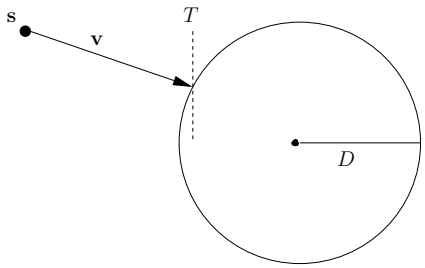

Figure 7: Case $\tau = T$, $|\mathbf{s}_z + T\,\mathbf{v}_z| < H$, and $\|(\mathbf{s} + T\,\mathbf{v})_{(x,y)}\| = D$

If the same conditions hold, but with $\mathbf{v}_z = 0$, then τ is not unique, and it can also be shown that a particular τ can be chosen so that $0 < \tau < T$, $|\mathbf{s}_z + \tau\,\mathbf{v}_z| = H$, and $\|(\mathbf{s} + \tau\,\mathbf{v})_{(x,y)}\| = D$.

Since, $1 = \Omega(\mathbf{v}) = \|\mathbf{s} + \tau\,\mathbf{v}\|_{\text{cyl}} = \max(\frac{\|(\mathbf{s}+\tau\,\mathbf{v})_{(x,y)}\|}{D}, \frac{|\mathbf{s}_z+\tau\,\mathbf{v}_z|}{H})$, this leaves the following cases.

1. Case $\tau = T$, $|\mathbf{s}_z + T\,\mathbf{v}_z| = H$, and $\|(\mathbf{s} + T\,\mathbf{v})_{(x,y)}\| < D$.
2. Case $\tau = T$, $|\mathbf{s}_z + T\,\mathbf{v}_z| < H$, and $\|(\mathbf{s} + T\,\mathbf{v})_{(x,y)}\| = D$.
3. Case $|\mathbf{s}_z + \tau\,\mathbf{v}_z| = H$ and $\|(\mathbf{s} + \tau\,\mathbf{v})_{(x,y)}\| = D$.
4. Case $0 < \tau < T$, $|\mathbf{s}_z + \tau\,\mathbf{v}_z| < H$, and $\|(\mathbf{s} + \tau\,\mathbf{v})_{(x,y)}\| = D$.

These four cases are illustrated in figures 6, 7, 8, and 9, respectively.

These cases will be formalized using four predicates: *vertical_case?* (Section 4.1), *circle_case_2D?* (Section 4.2), *circle_case_3D?* (Section 4.3), and *line_case?* (Section 4.4). It will be shown in Section 4.5 that these four predicates are sufficient to classify solutions to the equation $\Omega(\mathbf{v}) = 1$, even in the case where $\|\mathbf{s}\|_{\text{cyl}} = 1$.

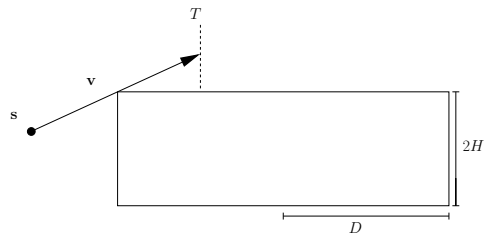

Figure 8: Case $|\mathbf{s}_z + \tau\,\mathbf{v}_z| = H$, and $\|(\mathbf{s} + \tau\,\mathbf{v})_{(x,y)}\| = D$

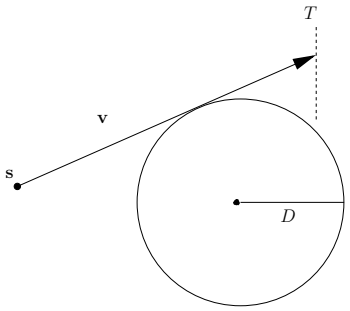

Figure 9: Case $0 < \tau < T$, $|\mathbf{s}_z + \tau\,\mathbf{v}_z| < H$, and $\|(\mathbf{s} + \tau\,\mathbf{v})_{(x,y)}\| = D$

4.1. Vertical Case

Consider the case 1 where $\tau = T$, $|\mathbf{s}_z + T\,\mathbf{v}_z| = H$, and $\|(\mathbf{s} + T\,\mathbf{v})_{(x,y)}\| < D$, which is illustrated by Figure 6. In this case, if $(\mathbf{s}_z + T\,\mathbf{v}_z)\,\mathbf{v}_z > 0$, it is formally proved that there is some $t \in (0, T)$ such that $\|\mathbf{s} + t\,\mathbf{v}\|_{\text{cyl}} < 1$, which is a contradiction. This motivates the definition of the following predicate on $\mathbf{s}_z$, $\mathbf{v}_z$, a real number t, and an integer $\iota = \pm 1$.

$$\begin{aligned} \textit{vertical\_case?}(\mathbf{s}_z, \mathbf{v}_z, t, \iota) \equiv |\mathbf{s}_z + t\,\mathbf{v}_z| = H \text{ and} \\ \iota\ (\mathbf{s}_z + t\,\mathbf{v}_z)\,\mathbf{v}_z \geq 0. \end{aligned} \tag{13}$$

Intuitively, the number ι can be thought of as direction, with $\iota = -1$ corresponding to entry into the protected zone at time t, and $\iota = 1$ corresponding to exit.

Case 1 corresponds to *vertical_case?*$(\mathbf{s}_z, \mathbf{v}_z, T, -1)$. The condition

$$\|(\mathbf{s} + T\,\mathbf{v})_{(x,y)}\| < D$$

is explicitly not included in this predicate, because the more general form is useful when classifying other types of critical vectors. It is important to

note that if $|\mathbf{s}_z + T\,\mathbf{v}_z| = H$, then *vertical_case?*$(\mathbf{s}_z, \mathbf{v}_z, T, \iota)$ holds for some $\iota = \pm 1$.

Vectors $\mathbf{v}$ that satisfy the predicate *vertical_case?* are called *vertical solutions.*

4.2. Circle Case 2D

Consider the case 2 where $\tau = T$, $|\mathbf{s}_z + T\,\mathbf{v}_z| < H$, and $\|(\mathbf{s} + T\,\mathbf{v})_{(x,y)}\| = D$, which is illustrated by Figure 7. If $(\mathbf{s}_{(x,y)} + T\,\mathbf{v}_{(x,y)}) \cdot \mathbf{v}_{(x,y)} > 0$, then it is formally proved that there is some $t \in (0, T)$ such that $\|\mathbf{s} + t\,\mathbf{v}\|_{\text{cyl}} < 1$, which is a contradiction. This motivates the definition of the following predicate on $\mathbf{s}$, $\mathbf{v}$, a real number t, and $\iota = \pm 1$.

$$\begin{aligned} \textit{circle\_case\_2D?}(\mathbf{s}, \mathbf{v}, t, \iota) \equiv \|(\mathbf{s} + t\,\mathbf{v})_{(x,y)}\| = D \text{ and} \\ \iota\ (\mathbf{s}_{(x,y)} + t\,\mathbf{v}_{(x,y)}) \cdot \mathbf{v}_{(x,y)} \geq 0. \end{aligned} \tag{14}$$

Case 2 corresponds to *circle_case_2D?*$(\mathbf{s}, \mathbf{v}, T, -1)$. The condition

$$|\mathbf{s}_z + T\,\mathbf{v}_z| < H$$

is not included in this predicate, because it will be used, along with *vertical_case?*, to classify other types of critical vectors. As for the predicate *vertical_case?* above, an important property of *circle_case_2D?* is that $\|(\mathbf{s} + t\,\mathbf{v})_{(x,y)}\| = D$ implies that *circle_case_2D?*$(\mathbf{s}, \mathbf{v}, t, \iota)$ holds for some $\iota = \pm 1$.

Vectors $\mathbf{v}$ that satisfy the predicate *circle_case_2D?* are called *2D circle solutions.*

4.3. Circle Case 3D

Consider the case 3 where $|\mathbf{s}_z + \tau\,\mathbf{v}_z| = H$ and $\|(\mathbf{s} + \tau\,\mathbf{v})_{(x,y)}\| = D$, which is illustrated by Figure 8. It follows from the definitions of *vertical_case?* and *circle_case_2D?* that there exists ι_1, ι_2, each equal to -1 or 1, such that *vertical_case?*$(\mathbf{s}_z, \mathbf{v}_z, \tau, \iota_1)$ and *circle_case_2D?*$(\mathbf{s}, \mathbf{v}, \tau, \iota_2)$. If τ is positive and $\iota_1 = \iota_2$, it can be proved that either *vertical_case?*$(\mathbf{s}_z, \mathbf{v}_z, T, -1)$ or $\Omega(\mathbf{v}) < 1$. In classifying the solutions to the equation $\Omega(\mathbf{v}) = 1$, the case where *vertical_case?*$(\mathbf{s}_z, \mathbf{v}_z, T, -1)$ is true is handled separately. Since it holds that $\Omega(\mathbf{v}) = 1$, a requirement for the case where $|\mathbf{s}_z + \tau\,\mathbf{v}_z| = H$ and $\|(\mathbf{s} + \tau\,\mathbf{v})_{(x,y)}\| = D$ is therefore that $\iota_1 = -\iota_2$. This motivates the definition of the following predicate. Similar to the predicate *circle_case_2D?*, this predicate depends on $\mathbf{s}$, $\mathbf{v}$, $\iota = \pm 1$, and a real number t.

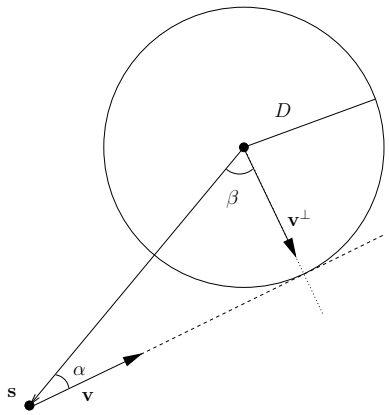

Figure 10: Line case: $\mathbf{v}$ is tangent to the circle

$$\begin{aligned} \textit{circle\_case\_3D?}(\mathbf{s},\mathbf{v},t,\iota) \equiv\ & t > 0 \text{ and} \\ & \textit{circle\_case\_2D?}(\mathbf{s},\mathbf{v},t,\iota) \text{ and} \\ & \textit{vertical\_case?}(\mathbf{s}_z,\mathbf{v}_z,t,-\iota). \end{aligned} \tag{15}$$

Vectors $\mathbf{v}$ that satisfy the predicate *circle_case_3D?* are called *3D circle solutions.*

4.4. Line Case

Consider the case 4 where $0 < \tau < T$, $|\mathbf{s}_z + \tau\,\mathbf{v}_z| < H$, and $\|(\mathbf{s} + \tau\,\mathbf{v})_{(x,y)}\| = D$, which is illustrated by Figure 9. As Figure 10 indicates, the fact that τ satisfies $\min_{t\in[0,T]} \|\mathbf{s}+t\,\mathbf{v}\|_{\text{cyl}} = \|\mathbf{s}+\tau\,\mathbf{v}\|_{\text{cyl}}$ can be used to show that the trajectory from $\mathbf{s}_{(x,y)}$ along $\mathbf{v}_{(x,y)}$ is tangent to the circle of radius D around the origin. In this figure, the vector $\mathbf{v}^\perp$ is the vector $(v_y, -v_x, \mathbf{v}_z)$.

It is immediately clear from Figure 10 that the angle α can be no greater than $\pi/2$. Since $\mathbf{s}_{(x,y)} \cdot -\mathbf{v}_{(x,y)} = \|\mathbf{s}_{(x,y)}\|\|\mathbf{v}_{(x,y)}\| \cos\alpha \geq 0$, it follows that $\mathbf{s}_{(x,y)} \cdot \mathbf{v}_{(x,y)} \leq 0$. In addition, $\cos\beta = \frac{D}{\|\mathbf{s}_{(x,y)}\|}$. Thus,

$$\begin{aligned} \mathbf{s}_{(x,y)} \cdot \mathbf{v}^\perp{}_{(x,y)} &= \|\mathbf{s}_{(x,y)}\|\|\mathbf{v}_{(x,y)}\| \cos\beta \\ &= D\|\mathbf{v}_{(x,y)}\|. \end{aligned} \tag{16}$$

This construction depends on a vector $\mathbf{v}_{(x,y)}$ that is tangent to the right side of the circle. The analogous construction for a vector $\mathbf{v}_{(x,y)}$ that is tangent to the left side of the circle would use $-\mathbf{v}^\perp$ in the place of the vector

$\mathbf{v}^{\perp}$. This motivates the definition of the following predicate, which depends on $\mathbf{s}$, $\mathbf{v}$, and a parameter ε, which is equal to either -1 for a right-tangent, or 1 for a left-tangent.

$$\begin{aligned} \textit{line\_case?}(\mathbf{s}, \mathbf{v}, \varepsilon) \equiv \mathbf{s}_{(x,y)} \cdot \mathbf{v}_{(x,y)} \leq 0 \text{ and} \\ -\varepsilon\ (\mathbf{s}_{(x,y)} \cdot \mathbf{v}^{\perp}{}_{(x,y)}) = D\|\mathbf{v}_{(x,y)}\|. \end{aligned} \tag{17}$$

Vectors $\mathbf{v}$ that satisfy the predicate *line_case?* are called *line solutions.*

4.5. The Classification Theorem

Critical vectors can be classified according to the following theorem.

Theorem 8. *If $\Omega(\mathbf{v}) = 1$, then one of the following conditions holds.*

1. $\|\mathbf{s}_{(x,y)}\| \geq D$ *and line_case?*$(\mathbf{s}, \mathbf{v}, \iota)$ *holds for some* $\iota = \pm 1$.
2. $|\mathbf{s}_z + T\,\mathbf{v}_z| < H$ *and circle_case_2D?*$(\mathbf{s}, \mathbf{v}, T, -1)$
3. *There exists a real number* $t > 0$ *such circle_case_3D?*$(\mathbf{s}, \mathbf{v}, t, \iota)$ *holds for some* $\iota = \pm 1$.
4. $\|\mathbf{s}_{(x,y)} + T\,\mathbf{v}_{(x,y)}\| \leq D$ *and vertical_case?*$(\mathbf{s}_z, \mathbf{v}_z, T, -1)$

This theorem can be used to show that a sequence L_ν computed by a prevention bands algorithm for ν over a closed interval I is Ω_ν-complete by proving that L_ν contains all the vectors that have one of the four forms. It follows from this that L_ν contains all points $x \in I$ such that $\Omega_\nu(x) = 1$.

5. Track Angle Prevention Bands

This section presents a formally verified algorithm, namely `track_bands`, for track angle prevention bands over the closed interval $[0, 2\pi]$, for the function $\nu_{\text{trk}} \colon \mathbb{R} \mapsto \mathbb{R}^3$, defined by Equation (2) in Section 2.2. The purpose is to illustrate the usefulness of the approach outlined in the previous sections for verifying prevention bands algorithms. Similar algorithms, for ground speed and vertical speed maneuvers, have been formally verified using this approach [6].

The definition of `track_bands` depends on the algorithms `track_line`, `track_circle_2D`, and `track_circle_3D`, which compute track angle maneuvers that are line solutions, 2D circle solutions, and 3D circle solutions, respectively. These three algorithms are proved to be *complete*, i.e., they compute all vectors that satisfy their respective predicate, and *correct*, i.e., only vectors that satisfy their respective predicate are computed. The correctness of `track_bands` depends on the completeness of `track_line`, `track_circle_3D`, and `track_circle_2D`.

5.1. A Special Case

For $\nu = \nu_{\text{trk}}$, the function Ω_ν, defined in Formula (9) of Section 3, characterizes conflict in the sense of Corollary 6 (Section 3.3). To prove the correctness of a track angle prevention bands algorithm, it must be shown that the finite sequence L_ν returned by the algorithm contains all track angles $\alpha \in [0, 2\pi]$ such that $\Omega_\nu(\alpha) = 1$. An obvious requirement is that there be only finitely many track angles in the interval $[0, 2\pi]$ for which this equation holds. As it turns out, there are several special cases where this equation has infinitely many solutions for track angles $\alpha \in [0, 2\pi]$. These special cases are specified by the following predicate.

$$\begin{aligned} \textit{track\_spc?}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, t) \equiv \mathbf{s}_{(x,y)} = t\mathbf{v}_{i(x,y)} \text{ and} \\ \|\mathbf{v}_{o(x,y)}\|^2 = \frac{D^2}{t^2}. \end{aligned} \tag{18}$$

The approach outlined in this paper for verifying a prevention bands algorithm using a function Ω_ν can be used in every case to verify the correctness of the algorithm `track_bands`. However, in some special cases where *track_spc?* holds, a special version of Ω_ν must be defined in place of the definition given by Formula (9) of Section 3. For simplicity, these cases have been excluded from the proofs in this paper. In the following sections, the exclusion of these cases is explicitly noted where applicable. For a complete discussion of the verification of the algorithm presented here, see [6].

5.2. Line Solutions For Track Angle Maneuvers

The algorithm `track_line`, defined below, takes as parameters $\mathbf{s}$, $\mathbf{v}_o$, $\mathbf{v}_i$, t, $\varepsilon = \pm 1$, and $\iota = \pm 1$. It returns a vector $\mathbf{v}'_o \in \mathbb{R}^3$ that is either the zero vector or is equal to $\nu_{\text{trk}}(\alpha)$ for some $\alpha \in [0, 2\pi)$ such that the relative velocity vector $\mathbf{v}' = \mathbf{v}'_o - \mathbf{v}_i$ is tangent to the circle, i.e., it satisfies *line_case?*$(\mathbf{s}, \mathbf{v}', \varepsilon)$.

The definition of `track_line` requires the definition an auxiliary function, namely `tangent_line`, that takes as parameter a relative position vector $\mathbf{s} \in \mathbb{R}^3$ such that $\|\mathbf{s}_{(x,y)}\| \geq D$ and a number $\varepsilon = \pm 1$, and returns a

vector in $\mathbb{R}^3$ that is tangent to the protected zone.

$$
\begin{array}{l}
\texttt{tangent\_line}(\mathbf{s}, \varepsilon) \equiv \\
\quad \texttt{if } \|\mathbf{s}_{(x,y)}\| = D \texttt{ then } \varepsilon\, \mathbf{s}^{\perp} \\
\quad \texttt{else} \\
\quad\quad \texttt{let } d = \|\mathbf{s}_{(x,y)}\|^2 \texttt{ in} \\
\quad\quad\quad (\dfrac{D^2}{d} - 1)\, \mathbf{s} + \dfrac{\varepsilon\, D\sqrt{d - D^2}}{d}\, \mathbf{s}^{\perp} \\
\quad \texttt{endif}
\end{array}
\tag{19}
$$

The proofs of the following lemmas rely on standard vector algebra.

Lemma 9. *If* $\|\mathbf{s}_{(x,y)}\| \geq D$ *and* $\varepsilon = \pm 1$*, then*

$$\mathit{line\_case?}(\mathbf{s}, \mathit{tangent\_line}(\mathbf{s}, \varepsilon), \varepsilon)$$

holds.

The algorithm `track_line` is defined as follows.

$$
\begin{array}{l}
\texttt{track\_line}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \varepsilon, \iota) \equiv \\
\quad \texttt{let} \\
\quad\quad k \;\; = \texttt{track\_only\_line}(\texttt{tangent\_line}(\mathbf{s}, \varepsilon)_{(x,y)}, \mathbf{v}_o, \mathbf{v}_i, \iota), \\
\quad\quad \mathbf{v}'_o = (k\ \texttt{tangent\_line}(\mathbf{s}, \varepsilon)_{(x,y)} + \mathbf{v}_{i(x,y)})\ \texttt{ with } [z \leftarrow \mathbf{v}_{oz}] \\
\quad \texttt{in} \\
\quad\quad \texttt{if } k \geq 0 \texttt{ then } \mathbf{v}'_o \texttt{else } \mathbf{0} \texttt{ endif}
\end{array}
\tag{20}
$$

where

$$
\begin{array}{l}
\texttt{track\_only\_line}(\mathbf{u}, \mathbf{v}_o, \mathbf{v}_i, \iota) \equiv \\
\quad \texttt{let} \\
\quad\quad a = \|\mathbf{u}\|^2, \\
\quad\quad b = 2\, \mathbf{v}_{i(x,y)} \cdot \mathbf{u}, \\
\quad\quad c = \|\mathbf{v}_{i(x,y)}\|^2 - \|\mathbf{v}_{o(x,y)}\|^2 \\
\quad \texttt{in} \\
\quad\quad \texttt{if } b^2 - 4ac \geq 0 \texttt{ then} \\
\quad\quad\quad \dfrac{-b + \iota\sqrt{b^2 - 4ac}}{2a} \\
\quad\quad \texttt{else } 0 \\
\quad\quad \texttt{endif}
\end{array}
\tag{21}
$$

The next lemma states that the algorithm `track_only_line` computes solutions for k to the equation $\mathbf{v}'_{o(x,y)} = k\ \mathbf{u} + \mathbf{v}_{i(x,y)}$, where $\|\mathbf{v}'_{o(x,y)}\| = \|\mathbf{v}_{o(x,y)}\|$.

Lemma 10. *If* $\mathbf{u} \neq \mathbf{0}$, *then* $\|\mathbf{v}'_{o(x,y)}\| = \|\mathbf{v}_{o(x,y)}\|$ *and* $k\ \mathbf{u} = \mathbf{v}'_{o(x,y)} - \mathbf{v}_{i(x,y)}$ *if and only if*

$$k = \mathit{track\_only\_line}(\mathbf{u}, \mathbf{v}_o, \mathbf{v}_i, \iota),$$

for some $\iota = \pm 1$.

The proofs of the correctness and completeness of `track_line` follow from its definition and Lemma 10.

Theorem 11 (Correctness and completeness of `track_line`)**.** *If* $\|\mathbf{s}_{(x,y)}\| \geq D$ *and* $\mathbf{v}'_{o(x,y)} \neq \mathbf{0}$, *then* $\|\mathbf{v}'_{o(x,y)}\| = \|\mathbf{v}_{o(x,y)}\|$, $\mathbf{v}'_{oz} = \mathbf{v}_{oz}$, *and line_case?*$(\mathbf{s}, \mathbf{v}'_o - \mathbf{v}_i, \varepsilon)$ *holds if and only if*

$$\mathbf{v}'_o = \mathit{track\_line}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \varepsilon, \iota),$$

for some $\iota = \pm 1$.

5.3. 2D Circle Solutions For Track Angle Maneuvers

The algorithm `track_circle_2D`, defined below, takes as parameters $\mathbf{s}$, $\mathbf{v}_o$, $\mathbf{v}_i$, t, $\iota = \pm 1$, and $\varepsilon = \pm 1$. It returns a vector $\mathbf{v}'_o \in \mathbb{R}^3$ that is either the zero vector or is equal to $\nu_{\text{trk}}(\alpha)$ for some $\alpha \in [0, 2\pi)$ such that the relative velocity vector $\mathbf{v}' = \mathbf{v}'_o - \mathbf{v}_i$ satisfies *circle_case_2D?*$(\mathbf{s}, \mathbf{v}', t, \iota)$.

$$
\begin{array}{l}
\texttt{track\_circle\_2D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, t, \iota, \varepsilon) \equiv \\
\quad \texttt{let} \\
\quad\quad \mathbf{u} = (\mathbf{s} - t\mathbf{v}_i)_{(x,y)}, \\
\quad\quad j = \dfrac{1}{2t}(D^2 - \|\mathbf{s}_{(x,y)}\|^2 - t^2(\|\mathbf{v}_{o(x,y)}\|^2 - \|\mathbf{v}_{i(x,y)}\|^2)) \\
\quad \texttt{in} \\
\quad\quad \texttt{if}\ \mathbf{u} \neq \mathbf{0}\ \texttt{then} \\
\quad\quad\quad \texttt{let} \\
\quad\quad\quad\quad \mathbf{v}'_o = \texttt{track\_only\_dot}(\mathbf{u}, \mathbf{v}_o, \mathbf{v}_i, j, \varepsilon) \\
\quad\quad\quad \texttt{in} \\
\quad\quad\quad\quad \texttt{if}\ \iota\ (\mathbf{s} + t\ (\mathbf{v}'_o - \mathbf{v}_i)) \geq 0\ \texttt{then}\ \mathbf{v}'_o \\
\quad\quad\quad\quad \texttt{else}\ \mathbf{0} \\
\quad\quad\quad\quad \texttt{endif} \\
\quad\quad \texttt{else}\ \mathbf{0} \\
\quad\quad \texttt{endif}
\end{array}
\tag{22}
$$

where

$$\begin{aligned}&\texttt{track\_only\_dot}(\mathbf{u}, \mathbf{v}_o, \mathbf{v}_i, j, \iota) \equiv \\ &\quad \texttt{let } k = \texttt{track\_only\_line}(\mathbf{u}^\perp, \mathbf{v}_o, \mathbf{v}_i + \frac{j}{\|\mathbf{u}_{(x,y)}\|^2}\mathbf{u}, \iota) \texttt{ in} \\ &\qquad (k\mathbf{u}^\perp + \mathbf{v}_{i(x,y)} + \frac{j}{\|\mathbf{u}_{(x,y)}\|^2}\mathbf{u}) \texttt{ with } [z \leftarrow \mathbf{v}_{oz}]\end{aligned} \tag{23}$$

The next lemma shows that the algorithm `track_only_dot` can be used to solve equations of the form $\mathbf{u} \cdot (\mathbf{v}'_{o(x,y)} - \mathbf{v}_{i(x,y)}) = j$ for $\mathbf{v}'_o$ when $\|\mathbf{v}'_{o(x,y)}\| = \|\mathbf{v}_{o(x,y)}\|$.

Lemma 12. *For all $j \in \mathbb{R}$, $\mathbf{u} \neq \mathbf{0}$, and $\mathbf{v}'_{o(x,y)} \neq \mathbf{0}$, $\|\mathbf{v}'_{o(x,y)}\| = \|\mathbf{v}_{o(x,y)}\|$, $\mathbf{v}'_{oz} = \mathbf{v}_{oz}$, and $\mathbf{u} \cdot (\mathbf{v}'_{o(x,y)} - \mathbf{v}_{i(x,y)}) = j$ if and only if*

$$\mathbf{v}'_o = \textit{track\_only\_dot}(\mathbf{u}, \mathbf{v}_o, \mathbf{v}_i, j, \iota),$$

for some $\iota = \pm 1$.

The correctness and completeness of `track_circle_2D` follow from its definition and Lemma 12.

Theorem 13 (Correctness of `track_circle_2D`)**.** *If $\mathbf{v}'_{o(x,y)} \neq \mathbf{0}$ and*

$$\mathbf{v}'_o = \textit{track\_circle\_2D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, t, \iota, \varepsilon),$$

then $\|\mathbf{v}'_{o(x,y)}\| = \|\mathbf{v}_{o(x,y)}\|$, $\mathbf{v}'_{oz} = \mathbf{v}_{oz}$, and circle_case_2D?$(\mathbf{s}, \mathbf{v}'_o - \mathbf{v}_i, t, \iota)$ holds.

Theorem 14 (Completeness of `track_circle_2D`)**.** *If $\|\mathbf{v}'_{o(x,y)}\| = \|\mathbf{v}_{o(x,y)}\|$, $\mathbf{v}'_{oz} = \mathbf{v}_{oz}$, and*

$$\textit{circle\_case\_2D?}(\mathbf{s}, \mathbf{v}'_o - \mathbf{v}_i, t, \iota)$$

holds, then either track_spc?$(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, t)$ holds or

$$\mathbf{v}'_o = \textit{track\_circle\_2D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, t, \iota, \varepsilon)$$

for some $\varepsilon = \pm 1$.

5.4. 3D Circle Solutions For Track Angle Maneuvers

Theorems 13 and 14 imply that the algorithm `track_circle_2D` can be used to compute vectors $\mathbf{v}'_o$ such that *circle_case_2D?*$(\mathbf{s}, \mathbf{v}'_o - \mathbf{v}_i, t, \iota)$ holds, where $t > 0$. By the definition of the predicate *circle_case_3D?* in Section 4.3, this algorithm can be used to compute vectors $\mathbf{v}'_o$ such that

circle_case_3D?$(\mathbf{s}, \mathbf{v}'_o - \mathbf{v}_i, \Theta_H(\mathbf{s}_z, \mathbf{v}_{oz} - \mathbf{v}_{iz}, -\iota), \iota)$ holds when $\Theta_H(\mathbf{s}_z, \mathbf{v}_{oz} - \mathbf{v}_{iz}, -\iota) > 0$, where

$$\Theta_H(\mathbf{s}_z, \mathbf{v}_z, \iota) \equiv \frac{\iota \; sign(\mathbf{v}_z)\, H - \mathbf{s}_z}{\mathbf{v}_z}, \text{ for } \mathbf{v}_z \neq 0. \tag{24}$$

It is easy to check that Θ_H satisfies $|\mathbf{s}_z + \Theta_H(\mathbf{s}_z, \mathbf{v}_z, \iota)\, \mathbf{v}_z| = H$. In addition,

$$\Theta_H(\mathbf{s}_z, \mathbf{v}_z, -1) < \Theta_H(\mathbf{s}_z, \mathbf{v}_z, 1). \tag{25}$$

Intuitively, the times $\Theta_H(\mathbf{s}_z, \mathbf{v}_z, -1)$ and $\Theta_H(\mathbf{s}_z, \mathbf{v}_z, 1)$ are the times at which the z-component of the trajectory from $\mathbf{s}$ along $\mathbf{v}$ enters and exits the interval $[-H, H]$, respectively.

This motivates the definition of the algorithm `track_circle_3D`, which takes as a parameters $\mathbf{s}$, $\mathbf{v}_o$, $\mathbf{v}_i$, $\iota = \pm 1$, and $\varepsilon = \pm 1$. It returns a vector $\mathbf{v}'_o \in \mathbb{R}^3$ that is either the zero vector or is equal to $\nu_{\text{trk}}(\alpha)$ for some $\alpha \in [0, 2\pi)$ such that the relative velocity vector $\mathbf{v}' = \mathbf{v}'_o - \mathbf{v}_i$ satisfies *circle_case_3D?*$(\mathbf{s}, \mathbf{v}', \Theta_H(\mathbf{s}_z, \mathbf{v}_{oz} - \mathbf{v}_{iz}, -\iota), \iota)$.

```
track_circle_3D(s, v_o, v_i, ι, ε) ≡
    if v_oz = v_iz then 0
    else
        let t = Θ_H(s_z, v_oz − v_iz, −ι) in
            if t > 0 then
                track_circle_2D(s, v_o, v_i, t, ι, ε)
            else 0
            endif
    endif
```
(26)

The following theorems state that `track_circle_3D` is correct and complete for 3D circle solutions that are track angle maneuvers.

Theorem 15 (Correctness of `track_circle_3D`)**.** *If* $\mathbf{v}'_{o(x,y)} \neq \mathbf{0}$ *and*

$$\mathbf{v}'_o = \textit{track\_circle\_3D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \iota, \varepsilon),$$

then $\|\mathbf{v}'_{o(x,y)}\| = \|\mathbf{v}_{o(x,y)}\|$, $\mathbf{v}'_{oz} = \mathbf{v}_{oz}$, *and*

$$\textit{circle\_case\_3D?}(\mathbf{s}, \mathbf{v}'_o - \mathbf{v}_i, \Theta_H(\mathbf{s}_z, \mathbf{v}_{oz} - \mathbf{v}_{iz}, -\iota), \iota)$$

holds.

Theorem 16 (Completeness of `track_circle_3D`). *If* $\|\mathbf{v}'_{o(x,y)}\| = \|\mathbf{v}_{o(x,y)}\|$, $\mathbf{v}'_{oz} = \mathbf{v}_{oz}$, $\mathbf{v}_{oz} \neq \mathbf{v}_{iz}$, *and circle_case_3D?*$(\mathbf{s}, \mathbf{v}'_o - \mathbf{v}_i, \Theta_H(\mathbf{s}_z, \mathbf{v}_{oz} - \mathbf{v}_{iz}, -\iota), \iota)$ *holds, then either track_spc?*$(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \Theta_H(\mathbf{s}_z, \mathbf{v}_{oz} - \mathbf{v}_{iz}, -\iota))$ *holds or*

$$\mathbf{v}'_o = \textit{track\_circle\_3D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \iota, \varepsilon),$$

for some $\varepsilon = \pm 1$.

5.5. A Prevention Bands Algorithm For Track Angle Maneuvers

Using the functions defined in the previous section, the prevention bands algorithm `track_bands` for the function $\nu_{\text{trk}} \colon \mathbb{R} \mapsto \mathbb{R}^3$ can be defined as follows. The function `sort` takes a set of real numbers as argument and returns the sequence of elements in the set that is sorted by increasing order. The function `track`, specified by Formula (3) in Section 2.2, takes a vector as argument and returns its track angle. It is assumed that $\texttt{track}(\mathbf{0}) = 0$. That function is implemented using a two-argument arc tangent function, usually called `atan2` in programming languages.

$$\begin{array}{l}
\texttt{track_bands}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i) \equiv \\
\quad \texttt{sort}(\{0, 2\pi\} \cup \\
\qquad \{\texttt{track}(\texttt{track_circle_3D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, -1, -1)), \\
\qquad\ \texttt{track}(\texttt{track_circle_3D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, -1, 1)), \\
\qquad\ \texttt{track}(\texttt{track_circle_3D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, 1, -1)), \\
\qquad\ \texttt{track}(\texttt{track_circle_3D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, 1, 1))\} \cup \\
\qquad \texttt{if } \|\mathbf{s}_{(x,y)}\| \geq D \texttt{ then} \\
\qquad\quad \{\texttt{track}(\texttt{track_circle_2D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, T, -1, -1)), \\
\qquad\quad\ \texttt{track}(\texttt{track_circle_2D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, T, -1, 1)), \\
\qquad\quad\ \texttt{track}(\texttt{track_line}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, -1, -1)), \\
\qquad\quad\ \texttt{track}(\texttt{track_line}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, -1, 1)), \\
\qquad\quad\ \texttt{track}(\texttt{track_line}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, 1, -1)), \\
\qquad\quad\ \texttt{track}(\texttt{track_line}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, 1, 1))\})\} \\
\qquad \texttt{else } \emptyset \\
\qquad \texttt{endif})
\end{array} \tag{27}$$

The finite, ordered sequence returned by `track_bands` is computed using every possible instantiation of the parameters ε and ι, both of which can be ± 1, in `track_line`, `track_circle_2D`, and `track_circle_3D`. For each

vector $\mathbf{v}'_o$ returned by one of these three algorithms for $\mathbf{s}$, $\mathbf{v}_o$, and $\mathbf{v}_i$, the track angle of $\mathbf{v}'_o$ is an element of the sequence returned by `track_bands`.

Theorem 17 (Correctness of `track_bands`)**.** *The track angle prevention bands algorithm* ***`track_bands`*** *is correct for ν_{trk} over the interval $[0, 2\pi]$.*

Proof. By Theorem 1 in Section 2.5 and Corollary 6 in Section 3.3, it suffices to prove that the function Ω_ν, defined in Formula 9 in Section 3, satisfies the following property: For all $\alpha \in [0, 2\pi]$, $\Omega_\nu(\alpha) = 1$ implies that $\alpha \in \texttt{track\_bands}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i)$.

This proof excludes the cases defined by the predicate *track_spc?* (Section 5.1). For an outline of the complete proof, see [6]. Here, the proof is restricted to the cases where neither of the following conditions hold.

1. $\mathbf{v}_{oz} \neq \mathbf{v}_{iz}$ and there exists $\iota = \pm 1$ such that *track_spc?*$(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, t)$ and $0 < t \leq T$, where $t = \Theta_H(\mathbf{s}_z, \mathbf{v}_{oz} - \mathbf{v}_{iz}, \iota)$.
2. *track_spc?*$(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, T)$ and $|\mathbf{s}_z + T\ (\mathbf{v}_{oz} - \mathbf{v}_{iz})| < H$.

Suppose that $\alpha \in [0, 2\pi]$ and $\Omega_\nu(\alpha) = 1$. Since $\Omega_\nu(\alpha) = \Omega(\nu_{\text{trk}}(\alpha) - \mathbf{v}_i)$, Theorem 8 in Section 4.5 implies that one of the following conditions holds, where $\mathbf{v} = \nu_{\text{trk}}(\alpha) - \mathbf{v}_i$.

- $\|\mathbf{s}_{(x,y)}\| \geq D$ and *line_case?*$(\mathbf{s}, \mathbf{v}, \varepsilon)$, for some $\varepsilon = \pm 1$.
- $|\mathbf{s}_z + T\mathbf{v}_z| < H$ and *circle_case_2D?*$(\mathbf{s}, \mathbf{v}, T, -1)$.
- There is some real number $t > 0$ such that *circle_case_3D?*$(\mathbf{s}, \mathbf{v}, t, \iota)$, for some $\iota = \pm 1$.
- $\|\mathbf{s}_{(x,y)} + T\mathbf{v}_{(x,y)}\| \leq D$ and *vertical_case?*$(\mathbf{s}_z, \mathbf{v}_z, T, -1)$.

These cases are now considered individually.

- Suppose first that $\|\mathbf{s}_{(x,y)}\| \geq D$ and *line_case?*$(\mathbf{s}, \nu_{\text{trk}}(\alpha) - \mathbf{v}_i, \varepsilon)$, for some $\varepsilon = \pm 1$. By completeness of `track_line` (Theorem 11), $\nu_{\text{trk}}(\alpha)$ is equal to $\texttt{track\_line}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \varepsilon, \iota)$, for some $\iota = \pm 1$. Thus, $\alpha = \texttt{track}(\nu_{\text{trk}}(\alpha))$ is equal to $\texttt{track}(\texttt{track\_line}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \varepsilon, \iota))$, which, by definition, is an element of $\texttt{track\_bands}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i)$.
- Suppose that $|\mathbf{s}_z + T\mathbf{v}_z| < H$ and *circle_case_2D?*$(\mathbf{s}, \nu_{\text{trk}}(\alpha) - \mathbf{v}_i, T, -1)$. By completeness of the algorithm `track_circle_2D` (Theorem 14), $\nu_{\text{trk}}(\alpha)$ is equal to $\texttt{track\_circle\_2D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, t, \iota, \varepsilon)$, for some $\iota = \pm 1$ and $\varepsilon = \pm 1$. Thus,

  $$\alpha = \texttt{track}(\nu_{\text{trk}}(\alpha)) = \texttt{track}(\texttt{track\_circle\_2D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, t, \iota, \varepsilon)).$$

  Hence, α is an element of $\texttt{track\_bands}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i)$.

- Suppose that there is a real number $t > 0$ such that

$$\textit{circle\_case\_3D?}(\mathbf{s}, \mathbf{v}, t, \iota),$$

where $\iota = \pm 1$. Assume that $\mathbf{v}_{oz} \neq \mathbf{v}_{iz}$. By completeness of the algorithm `track_circle_3D` (Theorem 16),

$$\nu_{\text{trk}}(\alpha) = \texttt{track\_circle\_3D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \iota, \varepsilon),$$

for some $\iota = \pm 1$ and $\varepsilon = \pm 1$. Thus, as above,

$$\alpha = \texttt{track}(\nu_{\text{trk}}(\alpha)) = \texttt{track}(\texttt{track\_circle\_3D}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \iota, \varepsilon)).$$

Hence, α is an element of $\texttt{track\_bands}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i)$. The case where $\mathbf{v}_{oz} = \mathbf{v}_{iz}$ can be equally discharged.

- Finally, suppose that $\|\mathbf{s}_{(x,y)} + T\mathbf{v}_{(x,y)}\| \leq D$ and

$$\textit{vertical\_case?}(\mathbf{s}_z, \mathbf{v}_z, T, -1).$$

In this case, $\nu_{\text{trk}}(\alpha)_z = \mathbf{v}_{oz}$ implies that $\textit{conflict?}(\mathbf{s}, \nu_{\text{trk}}(\alpha) - \mathbf{v}_i)$ does not hold for any $\alpha \in \mathbb{R}$. From there, the correctness of the algorithm `track_bands` is trivial.

□

6. Conclusion

This paper presents a general method for proving that a prevention bands algorithm for a set of maneuvers defined by a function $\nu : \mathbb{R} \mapsto \mathbb{R}^3$ is correct, i.e., that the algorithm correctly computes all the critical values where the prevention bands potentially change color. A direct proof that a prevention bands algorithm is correct is tedious, error prone, and difficult. This paper proposes a method that decomposes the correctness proof into two steps:

1. The definition of a continuous function $\Omega_\nu : \mathbb{R} \mapsto \mathbb{R}$ that characterizes conflicts, i.e., $\Omega_\nu(x) < 1$ if and only if $\textit{conflict?}(\mathbf{s}, \nu(x) - \mathbf{v}_i)$.
2. A proof that the algorithm is Ω_ν-complete, i.e., that it finds all values x where $\Omega_\nu(x) = 1$.

In most cases, the function Ω_ν can be defined independently of the algorithm using a function $\Omega : \mathbb{R}^3 \mapsto R$ provided in this paper. The proof that Ω_ν is continuous and correctly characterizes conflicts is given once and for all and only depends on the continuity of ν, which is typically easy to verify.

The method presented here also provides a classification theorem for Ω_ν that characterizes the vectors that satisfy $\Omega(\mathbf{v}) = 1$. Using this theorem, proving that the algorithm is Ω_ν-complete reduces to proving that the algorithm correctly computes all vectors that have a particular form. The method is illustrated with the proof of correctness of a prevention bands algorithm for track angle maneuvers that was originally presented, without verification, in [3].

This paper focuses on pairwise algorithms, i.e., it considers only one traffic aircraft, the intruder. Prevention bands algorithms for an arbitrary number of traffic aircraft can be obtained from a pairwise algorithm by simply letting the conflict bands for n-aircraft be the union of the conflict bands computed for the ownship and each individual traffic aircraft. The conflict-free bands can be computed as the complement of the conflict bands. The correctness of the algorithms for n-aircraft can be easily derived from the correctness of the pairwise prevention bands algorithms.

The prevention bands algorithm presented here is part of NASA's Airborne Coordinated Conflict Resolution and Detection (ACCORD) framework (`http://shemesh.larc.nasa.gov/people/cam/ACCoRD`). ACCoRD is a PVS development for the design and verification of state-based separation assurance systems, including formally verified algorithms for conflict detection, conflict resolution, loss of separation recovery, and conflict preventions bands. These algorithms are being integrated into NASA's air traffic simulation environments such as Autonomous Operations Planner (AOP) [10], Traffic Manager (TMX) [11], and Airspace Concept Evaluation System (ACES) [12].

The results presented in this paper have been mechanically checked using an interactive theorem prover, which provides strong guarantees that the mathematical development is correct. Although other researchers have looked into the formal verification of air traffic management systems [13, 14], the authors are not aware of formal verification efforts of air traffic management systems entirely based on theorem proving. The use of a mechanical theorem prover entails a detailed description of the problem and a meticulous proof process. This level of rigor is justified by the critical role that aircraft separation plays in the overall safety of the next generation of air traffic management systems.

References

[1] R. Butler, G. Hagen, J. Maddalon, C. Muñoz, A. Narkawicz, G. Dowek, How formal methods impels discovery: A short history of an air traf-

fic management project, in: C. Muñoz (Ed.), Proceedings of the Second NASA Formal Methods Symposium (NFM 2010), NASA/CP-2010-216215, NASA, Langley Research Center, Hampton VA 23681-2199, USA, 2010, pp. 34–46.

[2] J. Maddalon, R. Butler, C. Muñoz, G. 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, 2009.

[3] J. Maddalon, R. Butler, C. Muñoz, G. Dowek, Mathematical basis for the safety analysis of conflict prevention algorithms, Technical Memorandum NASA/TM-2009-215768, NASA, Langley Research Center, Hampton VA 23681-2199, USA (June 2009).

[4] J. M. Hoekstra, Designing for safety: The free flight air traffic management concept, Tech. Rep. 90-806343-2-8, Technische Universiteir Delft (Nov. 2001).

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

[6] A. Narkawicz, C. Muñoz, G. Dowek, Formal verification of air traffic prevention bands algorithms, Technical Memorandum NASA/TM-2010-216706, NASA, Langley Research Center, Hampton VA 23681-2199, USA (June 2010).

[7] S. Owre, J. Rushby, N. Shankar, PVS: A prototype verification system, in: D. Kapur (Ed.), Proceeding of the 11th International Conference on Automated Deductioncade, Vol. 607 of Lecture Notes in Artificial Intelligence, Springer, 1992, pp. 748–752.

[8] W. Rudin, Principles of Mathematical Analysis, 3rd Edition, McGraw-Hill, 1976.

[9] J. Maddalon, R. Butler, A. Geser, C. Muñoz, Formal verification of a conflict resolution and recovery algorithm, Tech. Rep. NASA/TP-2004-213015, NASA Langley Research Center, NASA LaRC,Hampton VA 23681-2199, USA (April 2004).

[10] D. A. Karr, D. A. Roscoe, R. A. Vivona, An integrated flight-deck decision-support tool in an autonomous flight simulation, in: Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit, no. AIAA 2004-5261, Providence, Rhode Island, 2004.

[11] F. J. L. Bussink, J. M. Hoekstra, B. W. Heesbeen, Traffic manager: A flexible desktop simulation tool enabling future ATM research, in: Proceedings of the 24th Digital Avionics Systems Conference, DASC 2005, Washington D.C., 2005.

[12] D. N. Sweet, V. Manikonda, J. S. Aronson, K. Roth, M. Blake, Fast-time simulation system for analysis of advanced air transportation concepts, in: Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit, no. AIAA 2002-4593, Monterey, CA, USA, 2002.

[13] D. Bushnell, D. Giannakopoulou, P. Mehlitz, R. Paielli, C. Pasareanu, Verification and validation of air traffic systems: Tactical separation assurance, in: Proceedings of the IEEE Aerospace Conference, Big Sky, Montana, USA, 2009.

[14] A. Platzer, E. M. Clarke, Formal verification of curved flight collision avoidance maneuvers: A case study, in: FM, Vol. 5850 of LNCS, Springer, 2009, pp. 547–562.

Appendix A. Summary of PVS Development

```
% This file summarizes all the results presented in the paper
% "Provably Correct Conict Prevention Bands Algorithms" by
% Anthony Narkawicz, Cesar Munoz, and Gilles Dowek submitted to
% Elsevier's Science of Computer Programming.

SCP[D,H,gsmin,gsmax:posreal,vsmin,vsmax:real,T:posreal] : THEORY
BEGIN
  ASSUMING
    gs_min_lt_max: ASSUMPTION gsmin < gsmax
    vs_min_lt_max: ASSUMPTION vsmin < vsmax
  ENDASSUMING
  IMPORTING bands[D,H,gsmin,gsmax,vsmin,vsmax]

  ilow,
  ihigh   : VAR real
```

```
s       : VAR Sp_vect3
ss,w    : VAR Vect3
vo,vi   : VAR Nzv2_vect3
trkb    : VAR (trk_fseq?)
gsb     : VAR (gs_fseq?)
trk     : VAR nnreal_lt_2pi
trk2    : VAR r: real | 0<=r AND r<=2*pi
gs      : VAR r: real | gsmin<=r AND r<=gsmax
vs      : VAR r: real | vsmin<=r AND r<=vsmax
vnu     : VAR [Nzv2_vect3->[real->Nzv2_vect3]]
L       : VAR [[Sp_vect3,posreal,Nzv2_vect3,Nzv2_vect3]->fseq]
cdalg   : VAR [[Sp_vect3,Nzv2_vect3,Nzv2_vect3]->bool]
A,B,x   : VAR real

contin_fun?(f:[real->real]) : bool =
  analysis@continuity_ms_def[reals.real,
   analysis@real_metric_space.real_dist,
   reals.real,
   analysis@real_metric_space.real_dist].continuous?(f)

% Conflict Detection Algorithms.
CdAlgorithmCorrect?(cdalg): bool =
  FORALL (s,vo,vi): conflict_3D?(s,T,vo-vi) IMPLIES cdalg(s,vo,vi)

CdAlgorithmComplete?(cdalg): bool =
  FORALL (s,vo,vi): cdalg(s,vo,vi) IMPLIES conflict_3D?(s,T,vo-vi)

% Definition 1.
CdAlgorithmCorrectAndComplete?(cdalg): bool =
  CdAlgorithmCorrect?(cdalg) and CdAlgorithmComplete?(cdalg)

% A correct conflict detection (CD) algorithm.
cd: VAR (CdAlgorithmCorrectAndComplete?)

% This theorem states that a correct CD algorithm exists.
CorrectCompleteCdAlgExists: LEMMA
  EXISTS (cdalg): CdAlgorithmCorrectAndComplete?(cdalg)

Iclosed(ilow,ihigh) : set[real] = r:real | ilow<=r AND r<=ihigh
Iopen(ilow,ihigh)   : set[real] = r:real | ilow< r AND r< ihigh

% Prevention Bands Algorithm.
PrevBandsAlgorithm?(ilow,ihigh)(L) : bool =
  FORALL (s,vo,vi): LET Lnu = L(s,T,vo,vi) IN
    Lnu`length >= 2 AND increasing?(Lnu) AND
```

```
    (FORALL (i:below(Lnu`length)): Iclosed(ilow,ihigh)(Lnu`seq(i)))
    AND Lnu`seq(0) = ilow AND Lnu`seq(Lnu`length-1) = ihigh

% Equation 6.
conflict_band?(cd,vnu)(s,vo,vi,A,B) : bool =
  cd(s,vnu(vo)((A+B)/2),vi)

% Definition 2.
BandsAlgCorrect?(cd,vnu,ilow,ihigh)
                (L:(PrevBandsAlgorithm?(ilow,ihigh))): bool =
  FORALL (s,vo,vi): LET Lnu = L(s,T,vo,vi) IN
  FORALL (i:below(Lnu`length-1)): LET A=Lnu`seq(i),B=Lnu`seq(i+1) IN
    A<B IMPLIES
    ((conflict_band?(cd,vnu)(s,vo,vi,A,B) IFF
     (FORALL (y:(Iopen(A,B))): conflict_3D?(s,T,vnu(vo)(y)-vi)))
    AND
    ((NOT conflict_band?(cd,vnu)(s,vo,vi,A,B)) IFF
     (FORALL (y:(Iopen(A,B))): NOT conflict_3D?(s,T,vnu(vo)(y)-vi))))

% Classification Functions.
OmegaNu : VAR [[Sp_vect3,Nzv2_vect3]->[real->real]]

% Definition 3.
ClassFun?(vnu)(OmegaNu): bool = FORALL (s,vo,vi,x):
  contin_fun?(OmegaNu(s,vi)) AND (OmegaNu(s,vi)(x) < 1 IFF
    conflict_3D?(s,T,vnu(vo)(x)-vi))

% Definition 4.
SeqComplete?(OmegaNu,ilow,ihigh)(L): bool =
  FORALL (s,vo,vi)(x:(Iclosed(ilow,ihigh))):
    OmegaNu(s,vi)(x) = 1 IMPLIES member(x,L(s,T,vo,vi))

Theorem1: THEOREM ilow < ihigh AND
  PrevBandsAlgorithm?(ilow,ihigh)(L) AND
  ClassFun?(vnu)(OmegaNu) AND SeqComplete?(OmegaNu,ilow,ihigh)(L)
  IMPLIES
    BandsAlgCorrect?(cd,vnu,ilow,ihigh)(L)

% Definition 5.
cyl_length(w) : nnreal = max(norm(vect2(w))/D,abs(w`z)/H)

% Definition 6.
cyl_dist(w1,w2:Vect3): nnreal = cyl_length(w1-w2)

Theorem2: THEOREM FORALL (ss:Vect3): los?(ss) IFF cyl_length(ss) < 1
```

```
% Definition of Omega (Equation 12).
Omega_fun(ss)(w) : real =
  IF on_H?(ss`z) AND on_D?(ss) THEN
    max(ss`z*w`z,vect2(ss)*vect2(w))+1
  ELSIF vertical_los?(ss`z) AND on_D?(ss) THEN
    vect2(ss)*vect2(w)+1
  ELSIF on_H?(ss`z) AND horizontal_los?(ss) THEN
    ss`z*w`z+1
  ELSE
    inf(LAMBDA(t: Lookahead(T)):cyl_length(ss+t*w))
  ENDIF

Omega_fun_3D_lt: LEMMA Omega_fun(ss)(w) < 1 IFF omega_v3(ss,T)(w) < 0

Omega_fun_3D_gt: LEMMA Omega_fun(ss)(w) > 1 IFF omega_v3(ss,T)(w) > 0

Omega_fun_3D_eq: LEMMA Omega_fun(ss)(w) = 1 IFF omega_v3(ss,T)(w) = 0

Theorem3: THEOREM FORALL (ss,w:Vect3): conflict_3D?(ss,T,w) IFF
  Omega_fun(ss)(w) < 1

% Theorem 4 is in the vect_analysis directory, in the file
% vect3_Heine.pvs. The lemma there is called curried_min_is_cont_3D.

Theorem5: THEOREM continuous?(Omega_fun(ss))

% Given vnu, the next function gives Omnu (Equation 10).

Omnu(ss,vo,vi)(vnu)(x:real) : real = Omega_fun(ss)(vnu(vo)(x)-vi)

Corollary6: COROLLARY Omnu(ss,vo,vi)(vnu)(x) < 1 IFF
  conflict_3D?(ss,T,vnu(vo)(x)-vi)

Corollary7: COROLLARY continuous?(vnu(vo)) IMPLIES
  contin_fun?(Omnu(ss,vo,vi)(vnu))

% The predicates line_case?, circle_case_2D?, circle_case_3D?,
% and vertical_case? in the paper correspond, respectively, to
% the following predicates in the PVS development:
% line_solution?, circle_solution_2D?, circle_solution?, and
% vertical_solution?. They are found in the files
% line_solutions.pvs, horizontal.pvs, circle_solutions.pvs,
% and vertical.pvs, respectively.
```

```
  % The classification theorem for critical vectors.
  Theorem8: THEOREM  FORALL (v:Vect3): Omega_fun(ss)(v) = 1 IMPLIES
    horizontal_sep?(ss) AND line_solution?(ss,v) OR
    vertical_los?(ss`z+T*v`z) AND
    circle_solution_2D?(ss,v,T,Entry) OR
    circle_solution?(ss,v) OR
    NOT strict_horizontal_sep?(ss+T*v) AND
    vertical_solution?(ss`z,v`z,T,Entry)

  % Lemma 9 corresponds to lemma line_solution_tangent_line in
  % the file tangent_line.pvs.
  % Lemma 10 follows from lemma trk_only_line_complete,
  % which is found in the file trk_only.pvs.
  % Theorem 11 follows from lemmas called trk_line_is_line_solution
  % and trk_line_complete, which are found in the file trk_line.pvs.
  % Lemma 12 follows from lemma trk_only_dot_complete, as well from
  % the definition of the function trk_only_dot, both of which are
  % found in the file trk_only.pvs.
  % Theorems 13 and 14 correspond to lemmas trk_only_circle_solution
  % and trk_only_circle_complete in the file trk_only.pvs.
  % Theorems 15 and 16 correspond to lemmas trk_circle_solution and
  % trk_circle_complete in the file trk_circle.pvs.
  % The algorithm trk_bands, which is called track_bands in the paper,
  % is defined in the file bands.pvs.
  % The function nu_trk in the paper, which appears in the
  % statement of Theorem 17, is called trk2v3 in the PVS development,
  % and it is defined in the file trk_bands_3D.pvs.

  Theorem17: THEOREM BandsAlgCorrect?(cd,trk2v3,0,2*pi)(trk_bands)
END SCP
```