NASA/TM-2008-215356

A Formal Framework for the Analysis of Algorithms That Recover From Loss of Separation

Ricky W. Butler
Langley Research Center, Hampton, Virginia

César A. Muñoz
National Institute of Aerospace, Hampton, Virginia

October 2008

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Phone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076-1320

NASA/TM-2008-215356

A Formal Framework for the Analysis of Algorithms That Recover From Loss of Separation

Ricky W. Butler
Langley Research Center, Hampton, Virginia

César A. Muñoz
National Institute of Aerospace, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

October 2008

Available from:

NASA Center for AeroSpace Information (CASI)
7115 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 605-6000

Abstract

We present a mathematical framework for the specification and verification of state-based conflict resolution algorithms that recover from loss of separation. In particular, we propose rigorous definitions of horizontal and vertical *maneuver correctness* that yield horizontal and vertical separation, respectively, in a bounded amount of time. We also provide sufficient conditions for independent correctness, i.e., separation under the assumption that only one aircraft maneuvers, and for implicitly coordinated correctness, i.e., separation under the assumption that both aircraft maneuver. An important benefit of this approach is that different aircraft can execute different algorithms and implicit coordination will still be achieved, as long as they all meet the explicit criteria of the framework. Towards this end we have sought to make the criteria as general as possible. The framework presented in this paper has been formalized and mechanically verified in the Prototype Verification System (PVS).

Contents

1 Introduction

This work is motivated by some recent TMX [1] studies of the KB3D [3, 4] Conflict Detection and Resolution (CD&R) algorithm. These studies explored the capabilities of KB3D to deal with multiple aircraft in complex traffic situations. The traffic density was approximately three times that of today's traffic and was generated by extrapolation from existing traffic patterns. There were almost no situations where a loss of separation occurred, but for the few cases where it did occur, it became clear that the algorithm should be generalized to recover from those situations.

In this paper, we present a mathematical framework for the specification and verification of state-based conflict resolution algorithms that recover from loss of separation. In particular, the framework provides:

- Rigorous definitions of horizontal and vertical *maneuver correctness* that yield horizontal and vertical separation, respectively, in a bounded amount of time.
- Sufficient conditions for maneuvers to be independently correct, e.g., separation is achieved under the assumption that only one aircraft maneuvers, and coordinately correct, e.g., separation is achieved under the assumption that both aircraft maneuver, but without hand-shaking or explicit information exchange.

The techniques developed in this paper will apply to any state-based algorithm used to recover from loss of separation and do not depend upon the use of the KB3D algorithm. It is expected that verification methods developed here will facilitate the proof of correctness of many different kinds of algorithms.

The framework presented in this paper has been formalized and mechanically verified in the Prototype Verification System (PVS) [6] and is electronically available from `http://research.nianet.org/fm-at-nia/KB3D`.

2 Basic Concepts

As typical of state-based approaches, our framework is centered around the idea of modeling aircraft trajectories as linear functions of time into a 3-dimensional vector space with coordinates `x`, `y`, and `z`. In PVS, we define the type of 3-dimensional vectors in a Cartesian coordinate system as follows

```
Vect3: TYPE = [#  x, y, z: real  #]
```

The components of a vector `v` are referenced using the back-quote operator, e.g., `v`x`, `v`y`, and `v`z`.

The standard operations on vectors are defined: if `u` and `v` are vectors and `a` is a scalar, `u + v`, `-u`, `u - v`, `u * v`, and `a*v` denote addition, negation, subtraction, dot product, and scalar multiplication, respectively. We also define

```
sq(v)   : nnreal = v*v
norm(v) : nnreal = sqrt(sq(v))
```

where `nnreal` is the type of non-negative real numbers.

The framework is concerned with two aircraft. We will refer to one as the *ownship* and the other as the *traffic aircraft*. The position and velocity vectors of the ownship and traffic aircraft are denoted `so,vo` and `si,vi`, respectively. For some definitions, it is convenient to use a relative coordinate system where the traffic aircraft is located at the origin of the system and is motionless. The relative position and velocity vectors of the ownship are denoted `s` and `v`, respectively, where `s = so-si` and `v=vo-vi`. Furthermore, new velocity vectors for the ownship and traffic aircraft are denoted `nvo` and `nvi`, respectively.

For clarity, we sometimes use standard mathematical notation instead of PVS code. In this case, vector variables are written in boldface, e.g., $\mathbf{v}$, and their components are referenced by sub-indices, e.g., v_x, v_y, and v_z. Position and velocity vectors for the ownship are denoted $\mathbf{s_o}$ and $\mathbf{v}_o$, respectively. Traffic vectors are indexed by i, e.g., $\mathbf{s_i}$ and $\mathbf{v}_i$, and new velocity vectors are denoted by primed variables, e.g., $\mathbf{v}'_o$ and $\mathbf{v}'_i$.

Within the translated frame of reference, some concepts can be elegantly defined. For instance, if `D` and `H` are, respectively, the diameter and height of the protected zone around each aircraft, the predicates that test if the aircraft are horizontally or vertically separated are conveniently defined in the relative coordinate system as follows

```
horizontal_separation?(s):bool =
  sq(s`x)+sq(s`y) >= sq(D)

vertical_separation?(s):bool =
  abs(s`z) >= H

separation?(s) : bool =
  horizontal_separation?(s) OR vertical_separation?(s)
```

From these predicates, we define *loss of separation* as follows

```
loss_of_separation?(s) : bool = NOT separation?(s)
```

Therefore, the fact that the ownship and traffic aircraft have lost separation can be simply expressed as `loss_of_separation?(so-si)`.

3 Correct Maneuvers for Loss of Separation Recovery

We are concerned with the situation where a loss of separation has already occurred. Many conflict detection and resolution algorithms do not address this situation. They are developed with the specific goal of detecting a conflict and recovering before there is a loss of separation. Nevertheless, it is prudent that systems based on these algorithms be able to provide outputs when they find themselves in a state they were designed to prevent. It should be noted the KB3D algorithm proofs do not currently cover the fully general case of multiple aircraft conflicts, and hence it

is theoretically possible for this undesirable situation to arise in practice. Simulation studies of KB3D at NASA Langley indicate that this will be a rare event even in complex traffic scenarios [2].

In this paper we describe a framework for reasoning about algorithmic solutions to the loss of separation situation. We propose a comprehensive approach to this problem based on two components:

- A formal definition of the concept of *maneuver correctness* that yields horizontal or vertical separation.
- A set of simple *conditions* or *criteria* that are easily calculated and that guarantee independent and coordinated correctness.

We prove that any algorithm that produces loss of separation recovery maneuvers, which satisfy the criteria, is correct.

An important benefit of this approach is that implicit coordination can be achieved without having every aircraft execute the same algorithm. Different airlines can use different algorithms and there still will be safe coordination in the airspace, as long as these algorithms satisfy the criteria. The criteria embody the "rules of the road". Therefore, it is important that the criteria be as general as possible. We do not want to unnecessarily rule out any good algorithm. In this work, we have sought to make the criteria as general as we could, but we have not offered a proof that the criteria are necessary as well as sufficient.

In a loss of separation situation, the protected zones of the ownship and traffic aircraft overlap. Intuitively, a correct algorithm that recovers from this situation should eventually achieve separation. In PVS, we can write this condition as follows

```
EXISTS (t: posreal): separation?(s+v*t)
```

In other words, there exists a time in the future, i.e., `t > 0`, where the aircraft are separated.

But there are two problems with this as a notion of correctness. Almost all trajectories (except parallel trajectories) eventually lead to this condition. In fact, they could result in a collision in the process. The second problem is that the time to reach separation may be extraordinarily long, e.g., when the paths are nearly parallel. So we need to augment our definition of correctness such that correct recovery maneuvers do not make things worse and achieve recovery in a bounded amount of time. To solve the first problem, we must ensure that aircraft do not get any closer. To solve the second problem, we must consider the time when aircraft recover from loss of separation.

State-based CD&R algorithms typically decompose the 3D airspace into a horizontal 2-dimensional xy-perspective and a vertical z-perspective. We follow the same approach and study the horizontal and vertical cases independently. Because position and velocity vectors are initially given in a 3D coordinate system, we use the function `vect2D` to convert from 3D to 2D vectors:

```
vect2D(v:Vect3): Vect2 = (v`x,v`y)
```

where `Vect2` is the type of 2D vectors:

```
Vect2: TYPE = [#  x, y: real  #]
```

3.1 Horizontal Maneuver Correctness

In order to express the property that aircraft that have lost separation do not get any closer in the horizontal plane, we define a notion of horizontal divergence in the relative coordinate system:

```
xy_divergent?(s,v): bool =
  FORALL (t: posreal): norm(vect2D(s)) < norm(vect2D(s+t*v))
```

We remark that the 2-dimensional norm of the relative position `s=so-si` is the horizontal distance between the ownship and traffic aircraft at time 0. Therefore, this predicate states that the horizontal distance between the aircraft at any time in the future `t` is greater than the horizontal distance at the current time. Furthermore, we note that `xy_divergent?` is equivalent to the seemingly more general predicate:[1]

```
FORALL (t1,t2: posreal): t1 <= t2 IMPLIES
  norm(vect2D(s+t1*v)) < norm(vect2D(s+t2*v))
```

We assume that there will be an operational constraint that provides a maximum time for recovery from the horizontal loss of separation condition. We call this configurable parameter `Th`. Thus, at time `Th`, the aircraft will be located at `s+Th*v`, which should be outside of the protection zone, i.e., `horizontal_separation?(s+Th*v)` should hold.

Therefore, we say that the ownship's velocity vector `vo` is *horizontally correct* with respect to the relative position `s` and the traffic's velocity vector `vi` if and only if

- `xy_divergent?(s,vo-vi)`, and
- `horizontal_separation?(s+Th*(vo-vi))`.

In PVS, we define the predicate `xy_correct?` as follows:

```
xy_correct?[Th](s,vi)(vo): bool =
  xy_divergent?(s,vo-vi) AND
  horizontal_separation?(s+Th*(vo-vi))
```

We remark that the predicate `xy_correct` uses a curryfied list of parameters where `s` is a relative position, and `vo` and `vi` are the untranslated ownship and traffic velocity vectors, respectively. Furthermore, `vo` is not grouped together with `s` and `vi`. This is a matter of style, but here we want to emphasize the fact that `xy_correct?(s,vi)` is a correctness predicate for `vo`. The parameter `Th` is in brackets because it is a parameter to an entire PVS theory.

[1] If the velocity vectors were generalized to non-constant functions of time, then a definition similar to this would be needed.

3.2 Vertical Maneuver Correctness

The concept of vertical divergence is one dimensional, and can be defined in the relative coordinate system as follows

```
z_divergent?(s,v): bool =
      FORALL (t: posreal): abs(s`z) < abs(s`z+t*v`z)
```

We also assume that there will be an operational constraint that provides a maximum recovery time. We will call this configurable parameter `Tv`. This would lead to the following definition of *vertical correctness*:

```
z_correct?[Tv](s,vi)(vo): bool =
  z_divergent?(s,vo-vi) AND
  vertical_separation?(s+Tv*(vo-vi))
```

As in the case of `xy_correct?`, the predicate `z_correct?` uses a curryfied list of parameters where `s` is a relative position, and `vo` and `vi` are the untranslated ownship and traffic velocity vectors, respectively. The parameter `Tv` is in brackets because it is a parameter to an entire PVS theory.

4 Time to Exit

Before defining the criteria for loss of separation recovery maneuvers, we introduce a function that computes the time to exit from horizontal and vertical loss of separation.

4.1 Horizontal

We would like to be able to select from the set of divergent maneuvers those which are most efficient. To do this we need to be able to compute the time to exit the protected zone. Towards this end, we introduce a function `tteh` which computes the horizontal exit time:

```
tteh(s: (xy_loss?), v:(gs?)): real = THETA(s`x,s`y,v`x,v`y,1)
```

where `THETA` is defined in PVS as follows

```
THETA(sx,sy,vx:real,vy:real|Delta_ge_0?(sx,sy,vx,vy), eps:Sign):real =
  (-sx*vx - sy*vy + eps * sqrt(Delta(sx,sy,vx,vy))) / (sq(vx)+sq(vy))

Delta(sx,sy,vx,vy) : real =
  sq(D) * (sq(vx) + sq(vy)) - sq(sx * vy - sy * vx)
```

where `Sign` returns values of 1 or -1:

```
Sign : TYPE = {i:int|i=1 OR i=-1}
```

and `Delta_ge_0?` is a predicate subtype that guarantees that `THETA` is a well-defined function, i.e., the square root exists and the division is not zero. Mathematically, the definition of `tteh` corresponds to

$$\texttt{tteh}(\mathbf{s},\mathbf{v}) \quad = \quad -s_x v_x - s_y v_y \pm \frac{\sqrt{\Delta(s_x, s_y, v_x, v_y)}}{v_x^2 + v_y^2}$$

where

$$\Delta(s_x, s_y, v_x, v_y) \quad = \quad D^2(v_x^2 + v_y^2) - (s_x v_y - s_y v_x)^2.$$

and the double sign $\pm$ is provided formally by the `eps` argument. Note that the arguments to `tteh` are constrained by the following predicates:

```
xy_loss?(s): bool = NOT horizontal_separation?(s)

gs?(v)     : bool = sq(v`x) + sq(v`y) /= 0
```

which restrict the use of `tteh` to situations where a loss of separation has occurred and to where the velocity vector is not zero in the xy-plane.

The key property about `tteh` is that at time $t_e =$ `tteh(s,v)`, the aircraft will be on the circle of radius `D`, i.e.,

$$(s_x + v_x t_e)^2 + (s_y + v_y t_e)^2 = D^2$$

or more formally[2]:

Lemma (`tteh_sq_D`).

```
sq(v`x) + sq(v`y) /= 0 AND
NOT separation?(s) IMPLIES
  sq(s`x + v`x*tteh(s,v)) + sq(s`y + v`y*tteh(s,v)) = sq(D)
```

Proof. The location of the aircraft at time t is $s + tv$. This line intersects the circle of radius `D` where

$$(s_x + t v_x)^2 + (s_y + t v_y)^2 = D^2$$

Expanding and collecting terms yields

$$(v_x^2 + v_y^2)\, t^2 + 2(s_x v_x + s_y v_y)\, t + s_x^2 + s_y^2 - D^2 = 0.$$

This is a quadratic equation $(at^2 + bt + c)$ in t with $a = (v_x^2 + v_y^2)$, $b = 2(s_x v_x + s_y v_y)$ and $c = s_x^2 + s_y^2 - D^2$. The quadratic formula provides a positive solution which is precisely `tteh(s,v)`. □

[2]In PVS, free variables are universally quantified.

4.2 Vertical

We would need to be able to select from the set of vertically divergent maneuvers those which are most efficient. To do this we need to be able to compute the time to exit the protected zone vertically. We introduce a function `ttez`, which computes the vertical exit time:

```
ttez(s:Vect3, v:(vnz?): real = (sign(v`z) * H - s`z) / v`z
```

where `vnz?(v):bool = v`z /= 0` and `sign` is the two-valued sign function:

```
sign(x:real) : Sign =
  IF x >= 0 THEN 1
  ELSE -1
  ENDIF
```

The following lemma characterizes the function `ttez`:

Lemma (`z_vnz_separation`)**.**

```
v`z /= 0 AND
NOT vertical_separation?(s) AND
tr >= ttez(s,v) IMPLIES
  vertical_separation?(s + tr * v)
```

Proof. Expanding the definition of `ttez(s,v)` and cross-multiplying yields:

- Case $v_z > 0$: $s_z + t_r v_z \geq H$.
- Case $v_z < 0$: $s_z + t_r v_z \leq -H$.

Both cases can be combined into $|s_z + t_r v_z| \geq H$, which is vertical separation. □

5 Loss of Separation Recovery Criteria

The goal of the framework is to establish some simple abstract properties i.e. criteria that

- are sufficient to prove correctness, and
- are easy to verify for a specific set of horizontal and vertical maneuvers.

5.1 Horizontal Maneuver Criteria

In this section, we will assume that all vectors are given in a 2D coordinate system. A 2D wrapper for 3D vectors is easily defined using `vect2D`.

The horizontal criteria is built around a simple predicate called `dot_prop`:

```
dot_prop?(s,v): bool = s * v >=0
```

This predicate is defined in terms of two parameters `s` and `v` of the translated frame of reference. The idea for this predicate comes from the observation that a good relative maneuver $\mathbf{v}$ is one where the angle between a head-on vector $-\mathbf{s}$ and $\mathbf{v}$ is in the range $[\frac{\pi}{2}, \frac{3\pi}{2}]$. This occurs where the dot product is not positive: $-\mathbf{s} \cdot \mathbf{v} \leq 0$ or more simply when $\mathbf{s} \cdot \mathbf{v} \geq 0$.

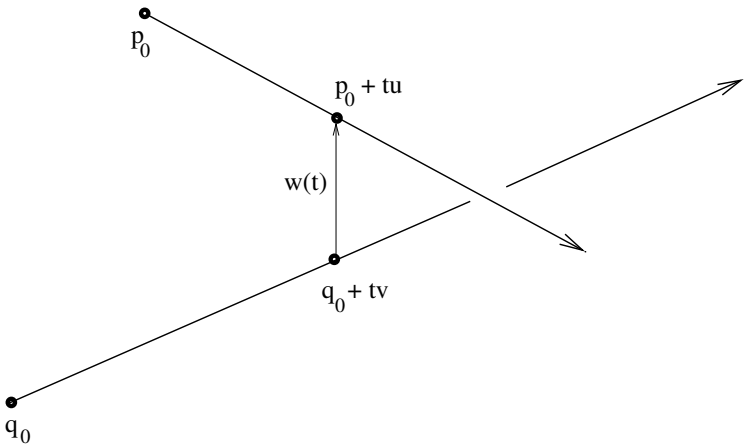
This property alone is enough to establish that aircraft will be on divergent paths: `dot_prop?(s,v)` implies `xy_divergent?(s,v)`. We will prove this theorem by first establishing the following lemma:

```
dot_nneg_tca_npos: LEMMA
  dot_prop?(so-si,vo-vi)
  IMPLIES
    time_closest(so,si,vo,vi) <= 0
```

where the time of closest approach `time_closest(so,si,vo,vi)`, denoted in mathematical notation by τ, is defined as follows

```
time_closest(p0,q0,u,v): real = IF norm(u-v) = 0 THEN
                                   0
                                ELSE
                                   -((p0-q0)*(u-v))/sq(norm(u-v))
                                ENDIF
```

Here `p0` and `q0` represent the location of the aircraft at time 0. The trajectory of the first is given by $p(t) = p_0 + tu$ and the trajectory of the second is given by $q(t) = q_0 + tv$.



Mathematically, for the non-parallel case, we have

$$\tau = \frac{-(\mathbf{p}_0 - \mathbf{q}_0) \cdot (\mathbf{u} - \mathbf{v})}{|\mathbf{u} - \mathbf{v}|^2}.$$

To see that this function indeed computes the time at which two moving particles achieve the minimum distance between them, we prove the following lemma:

Lemma (`time_cpa`).

```
t_cpa = time_closest(p0,q0,u,v)
IMPLIES
  is_minimum?(t_cpa,(LAMBDA t: sq_dist(p0+t*u,q0+t*v)))
```

Proof. The distance between the particles at time t is given by

$$d(t) = |\mathbf{p}(t) - \mathbf{q}(t)| = |\mathbf{w}(t)|, \tag{1}$$

where $\mathbf{w}(t) = \mathbf{w}_0 + t(\mathbf{u} - \mathbf{v})$ and $\mathbf{w}_0 = \mathbf{p}_0 - \mathbf{q}_0$. The distance $d(t)$ achieves a minimum where $d^2(t)$ is a minimum, so we can work with the square of the distance:

$$d^2(t) = \mathbf{w}(t) \cdot \mathbf{w}(t) = (\mathbf{u} - \mathbf{v}) \cdot (\mathbf{u} - \mathbf{v})\, t^2 + 2\mathbf{w}_0 \cdot (\mathbf{u} - \mathbf{v})\, t + \mathbf{w}_0 \cdot \mathbf{w}_0. \tag{2}$$

This function achieves a minimum when its derivative is 0:

$$\frac{d}{dt}[d^2(t)] \;=\; 2t[(\mathbf{u} - \mathbf{v}) \cdot (\mathbf{u} - \mathbf{v})] + 2\mathbf{w}_0 \cdot (\mathbf{u} - \mathbf{v}) \;=\; 0.$$

Solving for t we get

$$t \;=\; \frac{-\mathbf{w}_0 \cdot (\mathbf{u} - \mathbf{v})}{|\mathbf{u} - \mathbf{v}|^2},$$

which is `time_closest(p0,q0,u,v)`. This solution is valid if $|\mathbf{u} - \mathbf{v}|$ is not zero. If it is zero, then the lines are parallel. In this case the function `time_closest(p0,q0,u,v)` just returns 0, which is always less than or equal to the square distance. □

Next, we show that when this property holds, the time of closest approach is negative, i.e., it is in the past:

Lemma (`dot_nneg_tca_npos`).

```
dot_prop?(so-si,vo-vi)
IMPLIES
  time_closest(so,si,vo,vi) <= 0
```

Proof. The assumption `dot_prop?(so-si,vo-vi)` simplifies to

$$(\mathbf{s}_o - \mathbf{s}_i) \cdot (\mathbf{v}_o - \mathbf{v}_i) \geq 0$$

If $\mathbf{v}_o = \mathbf{v}_i$ then `time_closest(so,si,vo,vi) = 0` and we are done. Otherwise, `time_closest(so,si,vo,vi)` is

$$-\frac{(\mathbf{s}_o - \mathbf{s}_i) \cdot (\mathbf{v}_o - \mathbf{v}_i)}{|\mathbf{v}_o - \mathbf{v}_i|^2},$$

which, by the premise, must be less than or equal to zero. □

This leads us to the main result, that when `dot_prop?` is true, the aircraft trajectories are divergent:

Theorem (dot_prop_divergent)**.**

```
v /= zero AND
dot_prop?(s,v) IMPLIES
  xy_divergent?(s,v)
```

Proof. Equation 2 reveals that the square of the distance is a quadratic equation in t. Since the leading coefficient is positive, the parabola achieves a minimum. Furthermore, the square distance function is monotonically decreasing before the minimum point and monotonically increasing after the minimum point. From the lemma `dot_nneg_tca_npos` we have `time_closest(s,0,v,0) <= 0`. Therefore, for all $t > 0$, the distance is monotonically increasing, which is the definition of `xy_divergent(s,v)`. □

5.1.1 The Formal xy-Criteria

We introduce the following criteria for horizontal maneuvers:

```
criteria?(s,vo,vi)(nvo): bool =
  nvo /= vi AND
  dot_prop?(s,nvo-vi) AND
  (dot_prop?(s,vo-vi) IMPLIES
    (vo /= vi  AND
     dot_prop?(s,nvo-vo) OR
     vo = vi AND
     s*(nvo-vo) > 0))
```

where `s`, `vo`, and `vi` are 2D-vectors. We lift this predicate to a 3D version as follows:

```
xy_criteria?(s,vo,vi: Vect3)(nvo: Vect3): MACRO bool =
   criteria?(vect2D(s),vect2D(vo),vect2D(vi))(vect2D(nvo))
```

We need to augment the criteria with a constraint on the time to exit horizontally. Thus we have:

```
xy_criteria_tr?(s,vo,vi,tr)(nvo): bool =
  NOT horizontal_separation?(s) AND
  xy_criteria?(s,vo,vi)(nvo) AND
  tteh(s,nvo-vi) <= tr
```

The horizontal criteria thus require that

- The new ownship's velocity vector $\mathbf{v}'_o$ must be different from intruder's initial velocity vector $\mathbf{v}_i$.
- The property `dot_prop?`$(\mathbf{s}, \mathbf{v}'_o - \mathbf{v}_i)$ holds, which guarantees that the new ownship velocity vector $\mathbf{v}'_o$ is divergent with respect the initial intruder velocity vector $\mathbf{v}_i$.

- If the initial situation is one where the aircraft are already divergent, then we also must have `dot_prop`$(\mathbf{s}, \mathbf{v}'_o - \mathbf{v}_o)$ which further restricts the allowed maneuvers.
- In the special case where $\mathbf{v}_o = \mathbf{v}_i$, we require that the dot product be strictly greater than 0.

5.1.2 Some Rationale and Observations

The `xy_criteria?` at first appears to be more complicated than necessary. In fact, it is far more complicated than is necessary for the independent correctness theorem. In later sections, we will prove that this criteria enables the proof of cooperative correctness.

To establish divergence for the cooperative theorem, we need to prove `xy_divergent?(s,nvo-nvi)`. From the `dot_prop_divergent` theorem we know that we only need to establish `dot_prop?(s,nvo-nvi)`. The converse is also true. So the first question that arises is whether the two premises

```
dot_prop?(s,nvo-vi) AND
dot_prop?(-s,nvi-vo)
```

are sufficient to give us `dot_prop?(s,nvo-nvi)`. In other words if each aircraft independently ensures that its own new velocity vector is `dot_prop?` with respect to the other aircraft's unchanged velocity, is that sufficient to guarantee that the combined activity is divergent? Surprisingly this is the case when the aircraft are initially convergent, but if they are already divergent, these conditions are not adequate. When the aircraft are initially divergent, i.e., `dot_prop?(s,vo-vi)`, we also need `dot_prop?(s,nvo-vo)`.

5.2 Vertical Maneuver Criteria

In this section, we will assume that all vectors are given in a 3D coordinate system. Using a similar approach as the horizontal maneuver criteria, we define the vertical maneuver criteria as follows:

```
z_criteria?(s,vo,vi)(nvo): bool =
  nvo`x = vo`x AND nvo`y = vo`y
  (nvo-vi)`z /= 0 AND
  z_prop?(s,nvo-vi) AND
  (z_prop?(s,vo-vi) IMPLIES
    (vo-vi)`z /= 0 AND
      sign((vo-vi)`z)*(nvo-vo)`z >= 0 OR
    (vo-vi)`z = 0 AND
      break_vz_symm(s)*(nvo-vo)`z > 0)
```

where `z_prop?` is defined as

```
z_prop?(s,v): bool = s`z * v`z >=0
```

and `sign` is the two-valued sign function:

```
sign(x:real): Sign =
  IF x >= 0 THEN 1 ELSE -1 ENDIF
```

As in the horizontal case, we augment the criteria with a constraint on the time to exit vertically:

```
z_criteria_tr?(s,vo,vi,tr)(nvo): bool =
  NOT vertical_separation?(s) AND
  z_criteria?(s,vo,vi)(nvo) AND
  ttez(s,nvo-vi) <= tr
```

The `z_criteria?(s,vo,vi)(nvo)` predicate is sufficient to establish divergence in both the independent and coordinated correctness theorems. The premise `nvo`x = vo`x AND nvo`y = vo`y` states that `nvo` is a vertical maneuver, i.e., `nvo` may differ from `vo` only in the vertical component. The additional premise `ttez(s,nvo-vi) <= tr` is necessary to establish that the time to recover is sufficiently small. We will prove in a subsequent section that even though each aircraft calculates a new velocity vector using the original velocity vector of the other aircraft, that together they still diverge and meet the timeliness criteria.

As in the horizontal case, we need to consider a few special cases when the aircraft are originally diverging, i.e., `z_prop?(s,vo-vi)`:

- If the relative vertical speed is not zero, i.e., `vo`z /= vi`z`, we require `sign((vo-vi)`z)*(nvo-vo)`z >= 0`.
- Otherwise, i.e., `vo`z = vi`z`, we require `break_vz_symm(s)*(nvo-vo)`z > 0`.

In the latter case, we have to deal with the situation where the two aircraft are at exactly the same altitude, e.g., `s`z = 0`. Any algorithm `break_vz_symm(s:Vect3):Sign` that satisfies the following assumptions will suffice.

```
break_vz_symm_comm : ASSUMPTION
  FORALL (s:Vect3):
    s /= zero IMPLIES
    break_vz_symm(-s) = -break_vz_symm(s)

break_vz_symm_sz : ASSUMPTION
  FORALL (s:Vect3):
    s`z /= 0 IMPLIES
    break_vz_symm(s) = sign(s`z)
```

Here is an example of such an algorithm.

```
break_vz_symm(s:Vect3) : Sign =
  IF s`z > 0 OR
     s`z = 0 AND s`x < 0 OR
     s`z = 0 AND s`x = 0 AND s`y < 0 THEN
```

```
    1
  ELSE
    -1
  ENDIF
```

Our framework does not prescribe any particular algorithm. Note that `break_vz_symm` "breaks the tie" when the aircraft are at the exact same altitude. It insures that one will go up and the other will go down. The example algorithm resolves the problem by using the `x` and `y` components. Since all components cannot simultaneously be equal to zero, otherwise the aircraft would have already collided, there is always something to break the symmetry. Of course other techniques could be used such as the order of the aircraft identifiers or other operational factors.

6 Independent and Coordinated Correctness

We now focus our attention to the formal proofs that our criteria are sufficient conditions for independent and coordinated correctness. The independent case assumes that only one aircraft perform the recovery maneuver. The coordinated case assumes that both aircraft simultaneously perform the recovery maneuver.

6.1 Horizontal Correctness

In this section, we will assume that all vectors are given in a 2D coordinate system. In the *independent* case, we want to prove that any horizontal maneuver for the ownship `nvo` that satisfies `xy_criteria_tr?` is also `xy_correct?`. Formally, we want a theorem of the form

```
xy_criteria_tr?(s,vo,vi,Th)(nvo)
IMPLIES
  xy_correct?[Th](s,vi)(nvo)
```

When both aircraft seek to recovery from the loss of separation it is conceivable that the combined result might not be satisfactory. Therefore, we must establish criteria whereby we can be assured that the collective action produces an appropriate relative velocity vector. In other words, we want a distributed solution that is *coordinated*. Formally, we want a theorem of the form

```
  xy_criteria_tr?( s,vo,vi,Th)(nvo) AND
  xy_criteria_tr?(-s,vi,vo,Th)(nvi)
  IMPLIES
     xy_correct?[Th](s,nvi)(nvo)
```

That is if `nvo` and `nvi` satisfy the criteria with respect to their own frame of reference, i.e., `nvo` with respect to `s,vo,vi` and `nvi` with respect to `-s,vi,vo`, then together they are correct.

6.1.1 Independent Horizontal Correctness

We begin with the following theorem which provides a useful result:

Lemma (xy_indep_lem).
```
nvo /= vi AND
NOT separation?(s) AND
dot_prop?(s,nvo-vi) AND
Th >= tteh(s,nvo-vi)
IMPLIES
  xy_correct?[Th](s,vi)(nvo)
```

Proof. To establish `xy_correct?[Th](s,vi)(nvo)`, we must prove

1. `xy_divergent?(s, nvo-vi)` and

2. `horizontal_separation?(s + Th *(nvo - vi))`.

First, from lemma `dot_prop_divergent`, we have `xy_divergent?(s,nvo-vi)`. Now for the second part, we use lemma `tteh_sq_D`, to obtain

$$[s_x + t_e(v'_{ox} - v_{ix})]^2 + [s_y + t_e(v'_{oy} - v_{iy}))]^2 \quad = \quad D^2,$$

where $t_e =$ `tteh(s,nvo-vi)`. To establish `horizontal_separation?(s + Th * (nvo - vi))`, we must show that

$$[s_x + \mathtt{Th}(v'_{ox} - v_{ix})]^2 + [s_y + \mathtt{Th}(v'_{oy} - v_{iy})]^2 \quad \geq \quad D^2.$$

But this follows immediately from the fact that $t_e \leq$ `Th` and the fact that $\mathbf{v}'_o$ and $\mathbf{v}_i$ are xy-divergent. □

The predicate `dot_prop?(s,nvo-vi)` is a sufficient condition for independent correctness. However, we prefer the following final theorem:

Theorem (xy_independent).
```
xy_criteria_tr?(s,vo,vi,Th)(nvo)
IMPLIES
  xy_correct?[Th](s,vi)(nvo)
```

Proof. Since `xy_criteria?(s,vo,vi)(nvo)` implies the premises of `xy_indep_lem`, this theorem is an immediate consequence of `xy_indep_lem`. □

The advantage of this theorem is that the criteria is identical with what is needed for coordinated correctness. Using this strategy, we will be assured of correctness if only one or if both aircraft attempt to recover simultaneously.

6.1.2 Coordinated Horizontal Correctness

We have successfully proved the following theorem:

Theorem (`xy_coordinated`)**.**

```
xy_criteria_tr?( s,vo,vi,Th)(nvo) AND
xy_criteria_tr?(-s,vi,vo,Th)(nvi)
vect2D(nvo) /= vect2D(nvi) AND
Th >= tteh(s,nvo-nvi)
IMPLIES
  xy_correct?[Th](s,nvi)(nvo)
```

Since the definition of `xy_correct?` is symmetric:

```
xy_correct_symm: LEMMA
  xy_correct?[Th](s,nvi)(nvo) IFF xy_correct?[Th](-s,nvo)(nvi)
```

the theorem is true from either aircraft's perspective. We remark that the second and third premises `vect2D(nvo) /= vect2D(nvi)` and `Th >= tteh(s,nvo-nvi)` involve both `nvo` and `nvi`, the new velocity vectors of both aircraft. Unfortunately, only one of these is known locally in each aircraft. The presence of this premise means that the job of showing that a particular algorithm satisfies this premise cannot be inferred from `xy_criteria_tr?` alone and has to be done for each algorithm separately.

The premise `nvo /= nvi`, i.e., $\mathbf{v}'_o \neq \mathbf{v}'_i$, does not cause us much concern. If for some reason, a horizontal algorithm (that satisfies the criteria) puts the aircraft in this situation where they are parallel, we know that on the next iteration of the algorithm, that it will no longer be in the divergent situation. This next iteration will compute new resolutions that will remove this parallel condition.

We will prove this theorem by first establishing some lemmas. Once again we will work with 2D vectors. The first key lemma shows that if the maneuvers of the originally xy-converging aircraft meet the criteria, then they are simultaneously xy-diverging:

Lemma (`xy_converging_div_coordinated`)**.**

```
NOT dot_prop?(s,vo-vi) AND
dot_prop?(s,nvo-vi) AND
dot_prop?(-s,nvi-vo)
IMPLIES
  divergent?(s,nvo-nvi)
```

Proof. To obtain `divergent?(s,nvo-nvi)`, we use theorem `dot_prop_divergent` which was proven in Section 5.1. It tells us that we only need to prove that `nvo /= nvi` and `dot_prop?(s,nvo-nvi)`, e.g., $\mathbf{s} \cdot (\mathbf{v}'_o - \mathbf{v}'_i) \geq 0$. From the premises we have

$$\begin{aligned} \mathbf{s} \cdot (\mathbf{v}_o - \mathbf{v}_i) &< 0, \\ \mathbf{s} \cdot (\mathbf{v}'_o - \mathbf{v}_i) &\geq 0, \\ -\mathbf{s} \cdot (\mathbf{v}'_i - \mathbf{v}_o) &\geq 0. \end{aligned}$$

Expanding these, we obtain

$$\begin{aligned} -s_x v_{ox} - s_y v_{oy} + s_x v_{ix} + s_y v_{iy} &> 0, \\ s_x v'_{ox} + s_y v'_{oy} - s_x v_{ix} - s_y v_{iy} &\geq 0, \\ -s_x v'_{ix} - s_y v'_{iy} + s_x v_{ox} + s_y v_{oy} &\geq 0. \end{aligned}$$

Adding these equations together yields

$$s_x v'_{ox} + s_y v'_{oy} - s_x v'_{ix} - s_y v'_{iy} > 0$$

or more succinctly

$$\mathbf{s} \cdot (\mathbf{v}'_o - \mathbf{v}'_i) > 0,$$

which implies `dot_prop?(s,nvo-nvi)`. Also since the relation is strictly greater than 0, we have $\mathbf{v}'_o \neq \mathbf{v}'_i$ as needed. □

The next lemma shows that criteria-satisfying maneuvers of aircraft that are are originally xy-diverging aircraft are simultaneously xy-diverging.

Lemma (`xy_diverging_div_coordinated`)**.**

```
diverging_div_coordinated: LEMMA
  dot_prop?(s,vo-vi) AND
  nvo /= nvi AND
  dot_prop?(s,nvo-vo) AND
  dot_prop?(-s,nvi-vi)
  IMPLIES
    % Relative recovery maneuver nvo,nvi is diverging
    divergent?(s,nvo-nvi)
```

Proof. Once again we will obtain `divergent?(s,nvo,nvi)` using theorem `dot_prop_divergent`. We only need to prove `dot_prop?(s, nvo - nvi)` since `nvo /= nvi` is provided as a premise. The `dot_prop?` premises give us

$$\begin{aligned} \mathbf{s} \cdot (\mathbf{v}_o - \mathbf{v}_i) &\geq 0, \\ \mathbf{s} \cdot (\mathbf{v}'_o - \mathbf{v}_o) &\geq 0, \\ \mathbf{s} \cdot (\mathbf{v}_i - \mathbf{v}'_i) &\geq 0. \end{aligned}$$

Adding these together yields

$$\mathbf{s} \cdot (\mathbf{v}_o - \mathbf{v}_i + \mathbf{v}'_o - \mathbf{v}_o + \mathbf{v}_i - \mathbf{v}'_i) \geq 0,$$

which simplifies to

$$\mathbf{s} \cdot (\mathbf{v}'_o - \mathbf{v}'_i) \geq 0.$$

Hence, `dot_prop?(s, nvo - nvi)` holds. □

The next lemma shows that criteria-satisfying maneuvers of aircraft that are originally horizontally parallel, e.g., `vo = vi`, are simultaneously xy-diverging:

Lemma (`xy_static_div_coordinated`).

```
static_div_coordinated: LEMMA
  vo = vi AND
  s*(nvo-vo) > 0 AND
  -s*(nvi-vi) > 0
  IMPLIES
    divergent?(s,nvo-nvi)
```

Proof. Once again we seek to obtain `nvo /= nvi` and `dot_prop?(s,nvo-nvi)`. We substitute the first premise into premises two and three, obtaining

$$\begin{aligned} \mathbf{s} \cdot (\mathbf{v}'_o - \mathbf{v}_i) &> 0, \\ -\mathbf{s} \cdot (\mathbf{v}'_i - \mathbf{v}_i) &> 0, \end{aligned}$$

Adding these together yields

$$\mathbf{s} \cdot (\mathbf{v}'_o - \mathbf{v}_i) + \mathbf{s} \cdot (\mathbf{v}_i - \mathbf{v}'_i) > 0$$

Combining we have

$$\mathbf{s} \cdot (\mathbf{v}'_o - \mathbf{v}_i + \mathbf{v}_i - \mathbf{v}'_i) > 0$$

which simplifies to $\mathbf{s} \cdot (\mathbf{v}'_o - \mathbf{v}'_i) > 0$. Hence, `dot_prop?(s,nvo-nvi)` holds. Also the strict inequality insures that $\mathbf{v}'_o \neq \mathbf{v}'_i$ holds as required. □

Combining these three lemmas we get

Theorem (`xy_div_coordinated`).

```
nvo /= nvi AND
xy_criteria?(s,vo,vi)(nvo) AND
xy_criteria?(-s,vi,vo)(nvi)
IMPLIES
  divergent?(s,nvo-nvi)
```

Proof. By case analysis and lemmas `xy_diverging_div_coordinated`, `xy_static_div_coordinated`, and `xy_converging_div_coordinated`. □

In order to handle the time to exit horizontally, we use the following lemma.

Lemma (`xy_divergent_separation`).

```
xy_divergent?(s,v) IMPLIES
xy_los?(s) AND
tr >= tteh(s,v)
IMPLIES
  horizontal_separation?(s+tr*v)
```

Proof. Using lemma `tteh_sq_D` proven in Section 4.1, we have

$$(s_x + \texttt{tteh}(\mathbf{s}, \mathbf{v}_o - \mathbf{v}_i) * (v_{ox} - v_{ix}))^2 + (s_y + \texttt{tteh}(\mathbf{s}, \mathbf{v}_o - \mathbf{v}_i) * (v_{oy} - v_{iy}))^2 \quad = \quad D^2$$

Given `tr >= tteh(s,vo-vi)`, we easily obtain `horizontal_separation?(s+tr*(vo-vi))`. □

Finally, we can prove that the criteria imply coordinated correctness, assuming `Th >= tteh(s,nvo-nvi)` and `nvo /= nvi`.

Theorem (`xy_coordinated`)**.**

```
xy_criteria_tr?( s,vo,vi,Th)(nvo) AND
xy_criteria_tr?(-s,vi,vo,Th)(nvi) AND
vect2D(nvo) /= vect2D(nvi) AND
Th >= tteh(s,nvo-nvi)
IMPLIES
  xy_correct?[Th](s,nvi)(nvo)
```

Proof. This theorem follows immediately from lemmas `xy_divergent_separation` and
`xy_div_coordinated`. □

6.2 Vertical Correctness

In this section, we will assume that all vectors are given in a 3D coordinate system. As in the horizontal case, we want an independent correctness theorem of the form

```
z_criteria_tr?(s,vo,vi,Tv)(nvo)
IMPLIES
  z_correct?[Tv](s,vi)(nvo)
```

and a coordinated correctness theorem of the form

```
z_criteria_tr?( s,vo,vi,Tv)(nvo) AND
z_criteria_tr?(-s,vi,vo,Tv)(nvi)
IMPLIES
   z_correct?[Tv](s,nvi)(nvo)
```

6.2.1 Independent Vertical Correctness

The main result of this subsection is:

Theorem (`z_independent`)**.**

```
z_criteria_tr?(s,vo,vi,Tv)(nvo)
IMPLIES
  z_correct?[Tv](s,vi)(nvo)
```

Proof. This theorem follows immediately from the definition of `z_criteria_tr?` and lemma `z_prop_independent` below. □

Before we prove `z_prop_independent`, we need the following lemma:

Lemma (`z_prop_divergent`)**.**

```
z_prop?(s,vo-vi) AND
vo‘z - vi‘z /= 0
IMPLIES
  z_divergent?(s,vo-vi)
```

Proof. From `z_prop?(s,nvo-vi)`, we have $s_z(v_{oz} - v_{iz}) \geq 0$. This gives us two cases:

- Case $s_z \leq 0$ and $(v_{oz} - v_{iz}) \leq 0$: $-s_z < -(s_z + t(v_{oz} - v_{iz}))$.
- Case $s_z \geq 0$ and $(v_{oz} - v_{iz}) \geq 0$: $s_z < s_z + t(v_{oz} - v_{iz})$.

Together these cases yield $|s_z| < |s_z + t(v_{oz} - v_{iz})|$, which is `z_divergent?(s,vo-vi)`. □

Lemma (`z_prop_independent`)**.**

```
z_los?(s) AND
z_prop?(s,nvo-vi) AND
(nvo-vi)’z /= 0 AND
Tv >= ttez(s,nvo-vi)
IMPLIES
  z_correct?[Tv](s,vi)(nvo)
```

Proof. To establish `z_correct?[Tv](s,vi)(nvo)` we must prove `z_divergent?(s,nvo-vi)` and `vertical_separation?(s + Tv * (nvo - vi))`. Lemma `z_prop_divergent` suffices to prove the first conjunction and lemma `z_vnz_separation` proved in section 4.2, gives us `vertical_separation?(s + Tv * (nvo - vi))`. □

6.2.2 Coordinated Vertical Correctness

We have proven the following theorem:

Theorem (`z_coordinated`)**.**

```
s /= zero AND
z_criteria_tr?( s,vo,vi,Tv)(nvo) AND
z_criteria_tr?(-s,vi,vo,Tv)(nvi)
 IMPLIES
   z_correct?[Tv](s,nvi)(nvo)
```

In contrast to the horizontal case, we have proved vertical coordinated correctness without any premise that uses non-local information. The premise `s /= zero` states that the aircraft are not in the same position, e.g., the theorem holds as long as the aircraft have not yet collided.

The theorem `z_coordinated` is proven through a series of theorems and lemmas. The first theorem states that the predicate `z_criteria` implies coordinated vertical divergence.

Theorem (z_div_coordinated).

```
s /= zero AND
z_criteria?(s,vo,vi)(nvo) AND
z_criteria?(-s,vi,vo)(nvi)
IMPLIES
  z_divergent?(s,nvo-nvi)
```

Proof. Lemma `z_prop_divergent` establishes `z_divergent?(s,nvo-nvi)` given `z_prop?(s, nvo - nvi)` and `nvo‘z /= nvi‘z`. We consider three cases:

- Case `vo_z /= vi_z` and `z_prop?(s,vo-vi)`: From the premises and case analysis we obtain:

  ```
  s‘z * (vo - vi)‘z >= 0 AND
  sign((vo - vi)‘z) * (nvo - vo)‘z >= 0 AND
  sign((vi - vo)‘z) * (nvi - vi)‘z >= 0
  ```

  From the first premise, we get two cases:

  - Case `0 >= s‘z AND 0 > (vo - vi)‘z`: The two `sign` premises become

    ```
    (vo - nvo)‘z >= 0 AND
    (nvi - vi)‘z >= 0
    ```

  - Case `0 <= s‘z AND 0 < (vo - vi)‘z`: The two `sign` premises become

    ```
    (nvo - vo)‘z >= 0 AND
    (vi - nvi)‘z >= 0
    ```

  From either of these we have `s‘z * (nvo - nvi)‘z >= 0`, which is `z_prop?(s, nvo - nvi)`.

- Case `vo_z /= vi_z` and `NOT z_prop?(s,vo-vi)`: From the premises, we get

  ```
  s‘z * (nvo‘z - vi‘z) >= 0 AND
  -s‘z * (nvi‘z - vo‘z) >= 0 AND
  s‘z * (vo‘z - vi‘z) < 0
  ```

  From these we easily obtain `s‘z * (nvo - nvi)‘z >= 0`, which is `z_prop?(s, nvo - nvi)`.

- Case `vo_z = vi_z`: From the `z_criteria?` premises we have

  ```
  break_vz_symm(s) * (nvo - vo)‘z > 0 AND
  break_vz_symm(-s) * (nvi - vi)‘z > 0
  ```

  If `s‘z = 0`, we trivially have `s‘z * (nvo - nvi)‘z >= 0`, so we are done. If we have both `s‘z <= 0` and `(nvo - nvi)‘z <= 0` we are done.

  When `s‘z /= 0`, `break_vz_symm(s) = sign(s‘z)`. So when `s‘z >= 0`, these premises become

```
 1 * (nvo - vo)`z > 0 AND
-1 * (nvi - vi)`z > 0
```

But since `vo`z = vi`z` this gives us `(nvo`z - nvi`z) >= 0` from which we have `s`z * (nvo - nvi)`z >= 0`.

In all three cases, algebraic simplifications yield `nvo`z /= nvi`z`. □

The next three lemmas state that the predicate `z_criteria_tr?` implies coordinated vertical separation for originally diverging aircraft, originally converging aircraft, and originally parallel aircraft, respectively.

Lemma (`z_diverging_sep_coordinated`).

```
z_criteria_tr?( s,vo,vi,tr)(nvo) AND
z_criteria_tr?(-s,vi,vo,tr)(nvi) AND
z_prop?(s,vo-vi) AND (vo-vi)`z /= 0
IMPLIES
  vertical_separation?(s + tr*(nvo - nvi))
```

Proof. From lemma `z_vnz_separation` (Section 4.2), it suffices to prove

```
tr >= ttez(s, (nvo - nvi))
```

This is obtained by case analysis and simple algebraic manipulations. □

Lemma (`z_converging_sep_coordinated`).

```
z_criteria_tr?( s,vo,vi,tr)(nvo) AND
z_criteria_tr?(-s,vi,vo,tr)(nvi) AND
NOT z_prop?(s,vo-vi)
IMPLIES
  vertical_separation?(s + tr*(nvo-nvi))
```

Proof. We need to prove `nvo_z /= nvi_z` and `tr >= ttez(s, (nvo - nvi))`. The first part, `nvo_z /= nvi_z`, follows from the `z_prop?` premises after expanding `z_criteria_tr?`. To obtain the second part, we must prove that

```
tr >= (sign((nvo - nvi)`z) * H - s`z) / (nvo - nvi)`z
```

This is obtained by case analysis and simple algebraic manipulations. □

Lemma (`z_parallel_sep_coordinated`).

```
s /= zero AND
z_criteria_tr?( s,vo,vi,tr)(nvo) AND
z_criteria_tr?(-s,vi,vo,tr)(nvi) AND
z_prop?(s,vo-vi) AND (vo-vi)`z = 0
IMPLIES
  vertical_separation?(s + tr*(nvo-nvi))
```

Proof. We need to prove

```
NOT (nvo - nvi)'z = 0 AND tr >= ttez(s, (nvo - nvi))
```

so that we can use lemma `z_vnz_separation` to obtain:
`vertical_separation?(s + tr*(nvo-nvi))`. As in the previous cases, this is obtained by case analysis and algebraic manipulations. □

Lemma (`z_sep_coordinated`)**.**

```
s /= zero AND
z_criteria_tr?( s,vo,vi,tr)(nvo) AND
z_criteria_tr?(-s,vi,vo,tr)(nvi)
IMPLIES
  vertical_separation?(s + tr*(nvo-nvi))
```

Proof. By case analysis and lemmas `z_diverging_sep_coordinated`, `z_converging_sep_coordinated`, and `z_parallel_sep_coordinated`. □

Theorem. `z_coordinated`

```
s /= zero AND
z_criteria_tr?( s,vo,vi,Tv)(nvo) AND
z_criteria_tr?(-s,vi,vo,Tv)(nvi)
IMPLIES
  z_correct?[Tv](s,nvi)(nvo)
```

Proof. By lemmas `z_sep_coordinated` and `z_div_coordinated`. □

7 A Simple Vertical Algorithm

We have constructed a prototype vertical resolution algorithm that satisfies the vertical criteria. It can be expressed in PVS as follows

```
z_recovery(s,vo,vi:Vect3,t:posreal): Vect3 =
  LET v   = vo-vi,
      nvz = (sign_vz(s,v)*H - s`z)/t IN
  IF z_prop?(s,v) AND
     abs(v`z) >= abs(nvz) THEN
    (vo`x, vo`y, vo`z)
  ELSE
    (vo`x, vo`y, nvz+vi`z)
  ENDIF
```

with the following subfunctions:

```
% Break tie when relative vertical speed is zero
break_vz_symm(s:Vect3) : Sign =
  IF z(s) > 0 OR
     z(s) = 0 AND x(s) < 0 OR
     z(s) = 0 AND x(s) = 0 AND y(s) < 0 THEN
     1
```

```
    ELSE
      -1
    ENDIF

  sign_vz(s,v:Vect3): Sign =
    IF z_prop?(s,v) AND v`z /= 0 THEN
      sign(v`z)
    ELSE
      break_vz_symm(s)
    ENDIF
```

This algorithm has been proved to satisfy the vertical criteria:

```
  z_recovery_criteria_tr : THEOREM
    LET nvo = z_recovery(s,vo,vi,Tv) IN
      NOT vertical_separation?(s) IMPLIES
      z_criteria_tr?(s,vo,vi,Tv)(nvo)
```

The framework therefore provides an immediate proof that this algorithm satisfies the vertical correctness properties for both the independent version:

```
  z_recovery_independent : THEOREM
    NOT vertical_separation?(s) IMPLIES
      LET nvo = z_recovery(s,vo,vi,Tv) IN
      z_correct?[Tv](s,vi)(nvo)
```

and the coordinated version:

```
  z_recovery_coordinated : THEOREM
    NOT vertical_separation?(s) AND
    s /= zero IMPLIES
      LET nvo = z_recovery(s,vo,vi,Tv),
          nvi = z_recovery(-s,vi,vo,Tv) IN
      z_correct?[Tv](s,nvi)(nvo)
```

8 A Simple Horizontal Algorithm

9 Practicality of the Criteria

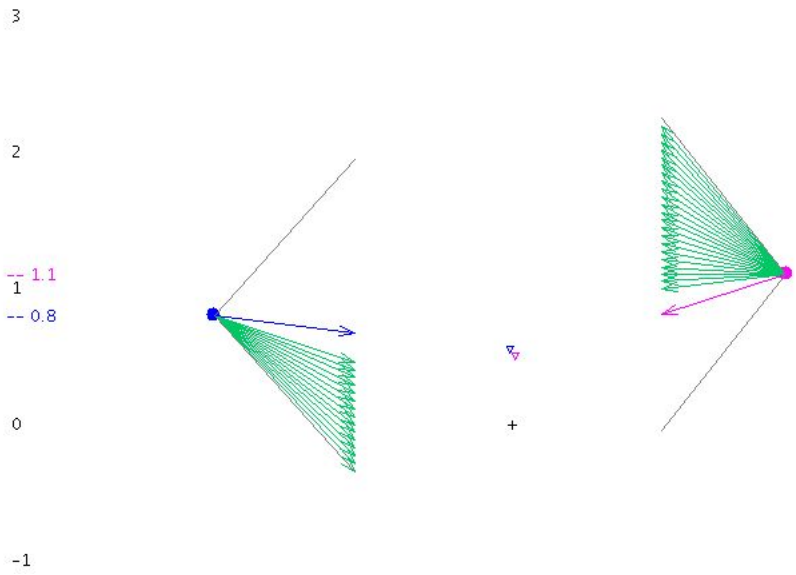
We have a theoretical result that establishes that the criteria are sufficient to meet our correctness properties. But can these criteria be satisfied by reasonable algorithms? For the vertical case, we have a prototype algorithm, so we know that the vertical criteria are satisfiable. But for the horizontal criteria we have not even shown that there exists an algorithm that satisfies the criteria. In both cases, we would also like to gain some insight into how restrictive the criteria are. Therefore, we have created some simple Java programs to graphically display the vectors which meet the criteria given a specific scenario.

9.1 Vertical Criteria Visualization

We begin with the following scenario:

$$\mathbf{s}_o = (1.5, 2.2, 0.8)$$
$$\mathbf{s}_i = (-1.0, -2.0, 1.1)$$
$$\mathbf{v}_o = (5.0, -180, -23.0)$$
$$\mathbf{v}_i = (-10.0, 160, -52.0)$$

The vectors displayed in green are the allowed vertical maneuvers:

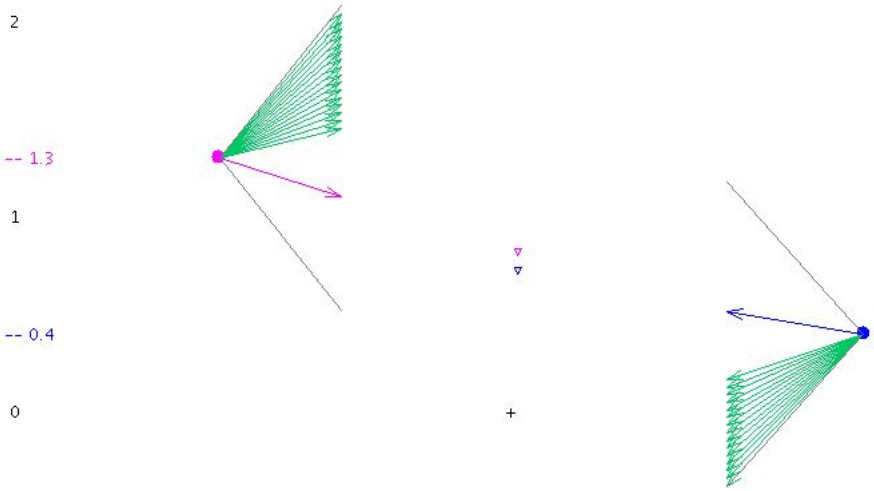


The original ownship vector is displayed in blue and the original traffic vector is displayed in magenta. The gray lines indicate the range of possible vectors when the vertical speed is limited to ± 200 mph or $\pm 17,600$ fpm, which clearly is a larger range than is really needed. As expected one aircraft is given vectors with increased vertical speed, while the other aircraft is given decreased vertical speed maneuvers. Note that some of the allowed traffic vectors have decreasing vertical speed. Thus, there are solutions where both aircraft have decreasing vertical speed.

For initial trajectories that lead to a close vertical encounter:

$$\mathbf{s}_o = (1.8, 0.0, 0.4)$$
$$\mathbf{s}_i = (-1.5, 0.0, 1.3)$$
$$\mathbf{v}_o = (-180, 0, 31.0)$$
$$\mathbf{v}_i = (160, 0, -52.0)$$

more drastic maneuvers are needed:



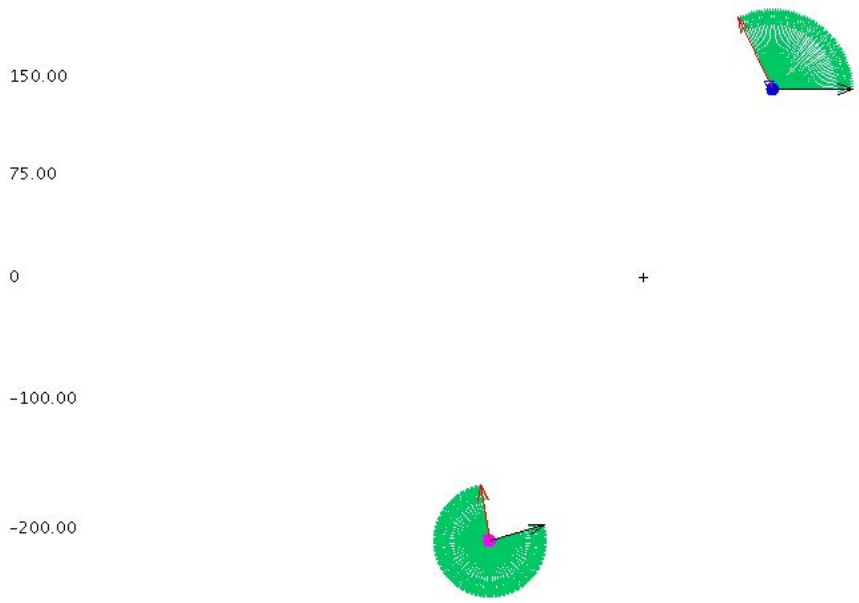
The triangles indicate the locations of the aircraft at the closest point of approach.

9.2 Horizontal Criteria Visualization

The following scenario places the aircraft in an initially divergent situation:

$$\mathbf{s}_o = (100, 150)$$
$$\mathbf{s}_i = (-120, -200)$$
$$\mathbf{v}_o = (80, 0)$$
$$\mathbf{v}_i = (-10, 56)$$

The vectors displayed in green are the maneuvers that satisfy the horizontal criteria where only the heading has been changed.

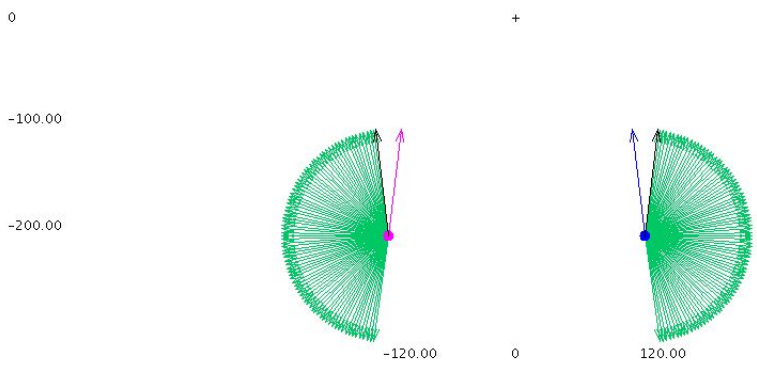


The original ownship vector is displayed in blue and the original traffic vector is displayed in magenta.

Next we look at an originally convergent case where the aircraft are nearly parallel.

$$\mathbf{s}_o = (120, -200)$$
$$\mathbf{s}_i = (-120, 200)$$
$$\mathbf{v}_o = (-10, 80)$$
$$\mathbf{v}_i = (10, 80)$$

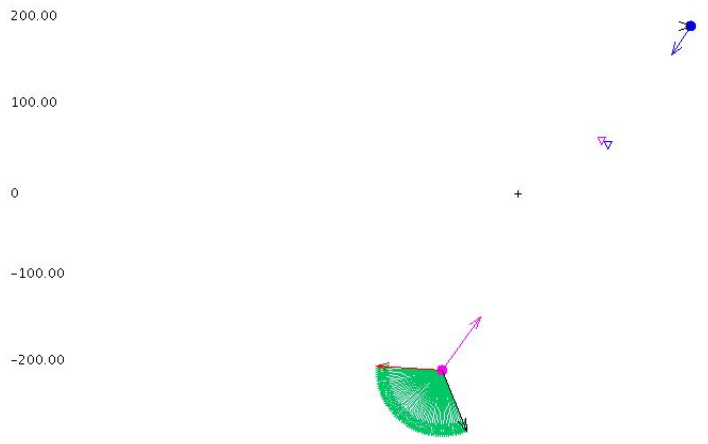
The criteria allows a large set of coordinated maneuvers.



There are some situations where one of the aircraft has no unilateral means of escape by modifying heading only.

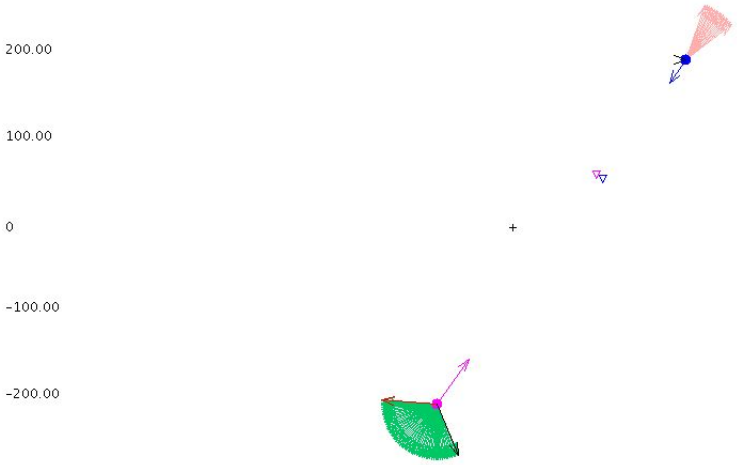
$$\mathbf{s}_o = (200, 200)$$
$$\mathbf{s}_i = (-90, -200)$$
$$\mathbf{v}_o = (-20, -30)$$
$$\mathbf{v}_i = (40, 56)$$

The ownship does not have sufficient speed and enough original separation to escape the traffic.



The triangles indicate the locations of the aircraft at the closest point of approach. The perspicacious reader may be wondering how this can happen given that we have a independent proof of correctness. The answer is simply that the proof merely guarantees that if there is a maneuver that satisfies the criteria, then divergence is guaranteed. However, in this case there are no heading-only vectors that satisfy the criteria. This illustrates that a proof that seeks to establish that an algorithm satisfies the criteria must show that a solution is always produced and not just that all generated solutions are correct.

In this case there are combined heading and ground speed recovery maneuvers that meet the criteria. For example, if the ground speed is doubled so that the ownship's speed exceeds that of the traffic aircraft, then the heading changes shown in pink meet the criteria:

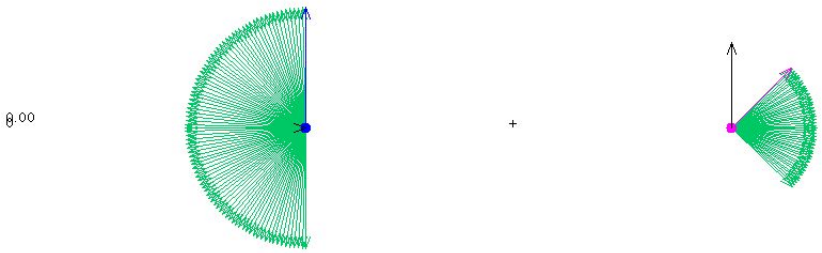


This example also suggests that our correctness property is too strong. In future work, we will explore how to weaken the correctness property to allow some temporary convergence before divergence is achieved.

The following scenario

$$\mathbf{s}_o = (-2.4, 0)$$
$$\mathbf{s}_i = (2.5, 0)$$
$$\mathbf{v}_o = (0, 400)$$
$$\mathbf{v}_i = (200, 200)$$

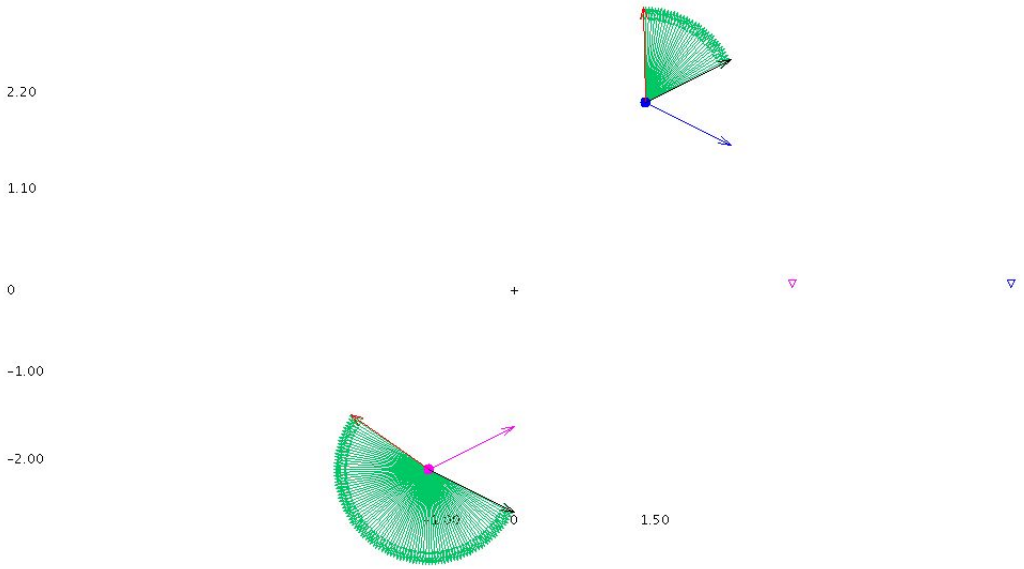
suggests that the current criteria does not allows find all possible solutions. Intuitively, there are clearly some more maneuvers available to the traffic (magenta) aircraft e.g., the opposite direction from the ownship up to an angle of 90^o and down to -90^o.



The following scenario

$$\mathbf{s}_o = (1.5, 2.2)$$
$$\mathbf{s}_i = (-1.0, -2.0)$$
$$\mathbf{v}_o = (200, -100)$$
$$\mathbf{v}_i = (200, 100)$$

shows a case where the criteria leads to fairly drastic horizontal maneuvers



10 Validation

In this section, we want to illustrate with a few examples how some of the concepts presented in this paper can be validated on specific values. This type of validation is useful during the verification process since it helps to catch specification errors before the formal proofs are attempted.

Given the following state information for the ownship and traffic aircraft (In PVS, comments are preceded by the character % and go to the end of the line):

```
so : Vect3 = (-3,2,16500) % [nm,nm,ft]
si : Vect3 = (1,1,16000)  % [nm,nm,ft]
vo : Vect3 = (600,0,0)    % [knots,knots,ft/min]
vi : Vect3 = (0,500,0)    % [knots,knots,ft/min]
```

we can use the PVSio [5] interface to the PVS ground evaluator, to check that the aircraft are in loss of separation:

```
<PVSio> horizontal_separation?(so-si);
==>
FALSE

<PVSio> vertical_separation?(so-si);
==>
FALSE

<PVSio> loss_of_separation?(so-si);
```

```
==>
TRUE
```

Assume that a given loss of separation recovery algorithm returns the following velocity maneuver for the ownship:

```
nvo:Vect3 = (50,700,1000) % [knots,knots,ft/min]
```

We can check that this new velocity guarantees horizontal separation at time `Th` equal to 1 minute, e.g., $\frac{1}{60}$ hours, and vertical separation at time `Tv` equal to $\frac{1}{2}$ minutes:[3]

```
<PVSio> horizontal_separation?((so-si)+Th*(nvo-vi));
==>
TRUE

<PVSio> vertical_separation?((so-si)+Tv*(nvo-vi));
==>
TRUE
```

Note that we cannot use the PVS ground evaluator to check whether `nvo` satisfies the correctness predicates or not. These predicates specify horizontal and vertical divergence, and those properties are expressed using unbounded universal quantification over *all* positive real numbers (for time t). This requires the full power of a general purpose theorem prover, such as PVS, not just a ground evaluator.

On the other hand, we can check using the PVSio interface that `nvo` satisfies both the horizontal and the vertical criteria:

```
<PVSio> xy_criteria?(so-si,vo,vi)(nvo);
==>
TRUE

<PVSio> z_criteria?(so-si,vo,vi)(nvo);
==>
TRUE
```

Furthermore, `nvo` satisfies the criteria when the time to exit horizontally `Th` is equal to 1 minute and the time to exit vertically `Tv` is equal to half a minute:

```
<PVSio> xy_criteria_tr?(so-si,vo,vi,Th)(nvo);
==>
TRUE

<PVSio> z_criteria_tr?(so-si,vo,vi,Tv)(nvo);
==>
TRUE
```

[3]The reason `Th` is given in hours and `Tv` in minutes is because ground speed is given in knots, i.e., nautical miles per hour, while vertical speed is given in feet per minute.

Therefore, by theorems `xy_independent` (Section 6.1.1) and `z_independent` (Section 6.2.1), we can formally establish that `nvo` is an independently correct horizontal and vertical maneuver for the ownship, i.e., `xy_correct?[Th](so-si,vi)(nvo)` and `z_correct?[Tv](so-si,vi)(nvo)` both hold.

Assume that at the same time as the ownship's recovery algorithm returns `nvo`, the traffic's recovery algorithm returns `nvi`:

```
nvi : Vect3 = (800,50,-1000) % [knots,knots,ft/min]
```

We can check that `nvi` also satisfies `xy_criteria_tr` and `z_criteria`, from the traffic perspective:

```
<PVSio> xy_criteria_tr?(si-so,vi,vo,Th)(nvi);
==>
TRUE

<PVSio> z_criteria_tr?(si-so,vi,vo,Tv)(nvi);
==>
TRUE
```

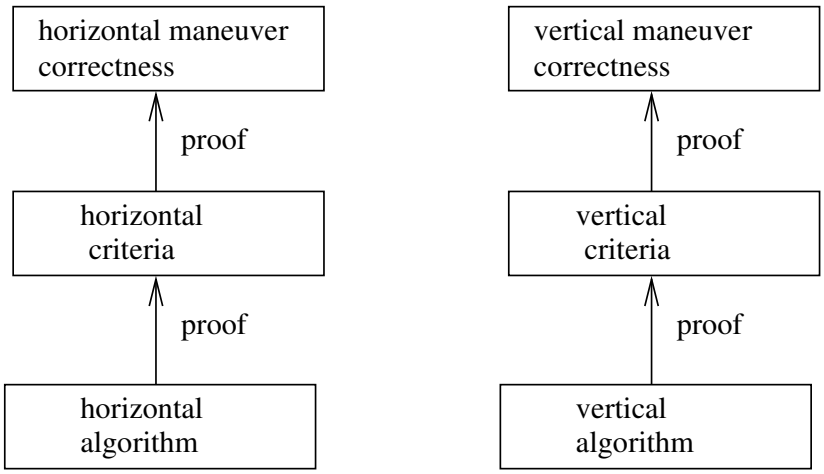
Therefore, by theorems `xy_div_coordinated` (Section 6.1.2) and `z_div_coordinated` (Section 6.2.2), we can formally establish that if the ownship maneuvers according to `nvo` and the traffic aircraft maneuvers according to `nvi` the aircraft will horizontally and vertically diverge, i.e., `xy_divergent?(so-si,nvo-nvi)` and `z_divergent?(so-si,nvo-nvi)` both hold. From Theorem `z_coordinated` (Section 6.2.2), we can also establish that `nvo` and `nvi` satisfy coordinated vertical correctness, e.g.,
`z_correct?[Tv](so-si,nvi)(nvo)` holds. Finally, from Theorem `xy_coordinated` (Section 6.1.2) and since

```
<PVSio> Th >= tteh(so-si,nvo-nvi);
==>
TRUE
```

we can establish that `nvo` and `nvi` satisfy coordinated horizontal correctness, e.g., `xy_correct?[Th](so-si,nvi)(nvo)` holds.

11 Concluding Remarks

In this paper, we have developed a framework for reasoning about the correctness of algorithms that recover from loss of separation. The framework develops concepts of correctness for both horizontal and vertical recovery maneuvers. The framework consists of simple criteria from which the correctness properties of general state-based loss of separation recovery algorithms can be proved. These criteria are easily computed, and therefore if one can show that maneuvers returned by specific algorithms satisfy the criteria, then their correctness is guaranteed. This verification approach is illustrated below:



The framework is illustrated with an example of a vertical algorithm that satisfies the criteria. Future work will develop horizontal algorithms that satisfy the criteria as well.

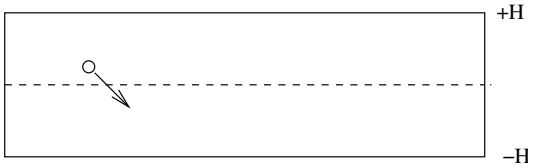
Not all problems are satisfactorily solved by this framework. We have seen cases where our xy-criteria is too strong. There are scenarios where one aircraft has no heading-only solutions[4]. There are also scenarios where the required maneuvers are very drastic. This is a consequence of a definition of correctness that requires immediate divergence. In future work we will explore definitions of correctness that allow temporary convergence before divergence is achieved. We have also mentioned the need of the non-local hypothesis `Th >= tteh(s,nvo-nvi)` in the coordinated horizontal correctness theorem. We had hoped to discharge this hypothesis from the local hypotheses `Th >=tteh(s,nvo-vi)` and `Th >= tteh(-s,nvi-vo)`, but unfortunately this is not always possible (see Appendix A). In practice, this means that our criteria are not strong enough to guarantee the time to exit horizontally when both aircraft maneuver. We will investigate deductive strategies for proving this premise for particular algorithms.

We have not explicitly dealt with the situation where one aircraft executes a horizontal maneuver and the other aircraft executes a vertical maneuver. This is easily handled as two applications of the independent recovery theorems. Nevertheless, we believe this is better handled when proving that a particular algorithm satisfies the criteria, rather than complicating the criteria itself.

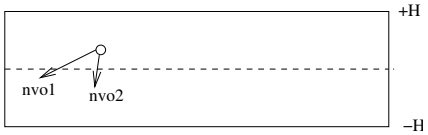
Finally, we remark that our notion of vertical correctness may also be, in some cases, too strong. Suppose you have an aircraft slightly above the midpoint vertically and currently descending:[5]

[4]However, if the ground speed is sufficiently increased then heading maneuvers often become available.

[5]This scenario was pointed out to us by David Wing of NASA Langley.



Within a few seconds the aircraft will be in the lower half. The aircraft will converge for a while, but eventually they will diverge. Therefore, it is imprudent to demand that the aircraft make an abrupt change in direction as our correctness criteria will require. One approach to this anomaly is to assume that there is an operational parameter which determines the maximum amount of time that an aircraft can continue on a convergent path before it must become divergent. However, in the vertical case, this concept would favor maneuvers that are slowly xy-divergent over those that are quickly xy-divergent. Note that the vector with a faster xy-divergence in the following figure



may be ruled out whereas the more vertical one may be allowed. We have still have not found an appropriate definition of correctness that allows some modest amount of convergence before becoming divergent. In this paper, we have restricted our attention to strong divergence and defer further consideration of this issue until later work.

Although we have not yet achieved all of our goals, we believe that the approach advocated in this paper is an important one. An air transportation system built around a criteria approach will be far more general and flexible than a concept where a particular algorithm is mandated (e.g. TCAS II). If safe and effective correctness criteria can be developed, then many different algorithms can be allowed in the airspace and strong safety guarantees maintained. The Air Transportation System can also more efficiently evolve as better algorithms are developed. New algorithms need only be proven to satisfy the criteria in order for there to be system-wide global guarantees that distributed pair-wise resolutions will be coordinated. Nevertheless, a general, mathematically rigorous approach to the high-density N aircraft problem is still very much in its infancy.

References

1. F. Bussink, J. Hoekstra, and B. Heesbeen. Traffic manager: A flexible desktop simulation tool enabling future ATM research. In *Proceedings of the 24th Digital Avionics Systems Conference, IEEE-0-7803-9307-4/05*, volume 3.B.4, pages 1–10, 2005.

2. V. Carreño. Evaluation of a pair-wise conflict detection and resolution algorithm in a multiple aircraft scenario. Technical Report NASA/TM-2002-211963,

NASA Langley Research Center, NASA LaRC,Hampton VA 23681-2199, USA, December 2002.

3. 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.

4. G. Dowek, C. Muñoz, and V. Carreño. Provably safe coordinated strategy for distributed conflict resolution. In *Proceedings of the AIAA Guidance Navigation, and Control Conference and Exhibit 2005, AIAA-2005-6047*, San Francisco, California, 2005.

5. C. Muñoz. Rapid prototyping in PVS. Report NIA Report No. 2003-03, NASA/CR-2003-212418, NIA-NASA Langley, National Institute of Aerospace, Hampton, VA, May 2003.

6. 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.

Appendix A

Theorem `xy_coordinated` Revisited

Our original goal was to prove a theorem of the form

```
xy_criteria_tr?( s,vo,vi,Th)(nvo) AND
xy_criteria_tr?(-s,vi,vo,Th)(nvi)
IMPLIES
   xy_correct?[Th](s,nvi)(nvo)
```

However, as stated in the text, we did not achieve this goal. We were able to prove the following

```
xy_criteria_tr?( s,vo,vi,Th)(nvo) AND
xy_criteria_tr?(-s,vi,vo,Th)(nvi) AND
IMPLIES
   xy_divergent?(s, nvi)(nvo) AND
```

with the additional assumption that `vect2D(nvo) /= vect2D(nvi)`,
but `vertical_separation?(s + Th * (nvo - nvi))` has proved elusive. We had originally expected that the `tteh` premises

```
tteh(s, nvo - vi) <= Th
tteh(-s, nvi - vo) <= Th
```

would together yield

$$[s_x + T_h * (v'_{ox} - v'_{ix})]^2 + [s_y + T_h * (v'_{oy} - v'_{iy})]^2 >= D^2$$

But even in the simpler case, where the aircraft are initially convergent, counter-examples were found. Thus, the following conjecture

```
xy_diverging_sep_coordinated : CONJECTURE
  NOT horizontal_separation?(s) AND
  NOT dot_prop?(s,vo-vi) AND
  dot_prop?(s,nvo-vi) AND
  dot_prop?(-s,nvi-vo) AND
  tr >= tteh(s,nvo-vi) AND
  tr >= tteh(-s,nvi-vo) AND
  IMPLIES
  horizontal_separation?(s + tr*(nvo - nvi))
```

was found to be falsified by:

```
vo= (-25,-25)
vi= (-25,-24)
nvo = (-11,-19)
nvi = (-7,-25)
s = (-2,12)
D = 25
```

Next we constructed a Java program to search for counterexamples by discretizing the geometry. We hoped to find additional constraints that would suffice. For example, we hoped that adding the following premises:

```
nvo*vo > 0.9 * norm(nvo)*norm(vo) AND
nvi*vi > 0.9 * norm(nvi)*norm(vi)
```

which restricts the cosines of the angles between the old and new velocity vectors to be greater than 0.9 would be sufficient. (In other words, the angle between them must be less that 25^o). With this restriction, counterexamples were still found. So next we added,

```
s*(nvo-vi) > 0.707 * norm(s)*norm(nvo-vi)
```

and sought to prove

```
horizontal_separation?(s + (1.2*tr)*(nvo - nvi))
```

giving 20% more time for the coordinated algorithm to achieve horizontal separation. For awhile it looked like this would suffice. Stepping each variable from -25 to 25 by 2 failed to find a counter-example. After $25^8 = 1.25 \times 10^{11}$ test cases, no counter-examples were seen. But after stepping each variable from -25 to 25 by 1, counter-examples began to appear. If we find suitable constraints that do not produce any counter-examples, we expect that a complete run of the Java program at step-level 1 would take over two weeks. But, even then we still need to construct a mathematical proof. The absence of counterexamples does not rule out the possibility that if we reduce the grid size again (e.g. to say 1/2), that counter-examples would not appear. This is a cogent reminder to us why simulation alone cannot establish safety or correctness.

Appendix B

Vectors Library

The NASA PVS library located at
`http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html` contains three distinct vectors libraries:

1. 2-dimensional vectors
2. 3-dimensional vectors
3. N-dimensional vectors

One might wonder why there should be 2D and 3D versions, when an N-dimensional version is available. The answer is that there are some notational conveniences for doing this. For example, in the 2D version we represent a vector as

```
Vector: TYPE = [#  x, y: real  #]
```

whereas in the N-dimensional library a vector is

```
Index     : TYPE = below(n)
Vector    : TYPE = [Index -> real]
```

where `n` is a formal parameter of type `posnat` to the theory. Thus, in the two dimensional case, the x-component of a vector `v` is `v`x` whereas in the N-dimensional library it is `v(0)`. Also certain operations are greatly simplified in the 2D case. The dot product is

```
*(u,v): real  =  u`x * v`x + u`y * v`y;          % dot product
```

in the 2-dimensional case, whereas in the N-dimensional case it is

```
*(u,v): real = sigma(0,n-1,LAMBDA i:u(i)*v(i));     % Dot Product
```

where sigma is a summation operator imported from the reals library.

All of the lemmas and definitions are as identical as possible to simplify the use of these libraries. The following theories are available

```
vect2D,             % Define 2-D Vector from N-dimensional vectors
vect3D,             % Define 3-D Vector from N-dimensional vectors
vectors2D,          % 2-dimensional vectors and operations
vectors3D,          % 3-dimensional vectors and operations
vectors_cos,        % Law of cosines for n-D vectors
vectors2D_cos,      % Law of cosines for 2D vectors
vectors3D_cos,      % Law of cosines for 3D vectors
position,           % using vectors for position, distance function
position2D,         % using vectors for 2D-position, distance function
```

```
  position3D,         % using vectors for 3D-position, distance function
  lines,              % Using vectors to define lines, and motion
  lines2D,            % Using vectors to define lines, and motion
  lines3D,            % Using vectors to define lines, and motion
  law_cos_pos2D,      % Law of cosines for 2D positions
  law_cos_pos3D,      % Law of cosines for 3D positions
  closest_approach,    % calculate t_cpa for moving particles
  closest_approach_2D, % calculate t_cpa for moving particles
  closest_approach_3D, % calculate t_cpa for moving particles
  perpendicular2D,     % line perpendicular to a line through a point
  perpendicular3D,     % line perpendicular to a line through a point
  intersections2D,     % finding intersection points of lines
  matrices,            % Theory of matrices
  vect_trig_2D,        % trigonometric properties of 2D vectors
  vect_trig_3D,        % trigonometric properties of 3D vectors
  cross_3D,            % cross-product
  linear_independence_3D, % linear independece
  sigma_2D,            % summations over 2D vectors
  sigma_3D,            % summations over #D vectors
```

There are several dozen lemmas available for manipulating vectors, including primitive operations such as

```
add_assoc           : LEMMA u+(v+w) = (u+v)+w
add_move_right      : LEMMA u + w = v IFF u = v - w
add_cancel_left     : LEMMA u + v = u + w IMPLIES v = w
neg_distr_sub       : LEMMA -(v - u) = u - v
dot_eq_args_ge      : LEMMA u*u >= 0
dot_distr_add_right : LEMMA (v+w)*u = v*u + w*u
dot_scal_left       : LEMMA (a*u)*v = a*(u*v)
dot_scal_canon      : LEMMA (a*u)*(b*v) = (a*b)*(u*v)
sqv_scal            : LEMMA sqv(a*v) = sq(a)*sqv(v)
sqrt_sqv_norm       : LEMMA sqrt(sqv(v)) = norm(v)
norm_eq_0           : LEMMA norm(v) = 0 IFF v = zero
cauchy_schwartz     : LEMMA sq(u*v) <= sqv(u)*sqv(v)
```

and more advanced capabilities such as

```
linearly_dependent?(a,b: Vect3): bool =
    (EXISTS (k1,k2: real): (k1 /= 0 OR k2 /= 0) AND
                            k1*a + k2*b = zero)

linearly_independent?(a,b: Vect3): bool = NOT linearly_dependent?(a,b)

aa,bb: VAR Nz_vect3
lin_indep_cross: LEMMA linearly_dependent?(aa,bb) IFF
                           cross(aa,bb) = zero
```

or properties of the mixed product:

```
[||](a,b,c): real = a*cross(b,c)

mixed_prod_perm     : LEMMA [|a,b,c|] = [|b,c,a|]
mixed_prod_scal1    : LEMMA [|k*a,b,c|] = k*[|a,b,c|]
mixed_prod_dist     : LEMMA [|a+d,b,c|] = [|a,b,c|] + [|d,b,c|]
cross_cross_mixed   : LEMMA cross(cross(a,b),cross(c,d)) =
                            [|a,b,d|]*c - [|a,b,c|]*d
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 10 - 2008	Technical Memorandum	

4. TITLE AND SUBTITLE

A Formal Framework for the Analysis of Algorithms That Recover From Loss of Separation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Butler, Ricky W.; and Muñoz, César A.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

411931.02.51.07.01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

L-19460

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)

NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

NASA/TM-2008-215356

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 61
Availability: NASA CASI (301) 621-0390

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We present a mathematical framework for the specification and verification of statebased conflict resolution algorithms that recover from loss of separation. In particular, we propose rigorous definitions of horizontal and vertical maneuver correctness that yield horizontal and vertical separation, respectively, in a bounded amount of time. We also provide sufficient conditions for independent correctness, e.g., separation under the assumption that only one aircraft maneuvers, and for implicitly coordinated correctness, e.g., separation under the assumption that both aircraft maneuver. An important benefit of this approach is that different aircraft can execute different algorithms and implicit coordination will still be achieved, as long as they all meet the explicit criteria of the framework. Towards this end have sought to make the criteria as general as possible. The framework presented in this paper has been formalized and mechanically verified in the Program Verification System (PVS).

15. SUBJECT TERMS

Conflict detection and resolution; Formal methods; Separation assurance; Software safety; Verification

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	46	(301) 621-0390