

Aircraft Trajectory Modeling and Alerting Algorithm Verification

Víctor Carreño¹ and César Muñoz²

¹ Assessment Technology Branch
Mail Stop 130, NASA Langley Research Center
Hampton, VA 23681-2199
`v.a.carreno@larc.nasa.gov`

² Institute for Computer Applications in Science and Engineering (ICASE)
Mail Stop 132C, 3 West Reid Street
NASA Langley Research Center
Hampton VA 23681-2199
`munoz@icase.edu`

Abstract. The Airborne Information for Lateral Spacing (AILS) program at NASA Langley Research Center aims at giving pilots the information necessary to make independent approaches to parallel runways with spacing down to 2500 feet in Instrument Meteorological Conditions. The AILS concept consists of accurate traffic information visible on the navigation display and an alerting algorithm which warns the crew when one of the aircraft involved in a parallel landing is diverting from its intended flight path. In this paper we present a model of aircraft approaches to parallel runways. Based on this model, we analyze the alerting algorithm with the objective of verifying its correctness. The formalization is conducted in the general verification system PVS.

1 Introduction

The Airborne Information for Lateral Spacing (AILS) [12, 3, 6] is a project being conducted at NASA Langley Research Center. Its objective is to reduce traffic delays and increase airport efficiency by enabling approaches to closely spaced parallel runways in Instrument Meteorological Conditions.

Approaches to parallel runways are currently limited to 4300 feet in Instrument Meteorological Conditions. Specially equipped airports with fast scan radars, high resolution monitoring systems, and approach-specific air traffic controllers can perform parallel approaches to 3400 feet [14, 8]. The AILS project aims at shifting the responsibility of maintaining separation during parallel approaches from the air traffic controller to the aircraft crew. Via the AILS concept, approaches to parallel runways 2500 feet apart in Instrument Meteorological Conditions are expected.

AILS eliminates the delay inherent in the communication between air traffic controller and crew by displaying parallel traffic information in the cockpit. The degree of safety is enhanced by an alerting system which warns the crew when

one of the aircraft involved in a parallel landing is deviating from the intended flight path. The alerting algorithm is a critical part of the AILS concept. Flaws in its logic could lead to non-alerted collision incidents. The algorithm has been extensively tested in simulators and in real flights.

The objective of this work is to conduct a formal analysis of the alerting algorithm in order to discover any possible errors that have not been detected during testing and simulation. In particular, we develop a formal model of parallel landing scenarios. Based on this model, we study the behavior of the AILS alerting algorithm with respect to collision incidents. In particular, we have found maximum and minimum times when an alarm will first sound prior to a collision. Indeed, we have proven that for any trajectory leading to a collision, an alarm is issued at least 4 seconds before the collision. Conversely, we have found that there exist trajectories leading to a collision where the alarm will not sound before 11 seconds. We believe that for all cases the largest time prior to a collision when the alarm will first sound is closer to 11 than to 4.

The paper is organized as follows. First, in section 2, we shortly review the alerting features which are integrated in the AILS concept. Next, in section 3, we describe in detail the AILS alerting algorithm. We model aircraft trajectories and collision scenarios in section 4. Section 5 contains the main properties that we have formally proven. Finally, we conclude with some remarks in section 6. The formalization presented in this paper has been developed in the general verification system PVS [11]. We use a stylized- \LaTeX PVS concrete syntax and assume the reader is familiar with standard notations of higher-order logic.

2 System Description

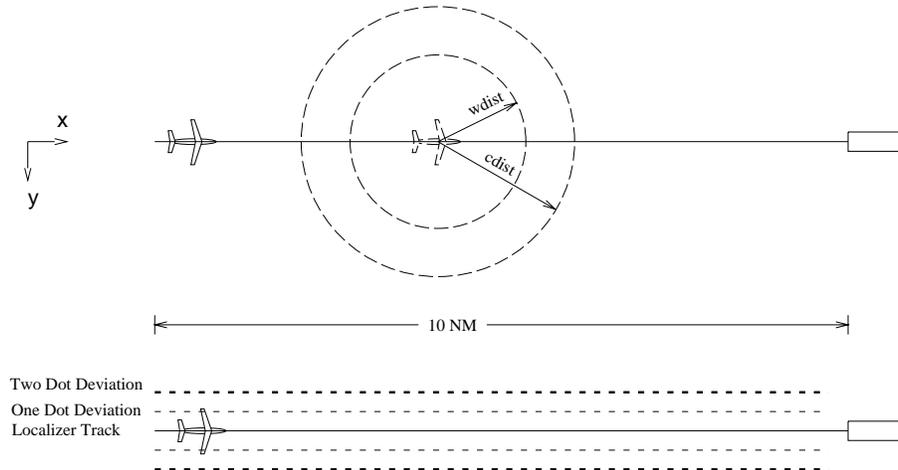


Fig. 1. Parallel runway approach

In a typical independent parallel approach, depicted in Figure 1, aircraft intersect their localizer track (longitudinal runway center) approximately 10 nautical miles from the runway threshold. During localizer intersection, aircraft have a 1000 feet vertical separation. After the aircraft are established in their localizer track, vertical separation is eliminated and aircraft start a normal glide path for landing.

The AILS alerting system starts operating when the aircraft are on their localizers. At this time the aircraft are approximately at the same altitude. As explained later, one aircraft is assumed to be the intruder and the other is assumed to be the evader. The scenario is then reversed. When the intruder aircraft deviates from its airspace, the AILS system provides 6 alert levels, depending on the severity of the deviation. Table 1 shows an alerting sequence as seen in the evader and intruder aircraft primary and navigation displays.

Table 1. Alerting sequence

	Evader	Intruder
1		Localizer alert (one dot deviation)
2		Localizer alert (two dot deviation)
3		Caution alert (traffic)
4	Caution alert (traffic)	
5		Warning alert (collision)
6	Warning alert (collision)	

All alerts in the intruder aircraft are expected to be followed by a corrective maneuver. The evader aircraft is not expected to perform an evasive maneuver until a warning alert is issued, at which time landing is aborted and an emergency escape maneuver is performed. Notice that the intruder aircraft always receives a caution or warning alert before the respective caution or warning alerts are issued to the evader.

An algorithm implementing the alerting features explained above runs independently on each aircraft. It runs twice every 0.5 seconds. The first time the algorithm assumes that the own-ship is the intruder aircraft and the adjacent aircraft is the evader. In the next iteration the algorithm assumes that the own-ship is the evader and the adjacent aircraft is the intruder.

Several assumptions were made by the AILS project researchers in the development of the alerting algorithm. These assumptions are justified by physical characteristics and operational constraints. They are as follows:

- Time is discrete and divided in increments of 0.5 seconds. In our model, we call this value `tstep`.
- The rate of turn is determined by the bank angle and ground speed.
- The speeds of the aircraft are constant. Henceforth, we use `intruderSpeed` and `evaderSpeed` as the constant speed values of the intruder and evader aircraft, respectively.

- The vertical separation between the aircraft is assumed to be 0 during a landing approach.
- Only the intruder aircraft will deviate from its path in a parallel approach. The evader aircraft is assumed to stay in its localizer with a heading angle of 0° .

It should be noted that the experimental AILS system, as currently designed, forms part of the Traffic Alert and Collision Avoidance System (TCAS) [13]. In this work, we assume that the AILS alerting algorithm is running in isolation from other aircraft components. In addition, we concentrate on the caution and warning alerting kernel of the AILS alerting system. The one dot and two dot deviation alerts present a simple scenario and can be easily added to our model by a separate function as it is done in the current implementation.

3 The AILS Alerting Algorithm

In this section we describe the alerting algorithm. We start in subsection 3.1 with a detailed, but informal, description of the actual algorithm. Then, in section 3.2, we abstract and formalize it in the PVS specification language.

3.1 Detailed description

The alerting algorithm determines when an alarm will be triggered by calculating possible collision trajectories and comparing the future aircraft locations with predetermined time and distance thresholds. The algorithm is executed in two modes every `tstep` seconds: (1) the first mode assumes its own aircraft is a threat to the adjacent aircraft and the adjacent aircraft is following the localizer; (2) the second mode assumes the adjacent aircraft is a threat to its own and the own is following the localizer. In either mode, one aircraft is the intruder and one is the evader.

The algorithm considers two cases depending on whether the intruder is changing direction or not. When the intruder aircraft is not changing direction, i.e., its bank angle is 0, the algorithm determines if the two aircraft are diverging or converging and the point of closest separation. This is done by obtaining the derivative of the distance between the aircraft and solving for time when the derivative equals zero as follows.

$$\Delta_x(t) = x_{in}(t) - x_{ev}(t) \quad (1)$$

$$\Delta_y(t) = y_{in}(t) - y_{ev}(t) \quad (2)$$

$$\frac{d}{dt}\Delta_x(t) = \text{intruderSpeed} \times \cos(\theta) - \text{evaderSpeed} \quad (3)$$

$$\frac{d}{dt}\Delta_y(t) = \text{intruderSpeed} \times \sin(\theta) \quad (4)$$

$$R(t) = \sqrt{\Delta_x(t)^2 + \Delta_y(t)^2} \quad (5)$$

$$\frac{d}{dt}R(t) = \frac{\Delta_x(t) \times \frac{d}{dt}\Delta_x(t) + \Delta_y(t) \times \frac{d}{dt}\Delta_y(t)}{\sqrt{R(t)}} \quad (6)$$

For a time t , $(x_{in}(t), y_{in}(t))$ and $(x_{ev}(t), y_{ev}(t))$ are the coordinates of the intruder and evader aircraft, respectively, and θ is the heading angle of the intruder aircraft. When $\frac{d}{dt}R(t + \tau) = 0$, we get the time τ , relative to t , of the point of closest separation of the aircraft. Time τ has been calculated as

$$\tau(t) = -\frac{\Delta_x(t) \times \frac{d}{dt}\Delta_x(t) + \Delta_y(t) \times \frac{d}{dt}\Delta_y(t)}{\frac{d}{dt}\Delta_x(t)^2 + \frac{d}{dt}\Delta_y(t)^2} \quad (7)$$

Equations 3, 4, 6, and 7 were formally deduced by using the computer algebra tool MuPAD [4]. Notice that τ is undetermined when the aircraft are parallel and the ground speeds are equal. In this case, the alerting algorithm defines $\tau(t) = 0$ for any t . Since the evader aircraft is assumed to stay in its localizer with a heading angle of 0° , it does not have a y -speed component. This is reflected in Equation 4.

For a time t , if $\tau(t)$ is negative or zero, the tracks are diverging or parallel, respectively. If $\tau(t)$ is greater than zero, the tracks are converging and $\tau(t)$ will be the time of closest separation (Figure 2). When tracks are diverging or parallel, the algorithm checks the aircraft separation at the present time against the threshold distance for warning or caution alert. When tracks are converging, the algorithm compares the time and distance of closest separation against time and distance thresholds, respectively. In either case, an alarm is triggered when the calculated time and distance are within the time and distance alert thresholds.

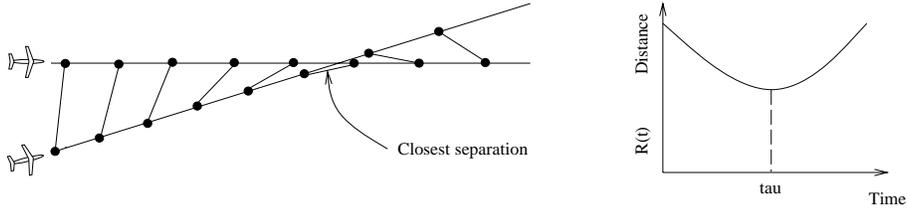


Fig. 2. Converging tracks

When the intruder aircraft is changing direction, i.e., its bank angle is not 0, the algorithm calculates the radius of the turn and the rate of change of direction. Tangential tracks are calculated from the arc path as to produce tangents which are 1.5° to 3° in angular separation (Figure 3). For each of these tangential tracks the algorithm determines whether the two aircraft tracks are diverging or converging and performs time and distance comparisons as explained above.

3.2 PVS abstraction

The original AILS algorithm was written in FORTRAN at Langley Research Center. It has been revised several times and the latest version flown in the Boe-

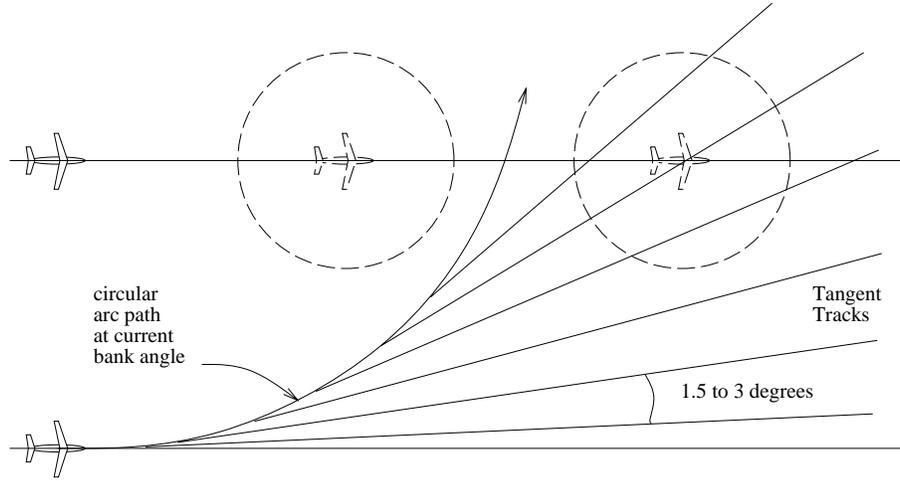


Fig. 3. Radial trajectory and tangential tracks

ing 757 experimental aircraft was provided by Honeywell. For the work presented in this paper, we created a high level abstract model of the alerting algorithm in the PVS language. The algorithm model uses the same strategy as the FORTRAN algorithm to determine if alarms are triggered, as explained above. All of the PVS declarations involved in the modeling of the algorithm can be seen in the theory file available at <http://shemesh.larc.nasa.gov/people/vac/ails.pvs>.

The model of the algorithm is a function which takes the states of the aircraft and returns a Boolean value corresponding to whether the alarm is triggered or not. The type of the alarm, caution or warning, depends on the threshold parameters. However, we only consider a generic type of alarm which abstracts from warning and caution alarms. The state of an aircraft is defined by a record with fields `x`, `y`: the position coordinates; `heading`: the angles between the flight path and the localizer track; and `bank`: the bank angle which range between -45° and 45° (type `Bank`). In PVS:

```
Bank : TYPE = {r:real | -45 ≤ r ≤ 45}
```

```
State : TYPE =
  [# x          : real,
   y          : real,
   heading    : real,
   bank       : Bank
  #]
```

Access to records can be written in PVS as function calls, i.e., if `s` is a `State`, `x(s)` refers to the field `x` of the state `s`.

The model of the alerting algorithm is given next.

```

larcalert(intruder, evader:State): bool =
  LET phi = bank(intruder) IN
  LET trkrate =  $g \times (180/\pi) \times \text{tand}(\text{phi}) / \text{intruderSpeed}$  IN
  IF trkrate = 0 THEN
    chktrack(intruder, evader, 0)
  ELSE
    LET arcrad =
      intruderSpeed2 / ( $g \times \text{tand}(\text{phi})$ ) IN
    LET idtrk =
      IF abs(trkrate)  $\geq 3$  THEN 1
      ELSIF abs(trkrate)  $\geq 1+1/2$  THEN 2
      ELSIF abs(trkrate)  $\geq 3/4$  THEN 4
      ELSE 8
      ENDIF IN
    arc_loop(intruder, evader, arcrad, trkrate, idtrk, 0)
  ENDIF

```

where g is the gravitational acceleration constant (approx. 32.2 feet/seconds²).

The first part of the function `larcalert` is exercised when the track rate (`trkrate`) is zero and there is no change in the intruder's heading. In that case, the function `chktrack` makes the calculation for converging or diverging tracks, according to Equations 1 to 7. If the tracks are diverging, the function `chkrange` is called to compare present locations against time and distance thresholds (`alertTime` and `alertRange`, respectively). If the tracks are converging, predicted locations at caution time or time of closest separation, whatever is smaller, are compared. An alarm is issued when calculated time and distance values are within the range of time and distance alert thresholds.

The structure of the definitions of `chkrange` and `chktrack` are given next.

```

chkrange(range, t:real): bool =
  range  $\leq$  alertRange  $\wedge$  t  $\leq$  alertTime

chktrack(intruder, evader:State, t:real): bool =
  LET range =  $R(t)$  IN
  LET tau =  $\tau(t)$  IN
  IF tau  $\leq$  0 THEN
    chkrange(range, t)
  ELSE
    IF t+tau > alertTime THEN
       $R(\text{alertTime}) \leq \text{alertRange}$ 
    ELSE
       $R(t+\text{tau}) \leq \text{alertRange}$ 
    ENDIF
  ENDIF

```

The second part of the function `larcalert` handles the case when the intruder is changing direction. The arc radius is calculated and the function `arc_loop` generates the tangential tracks from the arc trajectory. The function

`arc_loop` is a recursive function modeling a DO-LOOP statement. It is used to iterate the function `chktrack` on tangential tracks every `idtrk` time steps. The actual definition of `arc_loop` is too long to be included in the paper and can also be seen in the theory file as pointed above. The structure of the function is:

```
arc_loop(intruder,evader,arcrad,trkrate,idtrk,iarc): RECURSIVE bool =
  IF iarc = MaxStep THEN FALSE
  ELSE
    calculate positions of aircraft
    IF not time for a tangential track THEN
      IF chkrange(...) THEN      % Check range at that point.
        TRUE                      % Trigger an alarm.
      ELSE
        arc_loop(...,iarc+1)    % Go to new iteration.
      ENDIF
    ELSE
      % Time for tangential tracks.
      IF chktrk(...) THEN        % Check track at this point.
        TRUE                      % Trigger an alarm.
      ELSE
        arc_loop(...,iarc+1)    % Go to new iteration.
      ENDIF
    ENDIF
  ENDIF
```

Based on the `idtrk` argument and the step in the loop `iarc`, the function `arc_loop` determines if a tangential track is calculated or not. If a tangential track is not calculated, the function `chkrange` compares the distance between the calculated positions of the aircraft and the distance threshold. The function `chktrk` is used to check for collisions on all the tangential tracks in the loop. The function `arc_loop` terminates when one of the functions `chkrange` or `chktrack` triggers an alarm or when `iarc` has reached a constant `MaxStep` defined as `alert_time/tstep`.

In the PVS model, we are using an axiomatic definition of the square root function (`sqrt`, see section 5). Trigonometric functions (`sind`, `cosd`, and `tand`, for sine, cosine, and tangent of angles in degrees, respectively) are defined by series approximations. However, as we will see in section 5, we also provide axioms about trigonometric functions to facilitate the proofs.

As we have seen, the AILS algorithm considers a limited set of possible trajectories for the intruder aircraft, i.e., assuming a constant radius turn at the original bank angle, only tangent track escapes to the turn arc are considered. The developers of the algorithm state that this assumption is reasonable under normal circumstances, i.e., the intruder aircraft is not intentionally trying to collide with the evader aircraft. However, to evaluate the behavior of the algorithm in a wider range of possible landing scenarios, a more general model of trajectories for the intruder aircraft is necessary. In the next section, we develop such a model.

4 Parallel Landing Scenarios

According to the characteristics and assumptions of the AILS algorithm, we propose a time-discrete model of trajectories with time increments of `tstep` seconds. In that model, as in the case of the alerting algorithm, intrusion paths are determined by the bank angle and ground speed of the intruder aircraft. Given a ground speed $gs > 0$, a bank angle ϕ , the heading turn rate is given by

$$\text{trkrate}(gs, \phi) = \frac{\text{tand}(\phi) \times g \times 180}{gs \times \pi},$$

where g is the gravitational acceleration constant.

Although under normal operation the bank angle of a commercial aircraft is limited to -30° to 30° , we allow the bank angle to range from -45° to 45° . For a minimum ground speed of 180 feet per second, it means a maximum heading turn rate of about 6° per second. These values produce very aggressive blundering situations quite consistent with worst cases scenarios tested by the AILS developing group. Incidentally, the function `trkrate` is well-defined for bank angles in that range.

Definition 1 (Intruder trajectory). *An intruder trajectory of length n for an aircraft with state s and ground speed gs is a sequence of states $in_0 \dots in_n$ such that $in_0 = s$ and for $0 < i \leq n$,*

1. $|\text{heading}(in_i) - \text{heading}(in_{i-1})| = \text{tstep} \times \text{trkrate}(gs, \text{bank}(in_i))$,
2. $x(in_i) = x(in_{i-1}) + gs \times \text{tstep} \times \text{cosd}(\text{heading}(in_i))$, and
3. $y(in_i) = y(in_{i-1}) + gs \times \text{tstep} \times \text{sind}(\text{heading}(in_i))$.

In PVS, we define the next state of an intruder aircraft at state `s` and bank angle ϕ by the function

```
next_intruder_state(s:State, phi:Bank): State =
  s WITH [
    x      := x(s) + intruderSpeed * tstep * cosd(heading(s)),
    y      := y(s) + intruderSpeed * tstep * sind(heading(s)),
    heading := heading(s) + tstep * trkrate(intruderSpeed, bank(s)),
    bank   := phi
  ]
```

The notation `WITH` is the record (and function) overriding operator in PVS.

We model an intruder trajectory by a recursive function having as parameters an initial state `s`, a bank angle assignment for each iteration step `tr`, and the iteration step `n`, as follows

```
intruder_trajectory(s:State, tr:[posnat -> Bank], n:nat):
  RECURSIVE State =
    IF n = 0 THEN s
```

```

ELSE
  next_intruder_state(intruder_trajectory(s, tr, n-1),tr(n))
ENDIF
MEASURE n

```

For example, given an intruder aircraft at initial state s and bank angle equal to 0, a trajectory of length n such that the plane follows a straight line to its current heading angle is given by $in_0 \dots in_n$, where $in_0 = s$ and for $0 < i \leq n$,

$$in_i = \text{intruder_trajectory}(s, \lambda(n : \text{posnat}) : 0, i).$$

For the evader aircraft, we assume that it stays in its localizer with a constant speed and constant heading of 0° . Heading and bank angles are irrelevant in the definition of an evader trajectory.

Definition 2 (Evader trajectory). *An evader trajectory of length n for an aircraft with state s and ground speed gs is a sequence of states $ev_0 \dots ev_n$ such that $ev_0 = s$ and for $0 < i \leq n$,*

1. $x(ev_i) = x(ev_{i-1}) + gs \times \text{tstep}$ and
2. $y(ev_i) = y(ev_0)$.

For an initial state s of an aircraft, its state after n steps in a evader trajectory is defined by $\text{evader_trajectory}(s, n)$ as follows

```

evader_trajectory(s:State, n:nat): State =
  (#
    x      := x(s) + evaderSpeed×tstep×n,
    y      := y(s),
    heading := heading(s),
    bank   := bank(s)
  #)

```

We are interested in trajectories leading to collision incidents. Aircraft are said to be in *collision* if the distance between them is less than or equal to collisionRange . In our development, we consider 200 feet for collisionRange , which is approximately the wing span of a Boeing 747.

```

distance(s1,s2:State): real =
  sqrt((x(s2)-x(s1))2 + (y(s2)-y(s1))2)

```

```

collision(s1,s2:State): bool =
  distance(s1,s2) ≤ collisionRange

```

Definition 3 (Collision scenario). *Given an intruder trajectory $in_0 \dots in_n$ and an evader trajectory $ev_0 \dots ev_n$, we said that they lead to a collision incident at step i , for $0 \leq i \leq n$, if $\text{collision}(in_i, ev_i)$ holds.*

A collision scenario is defined in PVS as follows

```

collision_scenario(intruder, evader:State, tr:[posnat → Bank],
                  i:nat):bool =
  collision(intruder_trajectory(intruder, tr, i),
           evader_trajectory(evader, i))

```

We have implemented the model of trajectories, together with our high-level version of the alerting algorithm, in Java. The implementation, available in the same location as the PVS theory files, serves a double purpose. First, it allows us to graphically visualize all the collision trajectories for a given time and initial values of the intruder and evader aircraft. Trajectories are difficult to visualize in PVS given the huge amount of data generated as output by the model. Second and more importantly, by studying those trajectories, we were able to extract conjectures that we have then formally proven in PVS. Conversely, as we will mention later, we have rejected some conjectures by finding counter-examples via simulation of collision trajectories,

In the next section, we formally study in PVS the behavior of the alerting algorithm with respect to our model of collision trajectories.

5 Main Properties

The objective of this modeling and verification work is (1) to show that the method implemented in the algorithm to predict trajectories and trigger alarms is adequate and does not lead to dangerous situations, and (2) to explore possible trajectory scenarios which lead to unacceptable risk. To this effect we created models of the algorithm and aircraft trajectories in PVS, created simulations in JAVA to graphically visualize the behavior and characteristics of the landing scenario, and derived in the computer algebra tool MuPAD equations of section 3.

5.1 Axioms on continuous mathematics

Before stating the main properties, it should be said that most of the proofs require reasoning on continuous mathematics. We have assumed some uninterpreted functions and axioms in PVS, for instance

```
sqrt(x:real) : {z:real | z2 = x and z ≥ 0}
```

```
sin_cos_sq_one : AXIOM
  ∀ (x:real): sind(x)2 + cosd(x)2 = 1
```

More involved properties, grounded on Equations 1 to 7, are also necessary, e.g.,

```
derivative_eq_zero_min : AXIOM
```

$$\forall (t_1, t_2: \text{real}): R(t_1 + \tau(t_1)) \leq R(t_1 + t_2)$$

`decrease_zero_to_tau` : AXIOM

$$\begin{aligned} \forall (t, t_1, t_2: \text{real}) : \\ \tau(t) \geq 0 \wedge t_2 \leq \tau(t) \wedge t_1 \leq t_2 \\ \Rightarrow \\ R(t+t_1) \geq R(t+t_2) \end{aligned}$$

`increase_tau_to_zero` : AXIOM

$$\begin{aligned} \forall (t, t_1, t_2: \text{real}) : \\ \tau(t) \leq 0 \wedge t_2 \geq \tau(t) \wedge t_1 \geq t_2 \\ \Rightarrow \\ R(t+t_1) \geq R(t+t_2) \end{aligned}$$

Axiom `derivative_eq_zero_min` states that at time t , $\tau(t)$ would be the time of closest separation between the aircraft. Axioms `decrease_zero_to_tau` and `increase_tau_to_zero` state that function R asymptotically decreases for times less than $\tau(t)$ and asymptotically increases for times greater than $\tau(t)$, respectively.

5.2 Finding a time prior to a collision

Our intention is to show that for all aircraft trajectories which lead to a collision and all initial states¹, an alarm is issued *time* seconds before a collision. In our formal development, we have found maximum and minimum bounds for the values of *time*.

In first place, we have proven that an alarm (it can be caution or warning) is triggered when the distance between the aircraft is within the alerting range (`alertRange`). This property holds independently of the values of any other state variables of the aircraft.

`alarm_when_alerting_distance` : THEOREM

$$\begin{aligned} \forall (\text{evader}, \text{intruder}: \text{State}) : \\ \text{alerting_distance}(\text{evader}, \text{intruder}) \Rightarrow \text{larcalert}(\text{evader}, \text{intruder}) \end{aligned}$$

The theorem above establishes the largest lower bound on the elapsed time between an alert and a collision that we have found so far. For an alerting distance of 1400 feet and an intruder ground speed of 250 feet per second this results in an alarm at least 4 seconds before collision.

An effort to prove that a caution is issued for a value of (`alertTime-1`) (`alertTime` being defined as 19 seconds) failed. Indeed, we have found a collision trajectory which allows two aircraft to fly from a 2500 feet y -separation to a distance of less than 1900 feet, without triggering an alarm 11 seconds before the collision.

¹ Recall from section 2 that initial states are when the aircraft are on their localizers.

```

move_2500_to_1900_no_alarm_before_11_seconds : THEOREM
  ∃ (intruder, evader: State, tr: [posnat → Bank], n: nat) :
    collision_scenario(intruder, evader, tr, n+11/tstep) ∧
    abs(y(intruder)-y(evader)) = 2500 ∧
    distance(intruder_trajectory(intruder, tr, n),
              evader_trajectory(evader, n)) < 1900 ∧
  ∀ (i: [0..n]):
    ¬ larcalert(evader_trajectory(evader, i),
                intruder_trajectory(intruder, tr, i))

```

Intruder and evader trajectories that satisfy the above property are $in_0 \dots in_n$, $ev_0 \dots ev_n$, where

```

in_0 = (# x := 860, y := 0, heading := 3, bank := 0 #)
ev_0 = (# x := 0, y := 2500, heading := 0, bank := 0 #)
tr = λ(n : posnat) : IF n ≤ 122 THEN 0 ELSE 45 ENDIF

```

and for $0 < i \leq n$,

```

in_i = intruder_trajectory(in_0, tr, i)
ev_i = evader_trajectory(ev_0, i)

```

By combining these theorems, we can state that (1) there is a trajectory for which an alarm will not sound before 11 seconds and (2) for all trajectories an alarm will sound at least 4 seconds before a collision. We believe that for all cases the largest time prior to a collision when the alarm will first sound is closer to 11 than to 4.

5.3 Closing the gap

In order to find a largest time prior to a collision, we need to find strong invariants on collision trajectories. Notice, for example, that for an intruder trajectory $in_0 \dots in_n$ and an evader trajectory $ev_0 \dots ev_n$, it cannot be the case that they lead to a collision incident at step n when $\text{distance}(in_0, ev_n) > R$, where

$$R = \text{collisionRange} + \text{intruderSpeed} \times n \times \text{tstep}.$$

Indeed, any intruder aircraft out of the circle of center $(x(ev_n), y(ev_n))$ and radius R , needs a larger time than $n \times \text{tstep}$ to reach any point of the circle of center $(x(ev_n), y(ev_n))$ and radio collisionRange . The property above can be expressed in PVS as follows.

```

collision_invariant : LEMMA
  ∀ (intruder, evader: State, tr: [posnat → Bank], n: nat) :
    collision_scenario(intruder, evader, tr, n)
  ⇒

```

```

 $\forall (i:[0..n]):$ 
  distance(intruder_trajectory(intruder,tr,i),
    evader_trajectory(evader,n))  $\leq$ 
  collisionRange+intruderSpeed $\times$ (n-i) $\times$ tstep

```

The proof of the invariant above requires the following lemma.

```

distance_invariant : LEMMA
 $\forall$  (intruder,evader:State, tr:[posnat  $\rightarrow$  Bank], n:nat) :
  distance(intruder_trajectory(intruder,tr,n),evader)  $\leq$ 
  distance(intruder_trajectory(intruder,tr,n+1),evader) +
  intruderSpeed $\times$ tstep

```

Lemma `distance_invariant` states that with respect to a fix evader position, one step in a straight trajectory leads farther than one step in any other direction.

We intend to use the above invariant and lemmas, together with properties derived from the physical trajectories, to find a bound greater than 4 seconds for any collision scenario. Under the assumption that the intruder bank angle is zero, we have proven that an alarm is issued 19 seconds before a collision. That property is expressed in PVS as follows

```

alarm_before_19_seconds_to_collision : THEOREM
  bank(intruder) = 0  $\wedge$ 
  collision_scenario(intruder,evader,straight_trajectory,m+38)
 $\Rightarrow$ 
  ( $\forall$  (i:subrange(m,m+38)):
    larcalert(intruder_trajectory(intruder,straight_trajectory,i),
      evader_trajectory(evader,i)))

```

We are trying to generalize the proof for an arbitrary trajectory and a time of 9 seconds.

6 Conclusion

Several case studies have been performed on the application of hybrid automata to the modeling of systems which include continuous and discrete domains. In particular, a simplified TCAS system was modeled in [9] using hybrid automata. That work focuses on establishing a hybrid model of the closed loop system formed by several aircrafts flying under TCAS assumptions. Although it is claimed that the model is suitable for formal analysis, there is no explicit attempt to automate the proof process. On the other hand, state exploration techniques have been used to analyze the system requirements specification of TCAS II written in RSML [7]; we refer for instance to [5, 2]. These works focus on the reactive aspect of the whole system.

In the work presented in this paper, we constructed a formal model of the kernel of an alerting algorithm and we studied its behavior with respect to a

model of collision trajectories. We defer the integration of the alerting algorithm with rest of the system, for example TCAS, for future research.

An abstract model of the algorithm and its properties were developed in the general verification system PVS. We complemented the prover capabilities with computer algebra tools. Indeed, differential equations, resulting from physical phenomena, were mechanically verified in MuPAD. Models of the algorithm and collision trajectories were implemented in Java. The implementation allowed us to graphically explore collision scenarios before performing rigorous attempts to prove properties.

Although we have confidence in the conjectures that have been declared as axioms, work is being performed [10] in the development of a PVS library on transcendental functions which complements a previous work on mathematical analysis in PVS [1]. Hence, it might be possible in the near future to replace the axiomatic definitions with theorems.

Lower and upper bounds for a time when an alarm will be issued before a collision were found. Our immediate goal, in the verification of the AILS algorithm, is to prove certain facts about the characteristics of the aircraft trajectories. We hope that these facts allow us to prove the adequacy of the alerting algorithm for a time large enough to avoid any possible collision incident.

References

1. B. Dutertre. Elements of mathematical analysis in PVS. In J. Von Wright, J. Grundy, and J. Harrison, editors, *Ninth international Conference on Theorem Proving in Higher Order Logics TPHOL*, volume 1125 of *Lecture Notes in Computer Science*, pages 141–156, Turku, Finland, August 1996. Springer Verlag.
2. W. Chan, R. Anderson, P. Beame, and D. Notkin. Improving efficiency of symbolic model checking for state-based system requirements. Technical Report TR-98-01-03, University of Washington, Department of Computer Science and Engineering, January 1998.
3. T. Doyle and F. McGee. Air traffic and operational data on selected u.s. airports with parallel runways. Technical Report NASA/CR-1998-207675, NASA, May 1998.
4. B. Fuchssteiner. *MuPAD User's Manual*. John Wiley and Sons, Chichester, New York, first edition, March 1996. Includes a CD for Apple Macintosh and UNIX.
5. M.P.E. Heimdahl and N.G. Leveson. Completeness and Consistency Analysis of State-Based Requirements. In *Proceedings of the 17th International Conference on Software Engineering*, pages 3–14, April 1995.
6. S. Koczo. Coordinated parallel runway approaches. Technical Report NASA-CR-201611, NASA, October 1996.
7. N.G. Leveson, M.P.E. Heimdahl, H. Hildreth, and J.D. Reese. Requirements specification for process-control systems. Technical Report ICS-TR-92-106, University of California, Irvine, Department of Information and Computer Science, November 1992.
8. A.M. Lind. Two simulation studies of precision runway monitoring of independent approaches to closely spaced parallel runways. Technical Report AD-A263433 ATC-190 DOT/FAA/NR-92/9, NASA, March 1993.

9. J. Lygeros and N. A. Lynch. On the formal verification of the TCAS conflict resolution algorithms. In *Proceedings 36th IEEE Conference on Decision and Control*, San Diego, CA, pages 1829–1834, December 1997. Extended abstract.
10. U. Martin and H. Gottliebsen. Computational logic support for differential equations and mathematical modeling. Personal communication, 2000.
11. S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, June 1992. Springer-Verlag.
12. L. Rine, T. Abbott, G. Lohr, D. Elliott, M. Waller, and R. Perry. The flight deck perspective of the NASA Langley AILS concept. Technical Report NASA/TM-2000-209841, NASA, January 2000.
13. RTCA. Minimum operational performance standards for traffic alert and collision avoidance system (TCAS) airborne equipment – consolidated edition. Guideline DO-185, Radio Technical Commission for Aeronautics, One McPherson Square, 1425 K Street N.W., Suite 500, Washington DC 20005, USA, 6 September 1990.
14. G. Wong. Development of precision runway monitor system for increasing capacity of parallel runway operations. *AGARD, Machine Intelligence in Air Traffic Management*, page 12, October 1993.