On the Formal Verification of the TCAS Conflict Resolution Algorithms [1]

John Lygeros and Nancy Lynch

Laboratory for Computer Science
Massachusetts Institute of Technology
545 Technology Square, Cambridge, MA 02139
lygeros, lynch@lcs.mit.edu

Abstract

TCAS is an on-board protocol for detecting conflicts between aircraft and providing resolution advisories to the pilots. Because of its safety-critical role the TCAS software should ideally be "verified" before it can be deployed. The verification task is challenging, due to the complexity of the TCAS code and the hybrid nature of the system. We show how the essence of this very complicated problem can be captured by a relatively simple hybrid model, amenable to formal analysis. We then outline a methodology for establishing conditions under which the advisories issued by TCAS are safe.

1 Introduction

The Traffic Alert and Collision Avoidance System (TCAS) [1, 2] is an on-board aircraft conflict detection and resolution algorithm. Its task is to monitor air traffic in the vicinity of the aircraft and provide the pilot with information about neighboring aircraft that may pose a threat and advisories on how to resolve these conflicts. TCAS is a complex, safety-critical system that should be tested, or, even better, formally verified before it can be deployed. The TCAS software was developed through a sequence of progressive refinements, starting with abstract, high-level specifications that got refined down to a Statechart description, pseudo-code and finally regular computer code. Part of the verification problem involves proving that each level in this process implements the high-level specifications. Motivated by this example (and other applications to software development for large scale systems) techniques have been developed [3] for systematically carrying out this process. In addition, one also needs to investigate the performance of the closed loop system formed when the proposed algorithm is coupled with the aircraft dynamics. So far the primary verification technique used in this context has been simulation [4]. Successful results in extensive simulations provide a certain level of confidence in the algorithm. More importantly, unsuccessful simulation runs point to situations where performance is insufficient and often suggest modifications to improve it.

We believe that formal methods may be useful in this setting. The advantage of formal analysis over simulation is that it provides absolute guarantees about the system performance, under a set of assumptions. In addition, formal analysis may prove to be more efficient in the long run, as the results may be modified to accommodate changes in the algorithm; in comparison, a large number of simulations may have to be reexamined even for minor changes. So far the application of formal methods to this problem has been limited, primarily because of the complexity of the algorithm. Much of this complexity, however, is due to considerations such as human factors, which should be secondary to safety. In this paper we show how one can extract a relatively simple protocol, that encapsulates the essence of the TCAS algorithm from the safety point of view. The model we derive (outlined in Section 2) is amenable to formal analysis. This is illustrated in Section 3, where some preliminary analysis of the safety of the algorithm is conducted. We hope that once the analysis for this simple model is complete the complexity of the original algorithm can be gradually reintroduced, allowing us to prove more involved safety properties.

The TCAS system is *hybrid*, involving both continuous and discrete dynamics. The former arise from the aircraft, the sensors and the pilot reaction and the latter from the thresholds and discrete message passing used by the TCAS algorithm[1]. Therefore any verification effort will have to involve hybrid techniques. Our work makes use of a combination of techniques from control theory and distributed algorithms to tackle the verification problem. The methodology presented here has been successfully applied to other safety-critical transportation systems, such as automated highways [6, 7], personal rapid transit systems [8], train gate controllers

[1] Research supported by ARPA under F19628-95-C-0118, by AFOSR under F49620-97-1-0337, by UTC under DTRS95G-0001-YR8 and by PATH, under MOU-238.

[1] There are also important probabilistic effects, arising from sensor noise, uncertainty in the pilot response etc. These effects will be mostly suppressed in our work. For a discussion of probabilistic analysis for this problem the reader is referred to [5].

[9, 10] and aircraft conflict resolution [11].

2 System Model

2.1 Overview of the TCAS System

In cases of potential conflict the TCAS system enters one of two levels of alertness. In the lower level the system issues a *Traffic Advisory* (TA), to inform the pilot of a potential threat, without providing any suggestions on how to resolve the situation. If the danger of collision increases a *Resolution Advisory* (RA) is issued, providing the pilot with a maneuver that is likely to resolve the conflict. In this study we do not address TA's, because of the uncertainty in the pilot response and the low level of hazard involved.

The RA's issued by the TCAS II 6.04A version currently in use are restricted to the vertical plane. Maneuvers involve either climbing or descending at one of a finite number of fixed rates. If both aircraft are TCAS equipped, the algorithm [1, 2] uses a symmetry-breaking communication protocol to uniquely determine the maneuver that each aircraft should follow to resolve the conflict. Once a decision is reached the maneuver is presented to the pilots and is not altered until the conflict is resolved. TCAS II 6.04A has been extensively tested in simulation [4] and in practice.

A newer TCAS II version that is currently being tested also allows for *reversals*. RA's are still restricted to the vertical plane, but TCAS may change the advisory during a conflict. This feature was added primarily because of nondeterminism in the pilot response. If one (or both) of the pilots chooses not to follow the advisory, the original RA may become unsafe. TCAS detects this and changes the RA if necessary. Clearly, this type of algorithm is in greater need of verification; potential problems include live-lock and unnecessary reversals.

Future TCAS versions (TCAS IV) will produce RAs both in the horizontal and the vertical plane, while still maintaining the possibility of reversals. Our approach may be even more useful in this case, to provide design guidelines for TCAS versions that are still at a conceptual stage.

2.2 Overview of the Modeling Formalism

Following [12], we view a hybrid automaton, A, as a dynamical system that describes the evolution of a finite collection of variables, V_A. Variables are typed; for each $v \in V_A$ let *type*(v) denote the type of v. For $Z \subseteq V_A$, a *valuation* of Z is a function that to each $v \in Z$ assigns a value in *type*(v). Let $\mathbf{Z}$ denote the set of valuations of Z; we refer to $s \in \mathbf{V_A}$ as the *state* of A. In this paper we assume that the evolution of the variables is over the set of times $T^{\geq 0} = \{t \in \mathbb{R} | t \geq 0\}$. The evolution of the variables involves both continuous and discrete dynamics. Continuous dynamics are encoded in terms of *trajectories* over V_A, that is functions that map intervals of time to $\mathbf{V_A}$. Discrete dynamics are encoded by *actions*; upon the occurrence of an action the state instantaneously "jumps" to a new value.

More formally, a *hybrid automaton*, A is a collection, $(U_A, X_A, Y_A, \Sigma_A^{in}, \Sigma_A^{int}, \Sigma_A^{out}, \Theta_A, \mathcal{D}_A, \mathcal{W}_A)$, of three disjoint sets U_A, X_A, and Y_A of variables (called *input*, *internal*, and *output variables*, respectively) three disjoint sets Σ_A^{in}, Σ_A^{int}, and Σ_A^{out} of actions (called *input*, *internal*, and *output actions*, respectively) a non-empty set $\Theta_A \subseteq \mathbf{V_A}$ of *initial states*, a set $\mathcal{D}_A \subseteq \mathbf{V_A} \times \Sigma_A \times \mathbf{V_A}$ of *discrete transitions* and a set $\mathcal{W}_A$ of *trajectories* over V_A, where $V_A = U_A \cup X_A \cup Y_A$ and $\Sigma_A = \Sigma_A^{in} \cup \Sigma_A^{int} \cup \Sigma_A^{out}$. Some technical conditions need to be imposed on the above sets to guarantee that the definitions are consistent; see [12] for a discussion.

Let *fstate*(w) (*lstate*(w)) denote the initial (final) state of a trajectory $w \in \mathcal{W}_A$ defined over a left (right) closed interval. An *execution*, α, of A is an alternating sequence $\alpha = w_0 a_1 w_1 a_2 w_2 \cdots$, with $w_i \in \mathcal{W}_A$ defined over a left closed time interval, $a_i \in \Sigma_A$, *fstate*$(w_0) \in \Theta_A$, and if w_i is not the last trajectory in α then its domain is a right-closed interval and $(\textit{lstate}(w_i), a_{i+1}, \textit{fstate}(w_{i+1})) \in \mathcal{D}_A$. If α is a finite sequence we assume it ends with a trajectory. An execution is called *finite* if it is a finite sequence and the domain of its final trajectory is right-closed. A state $s \in \mathbf{V_A}$ is called *reachable* if it is the last state of a finite execution.

Hybrid automata "interact" through shared variables and shared actions. Consider two automata A and B with $X_A \cap V_B = X_B \cap V_A = Y_B \cap Y_A = \emptyset$ and $\Sigma_B^{int} \cap \Sigma_A = \Sigma_A^{int} \cap \Sigma_B = \Sigma_A^{out} \cap \Sigma_B^{out} = \emptyset$. Under some mild technical assumptions, the *composition*, $A \times B$, of A and B can be defined as a new hybrid automaton with $U_{A \times B} = (U_A \cup U_B) \setminus (Y_A \cup Y_B)$, $X_{A \times B} = X_A \cup X_B$, $Y_{A \times B} = Y_A \cup Y_B$ (similarly for $\Sigma_{A \times B}$). $\Theta_{A \times B}$, $\mathcal{D}_{A \times B}$ and $\mathcal{W}_{A \times B}$ are defined so that the executions of $A \times B$ are executions of both A and B when restricted to the corresponding variables and actions.

A *derived variable* of A is a function on $\mathbf{V_A}$. Derived variables are used to simplify the system description and to facilitate the analysis. A *property* of A is a boolean derived variable. A property is *stable* if, whenever it is true at some state, it is also true at all states reachable from that state. A property is *invariant* if it is true at all reachable states. Typically properties will be shown to be stable or invariant by induction on the length of the executions. It is easy to see that:

Lemma 1 *If for all reachable states s, P is true at s implies that P is true for all s' such that there exists $a \in \Sigma_A$ with $(s, a, s') \in \mathcal{D}_A$ or there exists $w \in \mathcal{W}_A$ with right closed domain and* $\mathrm{fstate}(w) = s$ *and* $\mathrm{lstate}(w) = s'$*, then P is a stable property of A. If further P is true at all $s \in \Theta_A$, then P is an invariant property of A.*

In some places differential equations will be used to simplify the description of the set $\mathcal{W}_A$ (or at least parts of it). In this case, $\mathcal{W}_A$ is assumed to be populated by

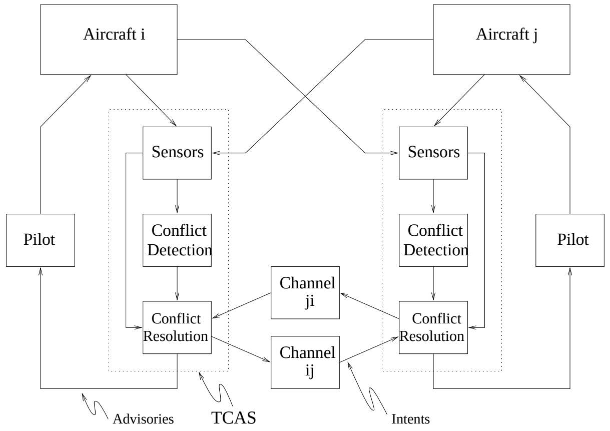

Figure 1: TCAS system components

all trajectories generated by the differential equation in the usual way. To simplify the description of $\mathcal{D}_A$, we will assign a *precondition* and an *effect* to each action. The precondition is a predicate on $\mathbf{V_A}$ while the effect is a predicate on $\mathbf{V_A} \times \mathbf{V_A}$. The corresponding transition can take place only from states that satisfy the precondition; moreover, the states before and after the transition should satisfy the effect.

2.3 The TCAS Model

We model TCAS by a composition of components (Figure 1). For each component a model was extracted from the TCAS documentation. The overall model is *closed*, in the sense that input variables and actions of one component are outputs of other components.

2.3.1 Aircraft Model: The system we consider consists of N aircraft, labeled $1, \ldots, N$. Each aircraft, i, is modeled by an HA, A_i. We assume $\Sigma_{A_i}^{in} = \Sigma_{A_i}^{int} = \Sigma_{A_i}^{out} = \emptyset$ and hence $\mathcal{D}_{A_i} = \emptyset$. Each aircraft is identified by a unique Mode_S number, stored in an output variable $Mode\_S_i \in \mathbb{N}$. Each aircraft may or may not be equipped with an altitude reporting transponder; if it is, it may also be equipped with TCAS. This hardware information is stored on an output variable $Equipment_i \in \{\text{None}, \text{Report}, \text{TCAS}\}$.

The physical movement of the aircraft is summarized by the trajectories of its position and velocity. Let $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $v_i = (v_i^x, v_i^y, v_i^z) \in \mathbb{R}^3$ and $a_i = (a_i^x, a_i^y, a_i^z) \in \mathbb{R}^3$ be the position, velocity and acceleration of the aircraft with respect to some fixed reference frame on the ground. We assume that all trajectories in $\mathcal{W}_{A_i}$ satisfy the differential equation:

$$\begin{bmatrix} \dot{p}_i(t) \\ \dot{v}_i(t) \end{bmatrix} = \begin{bmatrix} v_i(t) \\ a_i(t) \end{bmatrix} \tag{1}$$

We assume that the aircraft acceleration is under the direct control of the pilot and set $Y_{A_i} = \{Mode\_S_i, Equipment_i, p_i, v_i\}$, $U_{A_i} = \{a_i\}$ and $X_{A_i} = \emptyset$. The dynamics of equation (1) are very simple and ignore important aircraft characteristics such as the details of the aerodynamic forces, high frequency modes, the effect of structural controls and input constraints. Equation (1) should be sufficient in our case, however, as the maneuvers required by TCAS are rather mild.

2.3.2 Sensors: Each aircraft is equipped with sensors that return information about its state and the state of neighboring aircraft. The sensor of aircraft i is modeled by an HA, S_i with $U_{S_i} = \{p_j, v_j\}_{j=1}^N$. The output variables of S_i are estimates of the altitude, $h_i^j \in \mathbb{R}^+$, and vertical rate, $\dot{h}_i^j \in \mathbb{R}$, for all aircraft and the distance (range), $R_i^j \in \mathbb{R}^+$, and its rate $\dot{R}_i^j \in \mathbb{R}$ between aircraft i and each neighboring aircraft j. We set $\Sigma_{S_i}^{in} = \Sigma_{S_i}^{int} = \emptyset$.

The information that the sensors provide about the aircraft state is quantized spatially and sampled temporally. We assume that the output variables of the sensor automaton fall within an interval centered at the "correct" values dictated by the actual state of the system. Let n_A, n_{AR}, n_R and n_{RR} denote the width of the intervals for h_i^j, $\dot{h}_i^j$, R_i^j and $\dot{R}_i^j$ respectively. The output variables of the sensors are updated every T_s seconds, upon the occurrence of an output action $Sample_i$. An internal variable $T_i \in \mathbb{R}$ keeps track of the time that has elapsed since the last sample.

2.3.3 Conflict Detection: The role of the conflict detection automaton, D_i, is to determine whether neighboring aircraft pose a threat. The input variables of D_i are the output variables of S_i, as well as boolean variables $Threat_i^j$ which indicate whether the conflict resolution automaton is already aware of the threat. We set $\Sigma_{D_i}^{in} = \Sigma_{D_i}^{int} = \emptyset$ and $X_{D_i} = Y_{D_i} = \emptyset$.

Aircraft j is declared a threat by aircraft i upon the occurrence of an output action $Declare_i^j$ and ceases to be regarded as a threat upon the occurrence of an output action $Undeclare_i^j$. Two derived boolean variables, *Range_Test* and *Altitude_Test*, are used to determine the preconditions of these actions. The *Range_Test* encodes the conditions that the range and range rate need to satisfy for aircraft j to be declared a threat:

$$Range\_Test = (\dot{R}_i^j > 10ft/s \wedge \hat{R}_1) \vee (\dot{R}_i^j \leq 10ft/s \wedge \hat{R}_2)$$

where:

$$\hat{R}_1 = (R_i^j \leq \text{DM}) \wedge (R_i^j \dot{R}_i^j \leq \text{H1})$$
$$\hat{R}_2 = (R_i^j \leq 12nmi) \wedge \left(\frac{R_i^j - \text{DM}^2/R_i^j}{\min\{\dot{R}_i^j, -10ft/s\}} \leq \text{TR} \right)$$

The *Altitude_Test* is based on the predicted vertical separation at $\tau = |R_i^i / \min\{\dot{R}_i^j, -10ft/s\}|$, the "time of closest approach".

$$Altitude\_Test = \left| (h_i^i - h_i^j) - (\dot{h}_i^i - \dot{h}_i^j)\tau \right| \leq \text{ZT}$$

DM, H1, TR and ZT are TCAS parameters that depend on the current altitude.

At this stage we assume that j is declared a threat by i as soon as it "passes" both range and altitude tests. In practice a number of exceptions to this rule are introduced in the TCAS implementation, mostly to reduce the number of false alarms. Once declared a threat, j continues to be considered a threat until it fails the range test. At this point the action $\mathit{Undeclare}_i^j$ takes place.

2.3.4 Conflict Resolution: Conflict resolution is modeled by an HA, R_i, (Appendix A) with $U_{R_i} = Y_{S_i} \cup \{\mathit{Mode\_S}_j, \mathit{Equipment}_j\}_{j=1}^N$. The output variables of R_i are Threat_i^j and a resolution advisory for the pilot, consisting of a $\mathit{Sense}_i \in \{\text{Climb}, \text{Descend}, \perp\}$ and a $\mathit{Strength}_i \in \{$ -2000, -1000, -500, 0, 1500, 2500 $\}$ (in ft/min). The sense indicates whether i should try to pass above (Climb) or below (Descend) the intruding aircraft. $\mathit{Sense}_i = \perp$ (undefined) indicates that no action is needed. Strength provides a bound on the vertical speed to ensure sufficient vertical separation at time τ. R_i maintains two internal variables, the boolean $\mathit{Reversed}_i$ that keeps track of whether the sense selection has already been reversed during the current encounter and $\mathit{Intent\_Sent}_i^j \in \{\text{Climb}, \text{Descend}, \perp\}$ that keeps track of the last intent message sent by i to j. Intent messages can be thought of as "commands" to j as to which sense it should select[2].

R_i has no internal actions. Sense selection can happen when j is first declared a threat (upon the occurrence of input action $\mathit{Declare}_i^j$), whenever an intent message is received from another TCAS equipped aircraft (upon occurrence of input action $\mathit{Receive}_i^j(\text{dir})$ with dir $\in \{\text{Climb}, \text{Descend}\}$, and whenever new data comes in from the sensors (upon occurrence of input action Sample_i). The advisory is removed whenever the intruding aircraft ceases to be considered a threat (upon occurrence of input action $\mathit{Undeclare}_i^j$). i communicates its intent to j through an output action, $\mathit{Send}_i^j(\text{dir})$ with dir $\in \{\text{Climb}, \text{Descend}\}$.

Sense selection is based on the predicted vertical separation at time τ. Consider first the case of a climb advisory. To predict the vertical separation TCAS assumes that the intruding aircraft will maintain its current speed. If $\dot{h}_i^i \geq 1500$, TCAS assumes that the pilot will maintain the current climb rate. The vertical separation at time τ is then given by:

$$\Delta z(\text{Climb}) = (h_i^i - h_i^j) + (\dot{h}_i^i - \dot{h}_i^j)\tau$$

If $\dot{h}_i^i < 1500$ on the other hand, TCAS assumes that the pilot will respond to the advisory after a delay d by applying a constant vertical acceleration $a_i^z \equiv a$ until $\dot{h}_i^i = 1500 ft/min$. A similar expression produces the value of $\Delta z(\text{Climb})$ in this case. The climb separation is adequate if $\Delta z(\text{Climb})$ is above a threshold ALIM. The predicted separation in case of a descent advisory, $\Delta z(\text{Descend})$, can be similarly calculated.

Aircraft i issues an advisory against aircraft j for the first time when either the conflict detection automaton declares it a threat or when j sends an intent message (indicating that it has already issued an advisory against i). In the former case, i (the first of the two to detect the conflict) chooses an advisory sense based on a derived variable $\mathit{Indep\_Choice}$. If neither climb nor descent provide adequate separation, the one that produces the largest separation is chosen[3]. If one produces adequate separation but the other does not, the one that does is chosen. If both produce adequate separation preference is given to the non-crossing advisory (a climb if i is already higher and a descent if it is lower). If j has already issued an advisory, the complementary sense (encoded by the received intent) is typically chosen. The only exception is if i has a lower Mode_S number, the received intent is crossing (j is higher and has requested i to Climb or it is lower and has requested i to Descend) and i believes a non-crossing advisory is possible.

The sense may be reversed later on if, for example, one (or both) of the pilots thwarts the advisory. If j is not TCAS equipped, i reverses its advisory whenever it is predicted that the current advisory will not lead to adequate separation, while the reversed advisory will. The same is more or less true if j is TCAS equipped but i has a lower Mode_S number[4]. The only difference is that in this case i can only reverse once and then only if the current advisory is crossing. The new intent is communicated to j which is forced to change its advisory accordingly.

The advisory strength is updated every time new data becomes available. The choice of $\mathit{Strength}_i$ depends on the predicted vertical separation at time τ. The new strength is chosen according to a derived variable $\mathit{Strength\_Choice}$, which returns the smallest strength that will provide separation at least $ALIM$ at time τ. For example, if $\mathit{Sense}_i = \text{Climb}$, $(h_i^i - h_i^j) + (-500 - \dot{h}_i^j)\tau \geq ALIM$ and $(h_i^i - h_i^j) + (-1000 - \dot{h}_i^j)\tau < ALIM$ then $\mathit{Strength\_Choice} = -500$.

2.3.5 Communication Channel: Communication of intents is achieved through a communication channel automaton, C_{ij}. The automaton has an input action $\mathit{Send}_i^j(\text{dir})$, whose effect is to store the intent, dir, together with a time stamp in an internal multiset. The message is delivered (and removed from the multi set) upon occurrence of the output action $\mathit{Receive}_i^j(\text{dir})$. Delivery is guaranteed by at most d_{ij} time units from the time the message was sent.

2.3.6 Pilot: The pilot is modeled by an HA, P_i, with $U_{P_i} = \{\mathit{Sense}_i, \mathit{Strength}_i, \dot{h}_i^i\}$ and $Y_{P_i} = \{a_i\}$. The pilot may choose not to follow a particular advisory or to follow it after some delay. This information is stored

[2] In the TCAS code a Climb intent is referred to as a "Do not Descend" and a Descend intent as a "Do not Climb".

[3] It is assumed that conflict detection will take place early enough so that this case will never have to be exercised. We only include it here for completeness.

[4] The aircraft with the higher Mode_S number can not initiate a reversal.

in the boolean variable $Follow_i$ and the real variable d_i. The pilot automaton has no input or output actions. An internal action $New\_Advisory_i$ takes place whenever the advisory changes.

We assume that the pilot can apply a range of accelerations in each of the three directions, $a_i(t) \in [\underline{a}_i, \overline{a}_i] = [\underline{a}_i^x, \overline{a}_i^x] \times [\underline{a}_i^y, \overline{a}_i^y] \times [\underline{a}_i^z, \overline{a}_i^z]$. We also assume that the pilot tries to keep v_i^z in $[\underline{v}_i^z, \overline{v}_i^z]$. The width of the ranges reflects considerations such as passenger comfort and standard pilot practice. To ensure that all advisories can be followed we assume that $\underline{a}_i \leq -a < 0 < a \leq \overline{a}_i$, $[-2500, 2500] \subseteq [\underline{v}_i^z, \overline{v}_i^z]$ and $v_i^z(0) \in [\underline{v}_i^z, \overline{v}_i^z]$.

Whenever a new advisory comes in the pilot decides if it will be followed and chooses a delay $d_i \in [\underline{d}_i, \overline{d}_i]$. We assume that if the pilot chooses not to follow an advisory (or when none is present) he/she arbitrarily sets the vertical acceleration in the interval $[\underline{a}_i, \overline{a}_i]$. If the pilot chooses to follow the advisory, he/she is assumed to respond by at most d_i, by applying a constant vertical acceleration $a_i^z = a$ until $Strength_i$ is reached; a pilot is assumed to set $a_i^z = 0$ if the current vertical rate meets the advisory strength. One can show that:

Lemma 2 $v_i^z(t) \in \left[\underline{v}_i^z - \frac{n_{AR}}{2}, \overline{v}_i^z + \frac{n_{AR}}{2}\right]$ *for all* $t \geq 0$.

3 Verification Example

To illustrate how safety properties of the TCAS algorithm may be analyzed, consider a *pair of well-behaved aircraft*, defined as a system that satisfies:

Assumption 1 $N = 2$, $Equipment_i = \text{TCAS}$, $a_i^x(t) = a_i^y(t) = 0$ *and* $Follow_i(t) = \text{True}$ *for* $t \geq 0$ *and* $i = 1, 2$.

Let $\Delta x = x_1 - x_2$, $\Delta v_x = v_1^x - v_2^x$, etc. Consider the case where after a finite number of advisory changes, the TCAS algorithm converges to a fixed pair of advisories $(Sense_1, Strength_1)$ and $(Sense_2, Strength_2)$. Assume that the final advisories are "consistent":

Assumption 2 *There exists* $d_a \geq 0$ *such that for all* $t \geq d_a$ *and for* $i = 1, 2$, $\text{Sense}_i(t)$ *are constant,* $\text{Sense}_i(t) \neq \perp$ *and* $\text{Sense}_1(t) \neq \text{Sense}_2(t)$.

Without loss of generality assume that $Sense_1 = \text{Climb}$. Let $\Delta v_z^a = Strength_1 + Strength_2$ represent the minimum difference in vertical speed dictated by the advisory. One can show that:

Lemma 3 *There exists* $d \geq 0$ *such that for all* $t \geq d$, $a_i^z(t) = 0$ *and* $\Delta v_z(t) \geq \Delta v_z^a - n_{AR}$.

Let $\delta = d - t$ and consider the derived variable:

$$\begin{aligned} S_{\Delta v_z^a} &= \Delta z + \delta(\underline{v}_1^z - \overline{v}_2^z - n_{AR}) \\ &\quad -(\Delta v_z^a - n_{AR}) \frac{(\Delta x + \delta \Delta v_x)\Delta v_x + (\Delta y + \delta \Delta v_y)\Delta v_y}{\Delta v_x^2 + \Delta v_y^2} \\ S_{\Delta v_z^a} &= \Delta z - (\Delta v_z^a - n_{AR}) \frac{\Delta x \Delta v_x + \Delta y \Delta v_y}{\Delta v_x^2 + \Delta v_y^2} \end{aligned}$$

if $t \leq d$ and $t > d$ respectively.

Lemma 4 $(S_{\Delta v_z^a} \geq ALIM)$ *is a stable property of a pair of well behaved aircraft.*

Proof (sketch): None of the quantities in the right hand side of $S_{\Delta v_z^a}$ are affected by any of the system actions. Therefore, if $(S_{\Delta v_z^a} \geq ALIM)$ is true at the pre-state of an action it is also true at its post-state. Note that $S_{\Delta v_z^a}$ is continuous as a function of time and $\dot{S}_{\Delta v_z^a} = \Delta v_z - (\underline{v}_1^z - \overline{v}_2^z - n_{AR})$ if $t < d$ and $\dot{S}_{\Delta v_z^a} = \Delta v_z - (\Delta v_z^a - n_{AR})$ if $t > d$. In either case $\dot{S}_{\Delta v_z^a}(t) \geq 0$ (by Lemmas 2 and 3 respectively). Therefore, if $(S_{\Delta v_z^a} \geq ALIM)$ is true at the first state of a trajectory, it will also be true at the last state. ■

The quantity $S_{\Delta v_z^a}$ is related to the safety of the system. Consider the horizontal separation of the two aircraft $R_{xy} = \sqrt{\Delta x^2 + \Delta y^2}$. Consider the time $T = -\frac{\Delta x(0)\Delta v_x + \Delta y(0)\Delta v_y}{\Delta v_x^2 + \Delta v_y^2}$ and assume that $T > d$; this simply requires that the aircraft be far enough for the pilots to implement the advisory before the point of closest horizontal approach.

Theorem 1 *If* $S_{\Delta v_z^a}(0) \geq ALIM$ *then the vertical separation at the point of closest horizontal approach will be at least* $ALIM$.

Proof (sketch): At time T, R_{xy} achieves its minimum value. By Lemma 4, $S_{\Delta v_z^a}(0) \geq ALIM$ implies $(S_{\Delta v_z^a} \geq ALIM)$ is an invariant property. At time T, $\Delta x(T)\Delta v_x + \Delta y(T)\Delta v_y = 0$. Therefore, $S_{\Delta v_z^a}(T) \geq ALIM$ implies $\Delta z(T) \geq ALIM$, i.e. the vertical separation when the horizontal separation becomes minimum being at least $ALIM$. ■

4 Current & Future Research

Section 3 contains only a small part of the argument needed to show safety even for this simplified system. Assumption 2 clearly needs to be shown to be a property of the algorithm. This will complete a safety theorem for a pair of well-behaved aircraft. The proof then needs to be extended by relaxing Assumption 1: we need to investigate what happens if multiple aircraft are present, if the pilots accelerate in the x and y directions and if one of the pilots chooses not to follow the advisory. The last extension should also provide insight into the case of an unequipped threat. The analysis is complicated further in this case as multiple reversals are possible.

All proofs discussed so far will be based on the assumption that the model of Section 2 adequately captures the system. This model contains a number of simplifications, in the aircraft dynamics, the TCAS algorithm and the pilot response. These simplifications can be progressively removed. We hope that once a proof for the above nominal case is available, it can be extended to other cases, possibly using *abstraction relations*.

References

[1] Radio Technical Commission for Aeronautics, "Minimum operational performance standards for

TCAS airborn equipment”, Tech. Rep. RTCA/DO-185, RTCA, September 1990, Consolidated Edition.

[2] The MITRE Corporation, “TCAS II collision avoidance subsystem requirements specification”, 1996.

[3] Nancy Leveson, *SafeWare : system safety and computers*, Addison-Wesley, 1995.

[4] A.C. Drumm, “Lincoln laboratory evaluation of TCAS II logic version 6.04a”, Tech. Rep. ATC-240, Lincoln Laboratory, MIT, February 1996.

[5] James K. Kuchar, *A Unified Methodology for the Evaluation of Hazard Alerting Systems*, PhD thesis, Massacussets Institute of Technology, 1995.

[6] John Lygeros, Datta N. Godbole, and Shankar Sastry, “A verified hybrid controller for automated vehicles”, in *CDC*, 1996, pp. 2289–2294.

[7] Ekaterina Dolginova and Nancy Lynch, “Safety verification for automated platoon maneuvers: a case study”, in *Proceedings of HART97*, number 1201 in LNCS, pp. 154–170. Springer Verlag, 1997.

[8] H.B. Weinberg, Nancy Lynch, and Norman Delisle, “Verification of automated vehicle protection systems”, in *Hybrid Systems III*, number 1066 in LNCS, pp. 101–113. Springer Verlag, 1996.

[9] C. Heitmeyer and N. Lynch, “The generalized railroad crossing: A case study in formal verification of real-time systems”, in *Proc. ICCC Real-Time Systems Symposium*, San Juan, Puerto Rico, 1994.

[10] John Lygeros, Datta N. Godbole, and Shankar Sastry, “A game theoretic approach to hybrid system design”, in *Hybrid Systems III*, number 1066 in LNCS, pp. 1–12. Springer Verlag, 1996.

[11] George J. Pappas, Claire Tomlin, and Shankar Sastry, “Conflict resolution for multi-agent hybrid systems”, in *CDC*, 1996.

[12] Nancy Lynch, Roberto Segala, Frits Vaandrager, and H.B. Weinberg, “Hybrid I/O automata”, in *Hybrid Systems III*, number 1066 in LNCS, pp. 496–510. Springer Verlag, 1996.

A Conflict Resolution Automaton R_i

Data Types:

$\mathrm{Dir} = \{Climb, Descend\}$, $\mathrm{Dir}_\perp = \mathrm{Dir} \cup \{\perp\}$
$\mathrm{Strengths} = \{-2000, -1000, -500, 0, 1500, 2500\}$
$\mathrm{Aircraft} = \{1, \ldots, N\}$, $\mathrm{Others}_i = \mathrm{Aircraft} \setminus \{i\}$

Input Variables:

$Mode\_S_j \in \mathbb{N}$, $j \in$ Aircraft
$Equipment_j \in \{\mathrm{None}, \mathrm{Report}, \mathrm{TCAS}\}$, $j \in$ Aircraft
$h_i^j \in \mathbb{R}^+$ and $\dot{h}_i^j \in \mathbb{R}$ for $j \in$ Aircraft
$R_i^j \in \mathbb{R}^+$ and $\dot{R}_i^j \in \mathbb{R}$, for $j \in \mathrm{Others}_i$

Internal Variables:

$Reversed_i \in$ **Bool**, initially False
$Intent\_Sent_i^j \in \mathrm{Dir}_\perp$, $j \in \mathrm{Others}_i$, initially $\perp$

Output Variables:

$Sense_i \in \mathrm{Dir}_\perp$, initially $\perp$
$Threat_i^j \in$ **Bool**, $j \in \mathrm{Others}_i$, initially False
$Strength_i \in$ Stengths, initially 0

Derived Variables (see text):

$\Delta z(\mathrm{dir}) \in \mathbb{R}$ and $OK(dir) \in$ **Bool**, dir $\in$ Dir
$Indep\_Choice \in$ Dir
$Strength\_Choice \in$ Strengths

Input Actions:

$Declare_i^j$ and $Undeclare_i^j$ for $j \in \mathrm{Others}_i$
$Receive_i^j(\mathrm{dir})$, $j \in \mathrm{Others}_i$, dir $\in$ Dir
$Sample_i$

Output Actions:

$Send_i^j(\mathrm{dir})$, $j \in \mathrm{Others}_i$, dir $\in$ Dir

Discrete Transitions:

```
$Declare_i^j$:
    Effect: if $\neg Threat_i^j$ then
        $Threat_i^j$ := True; $Sense_i = Indep\_Choice$
$Undeclare_i^j$:
    Effect: if $Threat_i^j$ then
        $Threat_i^j$ := False; $Intent\_Sent_i^j = \perp$
        $Sense_i := \perp$; $Reversed_i$ := False
$Receive_i^j$(dir):
    Effect: if ($Mode\_S_i > Mode\_S_j$) then $Sense_i$ := dir
        if $\neg Threat_i^j$ then
            $Threat_i^j$ := True
            if ($Mode\_S_i < Mode\_S_j$) then
                if (dir = Climb $\wedge h_i^i \geq h_i^j$)
                then $Sense_i$ := Climb
                elseif (dir = Descend $\wedge h_i^i \leq h_i^j$)
                then $Sense_i$ := Descend
                else $Sense_i = Indep\_Choice$
$Sample_i$:
    Effect: if $Threat_i^j$ then
        if $Equipment_j \neq TCAS \wedge OK$(Climb)
        $\wedge \neg OK$(Descend) then $Sense_i$ := Climb
        if ($Equipment_j \neq TCAS \wedge \neg OK$(Climb)
        $\wedge OK$(Descend) then $Sense_i$ := Descend
        if $Equipment_j = TCAS \wedge Mode\_S_i < Mode\_S_j$
        $\wedge \neg Reversed$) then
            if $Sense_i$ = Descend $\wedge OK$(Climb)
            $\wedge \neg OK$(Descend) $\wedge h_i^i \geq h_i^j$ then
                $Sense_i$ := Climb; $Reversed_i$ := True
            if $Sense_i$ = Climb $\wedge \neg OK$(Climb)
            $\wedge OK$(Descend) $\wedge h_i^i \leq h_i^j$) then
                $Sense_i$ := Descend; $Reversed$ := True
        $Strength_i = Strength\_Choice$
$Send_i^j$(dir):
    Precondition:
        ($Sense_i$ = Climb $\wedge Intent\_Sent_i^j \neq$ Descend
            $\wedge$dir := Descend)$\vee$
        ($Sense_i$ = Descend $\wedge Intent\_Sent_i^j \neq$ Climb
            $\wedge$dir := Climb)
    Effect: $Intent\_Sent_i^j$ := dir
```

Trajectories:

Input variables follow arbitrary trajectories.
Internal and output variables remain constant.
Trajectories stop as soon as the precondition of $Send_i^j(\mathrm{dir})$ becomes true.