High-Level Modeling and Analysis of TCAS*

Carolos Livadas, John Lygeros, and Nancy A. Lynch

Laboratory for Computer Science
Massachusetts Institute of Technology
545 Technology Square
Cambridge, MA 02139
E-mail: `clivadas, lygeros, lynch@theory.lcs.mit.edu`

Abstract

In this paper, we demonstrate a high-level approach to modeling and analyzing complex safety-critical systems through a case study in the area of air traffic management. In particular, we focus our attention on the Traffic Alert and Collision Avoidance System (TCAS) [11, 12]; an on-board conflict detection and resolution system which alerts pilots to the presence of nearby aircraft that pose a mid-air collision threat and issues conflict resolution advisories. Due to the complexity of the TCAS software and the hybrid nature of the closed-loop system, the traditional testing techniques through simulation do not constitute a viable verification approach. To aid people in analyzing and designing such systems, we advocate defining high-level mathematical system models that capture the behavior not only of the software, but also of the airplanes, sensors, and pilots—that is, high-level hybrid system models. In particular, we show how the core components of this complex system can be captured by relatively simple Hybrid I/O Automata (HIOA) [9, 10], which are amenable to formal analysis. We then outline a methodology for establishing conditions under which the conflict resolution advisories issued by TCAS guarantee sufficient separation in altitude for aircraft involved in mid-air collision threats. Although our results are intended only as illustrations of high-level modeling and analysis techniques, the TCAS system models provide a foundation for study of a wide range of properties of the system's behavior.

*Research supported by ARPA under F19628-95-C-0118, by AFOSR under F49620-97-1-0337, by UTC under DTRS95G-0001-YR8, and by the PATH program, Institute of Transportation Studies, University of California, Berkeley, under MOU-319.

1. Introduction

In practice, once safety-critical systems are designed and implemented they undergo an extensive phase of testing through simulation. Unfortunately, this approach has several shortcomings. First, as systems get more complex and their behavior is enriched, the number of simulations required to provide a particular level of confidence increases exponentially. Second, regression testing dictates that when modifications to the system are performed, the complete set of simulations must be reconducted so as to make sure that the modifications did not compromise the performance of the system. Finally, and most importantly, the performance guarantees obtained through extensive simulations are not absolute; in fact, it is not even possible to provide conditional performance guarantees.

In this paper, we demonstrate a high-level approach for modeling and analyzing complex safety-critical systems for which a "certain level of confidence" in the system's performance is insufficient. We advocate obtaining precise mathematical models of all core components of the closed-loop system at hand and reasoning about the system's closed-loop performance at a high-level. The advantages of this approach are numerous. First, modeling all components of the system at hand provides a complete characterization of the behavior of the system — behavior not limited to the discrete or software aspects of the system. Second, modeling the system at a high-level of abstraction captures the intuitive understanding of the behavior of the system — often, this intuition is lost when systems are solely specified at extreme detail by the system designers. Thirdly, when analyzing the behavior of the system model, we can take advantage of formal notions of *composition* and *model refinement*. Finally, this approach has several advantages in terms of testing the correctness or performance of the system at hand. The task of producing the mathematical model of the system exposes both assumptions made by the system designers and design errors. Moreover, once a precise system model has been defined, corrections resulting from errors can easily be made without the high overhead of regression testing. Finally, under explicitly stated assumptions, we are able to obtain absolute system performance guarantees. The methodology presented here has already been successfully applied to various safety-critical transportation systems, such as automated highways [1, 6], personal rapid transit systems [4, 14], train gate controllers [2], aircraft conflict resolution for free flight [13], and the Center-TRACON Automation System (CTAS) [8]. There has also been some preliminary work on modeling the Traffic Alert and Collision Avoidance System (TCAS) [7]. Although our current work concentrates on the verification of systems that have already been designed and implemented, we believe our approach would be more useful during a system's design phase.

The Traffic Alert and Collision Avoidance System (TCAS) [11, 12] is an on-board aircraft conflict detection and resolution system used by all commercial aircraft. TCAS's task is to monitor air traffic in the vicinity of the aircraft, to alert the pilot to nearby aircraft that may pose a collision threat, and to propose maneuvers so as to resolve these conflicts. The TCAS software was developed through a sequence of progressive refinements: informal high-level specifications, Statechart descriptions, pseudo-code, and finally low-level computer code. Part of the verification problem involves proving that each level in this process implements the high-level specifications. The need to develop reliable software for large scale systems such as TCAS, has led to the development of techniques [3] for systematically carrying out this refinement. However, such techniques often begin modeling and specifying the system at relatively detailed levels. In so doing, the intuitive understanding of the behavior of the system is overshadowed by the details and technicalities in low-level specifications. Moreover, such techniques focus their attention only on the software aspects of the system at hand and, therefore, do not model the system as a whole. It is plausible that this approach neglects to model possibly hazardous aspects of the system's behavior.

The overall TCAS system is *hybrid*, involving both continuous and discrete dynamics; the former arise from the aircraft, sensors, and pilot reaction and the latter from the thresholds and discrete message passing among aircraft. In order to model both the discrete and continuous aspects of the behavior of TCAS, we use the mathematical formalism of hybrid I/O automata [9, 10]. The verification techniques we use involve a combination of techniques from control theory and distributed algorithms.

We proceed by summarizing the modeling formalism of hybrid I/O automata. In Section 3, we describe the TCAS system in more detail and present a HIOA model of each of the core components of the TCAS system. In Section 4, we present a conditional safety analysis of an idealized closed-loop system comprised of a pair of aircraft. The proof involves splitting the aircraft encounters into categories and defining safety conditions for each such category. Finally, by combining the per-category safety conditions we obtain overall safety conditions. In Section 5, we conclude and suggest future research directions. For detailed treatment of the work presented in this paper, we refer the reader to Ref. 5.

2. Hybrid I/O automata

The modeling formalism used in this paper is the hybrid I/O automaton (HIOA) model [9, 10]. Systems are modeled as collections of state machines that describe both discrete and continuous evolution of their state. The continuous evolution of the system is described in terms of *trajectories* over the state variables, *i.e.*, descriptions of how the system's state evolves with the passage of time. The discrete evolution of the system is described in terms of *actions* whereby the state of the system can instantaneously "jump" from one value to another. The state machines comprising the model of the system at hand "communicate" through shared variables and actions.

More formally, a hybrid I/O automaton A is a (possibly) infinite state model of a system involving both discrete and continuous behavior. The automaton $A = (U, X, Y, \Sigma^{in}, \Sigma^{int}, \Sigma^{out}, \Theta, \mathcal{D}, \mathcal{W})$ consists of three disjoint sets U, X, and Y of variables (*input*, *internal*, and *output* variables, respectively), three disjoint sets Σ^{in}, Σ^{int}, and Σ^{out} of *actions* (*input*, *internal*, and *output* actions, respectively), a non-empty set Θ of *initial states*, a set $\mathcal{D}$ of *discrete transitions*, and a set $\mathcal{W}$ of trajectories over V, where $\Sigma = \Sigma^{in} \cup \Sigma^{int} \cup \Sigma^{out}$ and $V = U \cup X \cup Y$. The set of all valuations of V, or equivalently the set of all states of A, is denoted by $\boldsymbol{V}$, or equivalently *states*(A). The *limit time* of a trajectory $w \in \mathcal{W}$, denoted by $w.ltime$, is defined to be the supremum of the domain of w, *dom*(w). We define the *first state* of w, denoted by $w.fstate$, to be the state $w(0)$. Moreover, if the domain of w is right-closed, then we define the *last state* of w, denoted by $w.lstate$, to be the state $w(w.ltime)$. A *hybrid execution fragment* α of A is a finite or infinite alternating sequence $w_0 a_1 w_1 a_2 w_2 \cdots$, where $w_i \in \mathcal{W}$, $a_i \in \Sigma$, and if w_i is not the last trajectory of α then w_i is right-closed and the discrete transition $(w_i.lstate, a_{i+1}, w_{i+1}.fstate)$ is in $\mathcal{D}$, or equivalently $w_i.lstate \xrightarrow{a_{i+1}}_A w_{i+1}.fstate$. If $w_0.fstate \in \Theta$ then α is a *hybrid execution* of A. A hybrid execution α of A is *finite* if it is a finite sequence and the domain of its final trajectory is a right-closed interval and *admissible* if $\alpha.ltime = \infty$. The *hybrid trace* of a hybrid execution fragment α of A, denoted by *h-trace*(α), is the sequence obtained by projecting α onto the external variables of A and subsequently removing all inert internal and environment actions. The set of *hybrid traces* of A, denoted by *h-traces*(A), is the set of hybrid traces that arise from all the finite and admissible hybrid executions of A.

A *superdense time* in an execution fragment α of A is a pair (i, t), where $t \leq w_i.ltime$. We totally order superdense times in α lexicographically. An *occurrence of* a state s *in* α is a triple (i, t, s) such that (i, t) is a superdense time in α and $s = w_i(t)$. State occurrences in α are ordered according to their superdense times.

Two HIOA A_1 and A_2 are *compatible* if $X_i \cap V_j = Y_i \cap Y_j = \Sigma_i^{int} \cap \Sigma_j = \Sigma_i^{out} \cap \Sigma_j^{out} = \emptyset$, for $i, j \in \{1, 2\}, i \neq j$. If A_1 and A_2 are compatible then their *composition* $A_1 \times A_2$ is defined to be the tuple $A = (U, X, Y, \Sigma^{in}, \Sigma^{int}, \Sigma^{out}, \Theta, \mathcal{D}, \mathcal{W})$ given by $U = (U_1 \cup U_2) - (Y_1 \cup Y_2)$, $X = X_1 \cup X_2$, $Y = Y_1 \cup Y_2$, $\Sigma^{in} = (\Sigma_1^{in} \cup \Sigma_2^{in}) - (\Sigma_1^{out} \cup \Sigma_2^{out})$, $\Sigma^{int} = \Sigma_1^{int} \cup \Sigma_2^{int}$, $\Sigma^{out} = \Sigma_1^{out} \cup \Sigma_2^{out}$, $\Theta = \{s \in \boldsymbol{V} \mid s \lceil V_1 \in \Theta_1 \ \wedge \ s \lceil V_2 \in \Theta_2\}$, and sets of discrete transitions $\mathcal{D}$ and trajectories $\mathcal{W}$ each of whose elements projects to discrete transitions and trajectories, respectively, of A_1 and A_2. Two HIOA A_1 and A_2 are *comparable* if they have the same external interface, *i.e.*, $U_1 = U_2$, $Y_1 = Y_2$, $\Sigma_1^{in} = \Sigma_2^{in}$, and $\Sigma_1^{out} = \Sigma_2^{out}$. If A_1 and A_2 are comparable, then $A_1 \leq A_2$ is defined to denote that the hybrid traces of A_1 are included in those of A_2; that is, $A_1 \leq A_2 \triangleq$ *h-traces*$(A_1) \subseteq$ *h-traces*(A_2). If $A_1 \leq A_2$, then we say that A_1 *implements* A_2.

The initial states, the discrete transitions, and the trajectories of a HIOA must satisfy several technical conditions which are omitted here. For a detailed presentation of the HIOA model, the reader is referred to Refs. 10 and 9.

3. The TCAS system

The TCAS system has evolved through a series of versions. TCAS I and TCAS II-6.04A have already been deployed and are currently standard in the US for all general aviation and commercial aircraft, respectively. A more powerful version (TCAS II-7) has been fully developed and has recently been tested through simulation. A number of future versions are still in a preliminary, conceptual stage. In this section we give a brief outline of the functionality of the TCAS II-7 system and show how its core components can be modeled as HIOA [7].

3.1. Overview of the TCAS system

In cases of potential mid-air collisions, 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 our analysis of the TCAS system, we model and address the performance of the TCAS system in view only of RAs.

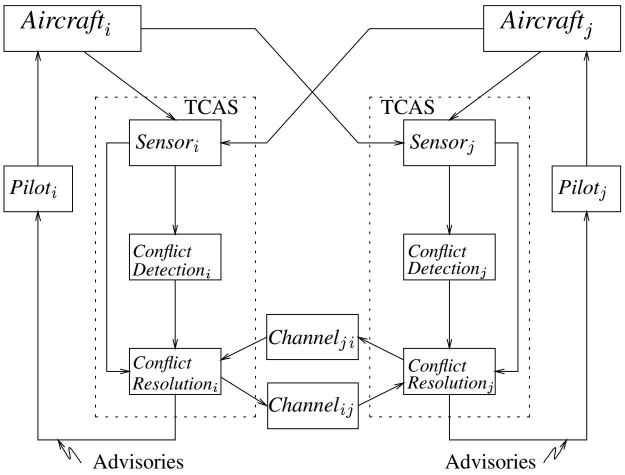

Figure 1. TCAS system block diagram

The RAs issued by TCAS II-7 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 TCAS system [11, 12] uses a symmetry-breaking communication protocol to uniquely determine the maneuver that each aircraft should follow to resolve the conflict. TCAS II-7 extends prior TCAS versions by allowing RA *reversals* — that is, TCAS may reverse RAs during a conflict.

It should be stressed that TCAS is a commercial product, intended for use on passenger aircraft. Therefore human factors issues, such as the comfort of the pilot, the passengers, and the air traffic controllers, also need to be considered. A substantial fraction of the TCAS code is devoted to such objectives; in our work we will mostly ignore these issues and concentrate only on the objective of safety.

3.2. Modeling TCAS using HIOA

As shown in Figure 1, we model the closed-loop TCAS system as a composition of smaller and simpler components. For each of the components we extract a model from the TCAS II-7 documentation. Even though the model allows the interaction of multiple aircraft, in our analysis we restrict our attention to conflicts involving only two aircraft.

3.2.1. Aircraft model. The system we consider consists of N aircraft, labeled $1, \ldots, N$. Each aircraft, $i \in \{1, \ldots, N\}$, is modeled by the HIOA $A_i = (U_{A_i}, X_{A_i}, Y_{A_i}, \Sigma_{A_i}^{in}, \Sigma_{A_i}^{int}, \Sigma_{A_i}^{out}, \Theta_{A_i}, \mathcal{D}_{A_i}, \mathcal{W}_{A_i})$. We assume there are no output or internal actions and no input action (other than the environment action), that is $\Sigma_{A_i}^{in} = \{e\}$,

$\Sigma_{A_i}^{int} = \Sigma_{A_i}^{out} = \emptyset$.

Each aircraft is identified by a unique integer, stored in an output variable $Mode\_S_i \in \mathbb{N}$, *i.e.*, $Mode\_S_i \neq Mode\_S_j$, for all $i, j \in \{1, \ldots, N\}$, $j \neq i$. 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 in an output variable $Equipment_i \in \{$None, Report, TCAS$\}$. All the aircraft considered in our analysis are either altitude-reporting or TCAS-equipped. The variables $Mode\_S_i$ and $Equipment_i$ are constant. The physical movement of the airplanes is summarized by the trajectories of their positions, $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, their velocities, $v_i = (v_{xi}, v_{yi}, v_{zi}) \in \mathbb{R}^3$, and their accelerations $a_i = (a_{xi}, a_{yi}, a_{zi}) \in \mathbb{R}^3$. We assume that all trajectories in $\mathcal{W}_{A_i}$ satisfy the differential equations $\dot{p}_i(t) = v_i(t)$ and $\dot{v}_i(t) = a_i(t)$. 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 $X_{A_i} = \emptyset$. As the maneuvers required by TCAS are rather mild and are unlikely to excite high order dynamics, induce input saturation, *etc.*, these simple aircraft dynamics are deemed sufficient for our analysis.

3.2.2. Sensors. Each aircraft is equipped with sensors that return information about its state and the state of neighboring aircraft. In particular, a number of hardware and software components contribute information to the TCAS algorithms, *e.g.*, the radio and pressure altimeters (which measure the aircraft altitudes), the radar (which measures the range to neighboring aircraft), the Mode_C or Mode_S transponders (which measure the altitude of neighboring aircraft), and the filtering algorithms (which "smooth" the received data and produce estimates of the range and altitude rates). The information that the sensors receive and provide about the aircraft state is quantized spatially and sampled temporally.

All such sensor hardware and software components of aircraft i are modeled by the HIOA $S_i = (U_{S_i}, X_{S_i}, Y_{S_i}, \Sigma_{S_i}^{in}, \Sigma_{S_i}^{int}, \Sigma_{S_i}^{out}, \Theta_{S_i}, \mathcal{D}_{S_i}, \mathcal{W}_{S_i})$. The input variables of S_i are the positions and velocities of all aircraft, $U_{S_i} = \{p_j, v_j\}_{j=1}^N$. The output variables of S_i are estimates of the altitude, $h_{ij} \in \mathbb{R}$, and vertical rate, $\dot{h}_{ij} \in \mathbb{R}$, for all aircraft and the distance (range), $R_{ij} \in \mathbb{R}^+$, and its rate, $\dot{R}_{ij} \in \mathbb{R}$, between aircraft i and each neighboring aircraft j. In other words, $Y_{S_i} = \{h_{ij}, \dot{h}_{ij}\}_{j=1}^N \cup \{R_{ij}, \dot{R}_{ij}\}_{j \neq i}$.

We assume that the output variables of the sensor automaton fall within an interval centered at the "correct" values corresponding to the actual state of the system. Let n_{Ai}, n_{ARi}, n_{Ri}, and n_{RRi} denote the size of the intervals for h_{ij}, $\dot{h}_{ij}$, R_{ij}, and $\dot{R}_{ij}$ respectively, and assume n_{Ai}, n_{ARi}, n_{Ri}, and n_{RRi} are constant. Moreover, let $[a \pm b]$, for $b \geq 0$, denote the interval $[a - b, a + b]$. The output variables of S_i are updated every T_{S_i} seconds (we assume that $T_{S_i} = 1$, throughout), upon the occurrence of an output action $Sample_i$. We set $\Sigma_{S_i}^{out} = \{Sample_i\}$. Upon the occurrence of $Sample_i$ the value of h_{ij} is reset to a new value $h_{ij} \in [z_j \pm n_{Ai}]$ (and similarly for the remaining output variables of S_i). The sensor automaton is assumed to have no input or internal actions ($\Sigma_{S_i}^{in} = \Sigma_{S_i}^{int} = \emptyset$).

3.2.3. Conflict detection. The role of the conflict detection module is to determine whether or not neighboring aircraft pose a threat. Aircraft are examined one at a time. The conflict detection module is modeled by the HIOA $D_i = (U_{D_i}, X_{D_i}, Y_{D_i}, \Sigma_{D_i}^{in}, \Sigma_{D_i}^{int}, \Sigma_{D_i}^{out}, \Theta_{D_i}, \mathcal{D}_{D_i}, \mathcal{W}_{D_i})$. The input variables of D_i are the output variables of S_i, as well as boolean variables $Threat_{ij}$ which indicate whether the conflict resolution module is already aware of the threat. Overall, $U_{D_i} = Y_{S_i} \cup \{Threat_{ij}\}_{j \neq i}$. D_i is assumed to have no input or internal actions ($\Sigma_{D_i}^{in} = \Sigma_{D_i}^{int} = \emptyset$) and no internal or output variables ($X_{D_i} = Y_{D_i} = \emptyset$). aircraft j is declared a threat by aircraft i upon the occurrence of an output action $Declare_{ij}$ and ceases to be regarded as a threat upon the occurrence of an output action $Undeclare_{ij}$, *i.e.*, $\Sigma_{D_i}^{out} = \{Declare_{ij}, Undeclare_{ij}\}_{j \neq i}$.

Two derived boolean variables, $Range\_Test_{ij}$ and $Altitude\_Test_{ij}$, are used to determine the preconditions of the $Declare_{ij}$ and $Undeclare_{ij}$ output actions. The $Range\_Test_{ij}$ encodes the conditions that the range and range rate need to satisfy for aircraft j to be declared a threat by aircraft i. The $Altitude\_Test_{ij}$ estimates the vertical separation at the estimated time to closest approach, τ_{ij}, where $\tau_{ij} = -R_{ij} / \min\{\dot{R}_{ij}, -10\}$. An intruding aircraft is declared a threat by TCAS as soon as it "passes" both range and altitude tests. Once the aircraft j is declared a threat by the aircraft i, the aircraft j remains a threat until it fails the range test; at which point, the $Undeclare_{ij}$ output action is scheduled.

3.2.4. Conflict resolution. For the conflict resolution module we restrict our attention to conflicts involving only two aircraft. The conflict resolution module is modeled by the HIOA $R_i = (U_{R_i}, X_{R_i}, Y_{R_i}, \Sigma_{R_i}^{in}, \Sigma_{R_i}^{int}, \Sigma_{R_i}^{out}, \Theta_{R_i}, \mathcal{D}_{R_i}, \mathcal{W}_{R_i})$. The input variables of R_i are the output variables of the sensor automaton and the $Mode\_S$ and equipment information from the aircraft automata, *i.e.*, $U_{R_i} = Y_{S_i} \cup \{Mode\_S_j, Equipment_j\}_{j=1}^N$. The output variables of R_i are the boolean $Threat_{ij}$ variables, indicating that aircraft i considers aircraft j a threat, and a resolution advisory for the pilot, consisting of a $Sense_i \in \{\text{Climb}, \text{Descend}, \perp\}$ and a $Strength_i \in$ Strengths $\equiv$ { -2000, -1000, -500, 0, 1500, 2500 } (units of ft/min). The $Sense_i$ indicates whether aircraft i should try to pass above ($Sense_i$ = Climb) or below ($Sense_i$ = Descend) the intruding aircraft. $Sense_i = \perp$ (undefined) indicates that no action is needed. In summary, the output variables of R_i are

$Y_{R_i} = \{Sense_i, Strength_i\} \cup \{Threat_{ij}\}_{j=1}^{N}$. R_i maintains three internal variables, the boolean *Reversed*$_i$ that keeps track of whether the sense selection has already been reversed during the current encounter, the boolean *Crossing*$_i$ which keeps track of whether the current RA implies that the aircraft would cross in altitude if they were to follow the RA, and *Intent_Sent*$_{ij} \in \{\text{Climb}, \text{Descend}, \perp\}$, which keeps track of the last intent message sent by aircraft i to aircraft j. The intent messages can be thought of as "commands" to aircraft j as to which RA it should issue. In summary, the internal variables of R_i are $X_{R_i} = \{Reversed_i, Crossing_i\} \cup \{Intent\_Sent_{ij}\}_{j \neq i}$.

R_i has no internal actions. Sense selection can happen when aircraft j is first declared a threat (upon the occurrence of the input action *Declare*$_{ij}$), whenever an intent message is received from another TCAS-equipped aircraft (upon the occurrence of an input action *Receive*$_{ij}$(dir), with dir $\in \{\text{Climb}, \text{Descend}\}$ being the intent of aircraft j), and whenever the system state is sampled by the sensors (upon the occurrence of input action *Sample*$_i$). The advisory is retracted whenever the intruding aircraft ceases to be considered a threat (upon the occurrence of the input action *Undeclare*$_{ij}$). In summary, the input actions of R_i are $\Sigma_{R_i}^{in} = \{Sample_i\} \cup \{Declare_{ij}, Receive_{ij}(\text{dir}), Undeclare_{ij}\}_{j \neq i}$. aircraft i sends its intentions to aircraft j through an output action, *Send*$_{ij}$(dir), where dir $\in \{\text{Climb}, \text{Descend}\}$ is the intent of aircraft i.

To predict the vertical separation at the estimated time to closest approach, τ_{ij}, TCAS assumes that the intruding aircraft, j, will maintain its current course, *i.e.*, $a_j \equiv [0 \quad 0 \quad 0]^T$. It also assumes that the pilot of aircraft i will respond to the advisory after a delay of exactly d (a parameter which depends on whether the advisory is new or a modification to an existing advisory), by applying a constant acceleration a, in the vertical direction until the desired vertical rate (given by *Strength*$_i$) is reached. If the current vertical speed meets the *Strength*$_i$ requirement or if τ_{ij} is less than the pilot delay, TCAS assumes that aircraft i will also maintain its current course.

More precisely, consider the derived variable, σ_i, which denotes the sense of the aircraft i, *i.e.*, $\sigma_i = 1$ if *Sense*$_i$ = Climb, $\sigma_i = 0$ if *Sense*$_i = \perp$, and $\sigma_i = -1$ if *Sense*$_i$ = Descend. For the sense $\sigma_i \in \{-1, 1\}$ and the resolution advisory strength *Strength* $\in$ Strengths, the vertical separation at the time of closest approach, denoted by the derived variable $SEP_{ij}(\sigma_i, Strength)$, is equal to $\sigma_i[(h_{ii} - h_{ij}) + (\dot{h}_{ii} - \dot{h}_{ij})\tau_{ij}]$, if $(\tau_{ij} \leq d) \vee (\sigma_i \dot{h}_{ii} \geq Strength)$, and $\sigma_i[(h_{ii} - h_{ij}) + (\dot{h}_{ii} - \dot{h}_{ij})d + (\sigma_i Strength - \dot{h}_{ij})(\tau_{ij} - d)]$, otherwise. The TCAS conflict resolution algorithm assumes that a Climb advisory will produce adequate separation at closest approach if $SEP_{ij}(1, 1500) \geq ALIM$, where *ALIM* is a system parameter that depends on the current altitude. Similarly, a Descend advisory is assumed to produce adequate separation if $SEP_{ij}(-1, 1500) \geq ALIM$. Note that in both cases the nominal strength is used. Throughout the following sections, we let $NStr = 1500$ and $\Delta NStr = 3000$.

aircraft i issues an advisory against aircraft j for the first time either when the conflict detection module declares j a threat or when aircraft i receives an intent message from aircraft j, indicating that aircraft j has already issued a RA against aircraft i. In the former case, aircraft i (the first of the two to detect the conflict) chooses an advisory sense independently of aircraft j. If neither a Climb nor a Descend resolution provides adequate separation, the one that produces the largest separation is chosen[1]. If one of the two produces adequate separation but the other one does not, the one that does is chosen. If both produce adequate separation preference is given to the non-crossing advisory (Climb if aircraft i is already higher or Descend if it is lower). If aircraft j has already issued an advisory, the complementary sense is typically chosen. The only exception is if aircraft i has a lower Mode_S number, the received intent is crossing (aircraft j is higher and has requested aircraft i to Climb or it is lower and has requested aircraft i to Descend) and aircraft i believes a non-crossing resolution is plausible.

The sense may be reversed later on if, for example, one (or both) of the pilots thwarts the advisory. If aircraft j is not TCAS equipped, or if it is but has a higher Mode_S number and the current advisory is crossing, aircraft i reverses its advisory whenever it is predicted that the current advisory will not lead to adequate separation, while the reversed advisory will. However, aircraft i can only reverse once; the internal variable *Reversed*$_i$ is used to enforce this requirement. The new intent is communicated to aircraft j which, due to it's higher Mode_S number, is forced to change its advisory accordingly.

The advisory strength is updated every time the state is sampled by *Sensor*$_i$, *i.e.*, upon occurrence of the *Sample*$_i$ action. The choice of *Strength*$_i$ again depends on the predicted vertical separation at time τ_{ij}. In particular, the conflict resolution automaton selects the weakest strength that guarantees sufficient vertical separation of the aircraft at the estimated time of closest approach.

3.2.5. Pilot. The pilot is modeled by the HIOA $P_i = (U_{P_i}, X_{P_i}, Y_{P_i}, \Sigma_{P_i}^{in}, \Sigma_{P_i}^{int}, \Sigma_{P_i}^{out}, \Theta_{P_i}, \mathcal{D}_{P_i}, \mathcal{W}_{P_i})$. The inputs variables are the selected advisory and the vertical rate of aircraft i, *i.e.*, $U_{P_i} = \{Sense_i, Strength_i, \dot{h}_{ii}\}$. The output variable is the acceleration of the aircraft, *i.e.*, $Y_{P_i} = \{a_i\}$. New advisories issued by TCAS are stored in an internal queue, *Adv_Q*$_i$. Each element of the queue contains the sense and strength of the corresponding advisory,

[1] We conjecture that conflict detection will take place early enough so that this case will never have to be exercised. We include it here mainly for completeness.

as well as upper and lower bounds on the time that may elapse before the pilot implements the respective advisory. The internal variables $Last\_Sense_i$ and $Last\_Strength_i$ store the last advisory issued by TCAS. The internal variables $Current\_Sense_i$ and $Current\_Strength_i$ store the last advisory implemented by the pilot; all "in-between" advisories are stored in Adv_Q_i.

The pilot automaton has no input or output actions. An internal action $New\_Advisory_i$ takes place whenever a new advisory is issued by TCAS. The effect of the action is to add the advisory to the tail of Adv_Q_i. The internal action $Implement\_Advisory_i$ takes place whenever an advisory from Adv_Q_i (*not necessarily the one at the head*) is implemented by the pilot. All earlier advisories are flushed from the queue and the pilot chooses non-deterministically whether to follow the advisory according to the value of the internal variable $Follow_i$. The implementation time for each advisory is guaranteed to be within interval $[\underline{d}_i, \overline{d}_i]$ from the time it gets issued by TCAS, unless it is "superseded" by the implementation of a later advisory.

We assume that the pilot can exert a range of accelerations in each of the three directions: $a_{xi}(t) \in [\underline{a}_{xi}, \overline{a}_{xi}]$, $a_{yi}(t) \in [\underline{a}_{yi}, \overline{a}_{yi}]$, and $a_{zi}(t) \in [\underline{a}_{zi}, \overline{a}_{zi}]$. We denote this compactly by $a_i \in [\underline{a}_i, \overline{a}_i]$. We also assume that the pilot tries to maintain the vertical velocity within a certain range, $[\underline{v}_{zi}, \overline{v}_{zi}]$. If the pilot chooses to follow an advisory, he/she is assumed to respond by applying a constant vertical acceleration $|a_{zi}| = a$ until the desired vertical rate is reached. A pilot is assumed to do nothing (set $a_{zi} = 0$) if he/she decides to follow the advisory and the current vertical rate meets the advisory strength. We assume that when no advisory is present or when the pilot chooses not to follow it, he/she arbitrarily sets the vertical acceleration in the interval $[\underline{a}_i, \overline{a}_i]$, in a way that will not cause the desired limits on vertical speed to be violated. More precisely, we assume that $\underline{a}_{zi} \leq -a < 0 < a \leq \overline{a}_{zi}$, $[-2500, 2500] \subseteq [\underline{v}_{zi}, \overline{v}_{zi}]$, and $v_{zi} \in [\underline{v}_{zi}, \overline{v}_{zi}]$.

3.2.6. Communication channel. Communication of intents is achieved through the communication channel HIOA C_{ij}. The automaton has an input action $Send_{ij}(\text{dir})$, for $dir \in Dir$, whose effect is to store the intent dir together with time stamps providing lower and upper bounds on the delivery time in an internal queue. The message is delivered (and removed from the queue) upon occurrence of the output action $Receive_{ji}(\text{dir})$, for $dir \in Dir$. The delivery time for each message is guaranteed to be within interval $[\underline{d}_{ij}, \overline{d}_{ij}]$ from the time the message was sent.

3.2.7. The closed-loop TCAS system. All the hardware and software related to the TCAS system are captured by the HIOA $TCAS_i$ which is the composition of S_i, D_i, and R_i with all variables in $Y_{S_i} \cup \{Threat_i^j\}_{j \neq i}$ and all actions in $\Sigma_{S_i}^{out} \cup \Sigma_{D_i}^{out}$ hidden. The interface of $TCAS_i$ with the outside world, *i.e.*, the pilots, the aircraft, and the communication channels, is through the input variables $U_{TCAS_i} = U_{S_i} \cup \{Mode\_S_j, Equipment_j\}_{j \neq i}$, output variables $Y_{TCAS_i} = \{Sense_i, Strength_i\}$, input actions $Receive_{ij}(\text{dir})$, for $dir \in Dir$, and output actions $Send_{ij}(\text{dir})$. A physical system of N aircraft, denoted by *PS*, is modeled as the composition of A_i, $TCAS_i$, P_i, and C_{ij}, for $i, j \in \{1, \ldots, N\}$, $i \neq j$, *i.e.*, $PS = \prod_{i,j \in \{1,\ldots,N\}, i \neq j} A_i \times TCAS_i \times P_i \times C_{ij}$.

4. Safety of a pair of well-behaved and TCAS-equipped aircraft

In this section, we present various safety conditions for a simplified version of a closed-loop system involving two aircraft. We begin by defining a pair of well-behaved and TCAS-equipped aircraft. Then, we proceed by categorizing the executions of this system and by providing safety conditions for each of the execution categories. We conclude by combining the per-category safety conditions into safety conditions for any execution of our system. In order to keep our analysis simple and tractable, we make several assumptions that seem restrictive. One should realize, however, that in this paper we are primarily interested in demonstrating our modeling and analysis approach. We defer the analysis of more complicated TCAS behavior to future research.

4.1. A pair of well-behaved and TCAS-equipped aircraft

In this section, we define a simple and idealized closed-loop system, *WBS*, that is comprised of only two aircraft. The aircraft are assumed to be TCAS-equipped, their sensors are assumed to be exact, pilots are assumed to always abide by the RAs issued by the TCAS system, and the aircraft are assumed to follow flight paths that have constant horizontal velocities. Moreover, in an effort to simplify the analysis of the TCAS system, we assume that the pilot can apply infinite acceleration in the vertical direction, *i.e.*, $a = \infty$, so as to be capable of attaining the resolution strength suggested by the TCAS system instantaneously. Although this assumption is not representative of reality, in effect it corresponds to analyzing a system where the pilot requires some additional delay in responding to a resolution advisory.

Thus, a pair of well-behaved and TCAS-equipped aircraft, *WBS*, is a system of 2 aircraft that satisfies $\text{Equipment}_i = TCAS$, $n_{Ai} = 0$, $n_{ARi} = 0$, $n_{Ri} = 0$, $n_{RRi} = 0$, $\text{Follow}_i = \texttt{True}$, $a_{xi} = a_{yi} = 0$, for $i \in \{1, 2\}$, and $a = \infty$. Let s denote the system state, $\overline{d}_p = \max\{\overline{d}_1, \overline{d}_2\}$, $\underline{d}_p = \min\{\overline{d}_1, \overline{d}_2\}$, $\overline{d}_c = \max\{\overline{d}_{12}, \overline{d}_{21}\}$, $\underline{d}_c = \min\{\overline{d}_{12}, \overline{d}_{21}\}$, $\Delta x = x_1 - x_2$, $\Delta y = y_1 - y_2$, $\Delta z = z_1 - z_2$, $\Delta v_x = v_{x1} - v_{x2}$, $\Delta v_y = v_{y1} - v_{y2}$, and $\Delta v_z = v_{z1} - v_{z2}$. Clearly, the sensors of *WBS* satisfy, for $i, j \in \{1, 2\}$, $h_{ij} = z_j$, $\dot{h}_{ij} = v_{zj}$, $R_{ij} = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$, and $\dot{R}_{ij} = dR_{ij}/dt$ at the times when the action $Sample_i$ is scheduled (similarly for j). As the "views" of the world available to the aircraft are accurate, we use z_i instead of h_{ii} and h_{ji}, R ($\dot{R}$) instead of R_{ij} and R_{ji} ($\dot{R}_{ij}$ and $\dot{R}_{ji}$) throughout the remaining sections, for $i, j \in \{1, 2\}$. To simplify the notation, we also assume that $\underline{v}_{z1} = \underline{v}_{z2}$ and $\overline{v}_{z1} = \overline{v}_{z2}$. Thus, we define the upper bound on the magnitude of relative vertical speed as $\overline{\Delta v}_z = -\underline{\Delta v}_z = \overline{v}_{z1} - \underline{v}_{z2} = \overline{v}_{z2} - \underline{v}_{z1}$. Lastly, let $\mathcal{S}$ denote the set of states of *WBS*, *i.e.*, $\mathcal{S} = states(WBS)$ and let *Admissible_Execs* denote the set of admissible executions of *WBS*, *i.e.*, $Admissible\_Execs = \{\alpha \in execs(WBS) \mid \alpha.fstate \in \Theta_{WBS}$ and $\alpha.ltime = \infty\}$.

We assume that the various parameters used by TCAS, such as *ALIM*, *etc*., remain constant throughout any execution of *WBS*. Without loss of generality, we assume that aircraft 1 is the high priority aircraft, *i.e.*, $Mode\_S_1 < Mode\_S_2$. In view of only considering TCAS resolutions that utilize nominal resolutions, we assume that the actions $Declare_{ij}$, $Sample_i$, and $Receive_{ij}(dir)$, for $i, j \in \{1, 2\}, i \neq j$ and $dir \in Dir$, only set the $Strength_1$ and $Strength_2$ variables to $1500 ft/min$, *i.e.*, $Strength_1 = Strength_2 = \mathrm{NStr} = 1500$, throughout any execution of *WBS*. Also, we assume that once pilots get alerted to a threat, they do not oppose the RA suggested by TCAS, *i.e.*, for any state $s \in \mathcal{S}$, $s.\sigma_i s.a_i \geq 0$, for $i \in \{1, 2\}$. Finally, we assume that once either an $Undeclare_{12}$ or $Undeclare_{21}$ action in scheduled by *WBS*, the aircraft no longer pose a threat to each other — that is, we assume that the TCAS system is conservative in undeclaring a potential threat and that it deems it appropriate to undeclare a threat when indeed it is safe to do so.

For any state $s \in \mathcal{S}$, let the *time to closest horizontal approach* be defined as $s.T = -(\Delta x \Delta v_x + \Delta y \Delta v_y)/(\Delta v_x^2 + \Delta v_y^2)$. For any execution $\alpha \in execs(WBS)$ and a state $s \in \mathcal{S}$, let $s \in \alpha$ denote that s is visited along α. By abusing the notation on state occurrences, when we refer to a state s occurring within an execution α of *WBS* we infer a particular state occurrence of the state s within α. Thus, the states visited along an execution are ordered, *i.e.*, for $s_1, s_2 \in \alpha$, we write $s_1 \leq s_2$ to denote that s_1 is visited by the finite prefix of α ending in s_2.

4.2. Agreement protocol

In Table 1, we define sets of states of *WBS* that represent incremental progress milestones of the agreement protocol used by TCAS to obtain consistent RAs when two aircraft are involved in a conflict.

Table 1. Milestone sets of protocol progress

$$
\begin{aligned}
&\textit{Local-Awareness} = \{s \in \mathcal{S} \mid \\
&\quad (s.\textit{Threat}_{12} \wedge s.\textit{Sense}_1 \neq \bot) \\
&\quad \vee (s.\textit{Threat}_{21} \wedge s.\textit{Sense}_2 \neq \bot)\} \\
&\textit{Local-Awareness-Sent} = \{s \in \mathcal{S} \mid \\
&\quad (s.\textit{Threat}_{12} \wedge s.\textit{Sense}_1 \neq \bot) \\
&\quad \vee (s.\textit{Threat}_{21} \wedge s.\textit{Sense}_2 \neq \bot \\
&\qquad \wedge s.\textit{Intent\_Sent}_{21} \neq \bot)\} \\
&\textit{Local-Resolution} = \{s \in \mathcal{S} \mid \\
&\quad s.\textit{Threat}_{12} \wedge s.\textit{Sense}_1 \neq \bot\} \\
&\textit{Local-Resolution-Sent} = \{s \in \mathcal{S} \mid s.\textit{Threat}_{12} \\
&\quad \wedge s.\textit{Sense}_1 \neq \bot \wedge s.\textit{Intent\_Sent}_{12} = \overline{s.\textit{Sense}_1}\} \\
&\textit{Global-Resolution} = \{s \in \mathcal{S} \mid \\
&\quad s.\textit{Threat}_{12} \wedge s.\textit{Threat}_{21} \wedge s.\textit{Sense}_2 \neq s.\textit{Sense}_1 \\
&\quad \wedge s.\textit{Sense}_1 \neq \bot \wedge s.\textit{Intent\_Sent}_{12} = \overline{s.\textit{Sense}_1} \\
&\quad \wedge s.\textit{Sense}_2 \neq \bot \wedge s.\textit{mset}_{12} = \emptyset\} \\
&\textit{Global-Resolution-Sent} = \{s \in \mathcal{S} \mid \\
&\quad s.\textit{Threat}_{12} \wedge s.\textit{Threat}_{21} \wedge s.\textit{Sense}_1 \neq s.\textit{Sense}_2 \\
&\quad \wedge s.\textit{Sense}_1 \neq \bot \wedge s.\textit{Intent\_Sent}_{12} = \overline{s.\textit{Sense}_1} \\
&\quad \wedge s.\textit{Sense}_2 \neq \bot \wedge s.\textit{Intent\_Sent}_{21} = \overline{s.\textit{Sense}_2} \\
&\quad \wedge s.\textit{mset}_{12} = \emptyset\} \\
&\textit{Global-Agreement} = \{s \in \mathcal{S} \mid \\
&\quad s.\textit{Threat}_{12} \wedge s.\textit{Threat}_{21} \wedge s.\textit{Sense}_1 \neq s.\textit{Sense}_2 \\
&\quad \wedge s.\textit{Sense}_1 \neq \bot \wedge s.\textit{Intent\_Sent}_{12} = \overline{s.\textit{Sense}_1} \\
&\quad \wedge s.\textit{Sense}_2 \neq \bot \wedge s.\textit{Intent\_Sent}_{21} = \overline{s.\textit{Sense}_2} \\
&\quad \wedge s.\textit{mset}_{12} = \emptyset \wedge s.\textit{mset}_{21} = \emptyset\}
\end{aligned}
$$

Lemma 1

1. *Global-Agreement* $\subseteq$ *Global-Resolution-Sent,*
2. *Global-Resolution-Sent* $\subseteq$ *Global-Resolution,*
3. *Global-Resolution* $\subseteq$ *Local-Resolution-Sent,*
4. *Local-Resolution-Sent* $\subseteq$ *Local-Resolution,*
5. *Local-Resolution* $\subseteq$ *Local-Awareness-Sent,*
6. *Local-Awareness-Sent* $\subseteq$ *Local-Awareness.*

The following lemma specifies the time that is needed to progress through the milestones of the TCAS agreement protocol, provided that neither RAs get undeclared, nor reversals occur. Once any aircraft gets alerted to the threat, the high priority aircraft gets alerted within $\overline{d}_c$ time units, a consistent RA is reached within $2\overline{d}_c$, and the protocol terminates within $3\overline{d}_c$. It is important to note that once a consistent RA is reached, both pilots may implement the RA within $\overline{d}_p$ time units provided they decide to follow the RA.

Thus, provided that the pilots follow RAs, $\overline{d}_c + \overline{d}_p$ time units after the high priority aircraft gets alerted to the advisory, both aircraft have already implemented the RAs. Let *Stable_Resolution_Frags* be the set of execution fragments consisting of all the execution fragments of *WBS* in which threats are not undeclared and reversals do not occur.

Lemma 2 *For any finite execution fragment α of WBS in Stable_Resolution_Frags such that α.fstate $\in$ Local-Awareness it is the case that:*

1. $\alpha.\mathit{ltime} > \overline{d}_c \implies \alpha.\mathit{lstate} \in$ *Local-Resolution,*
2. $\alpha.\mathit{ltime} > 2\overline{d}_c \implies \alpha.\mathit{lstate} \in$ *Global-Resolution,*
3. $\alpha.\mathit{ltime} > 3\overline{d}_c \implies \alpha.\mathit{lstate} \in$ *Global-Agreement,*
4. $\alpha.\mathit{ltime} > 3\overline{d}_c + \overline{d}_p \implies (\neg s.\mathit{Follow}_1 \vee (\sigma s.v_{z1} \geq \mathrm{NStr})) \wedge (\neg s.\mathit{Follow}_2 \vee (-\sigma s.v_{z2} \geq \mathrm{NStr}))$*, where $s = \alpha.\mathit{lstate}$ and $\sigma = 1$, if $s.\mathit{Sense}_1 = \mathit{Climb}$, and $\sigma = -1$, otherwise.*

4.3. Execution categorization

We partition the hybrid executions of *WBS* into the following four categories: 1) *Conflict_Free_Execs*, executions for which the TCAS protocol is not invoked; 2) *Non_Crossing_Execs*, executions where the TCAS protocol is initiated, a non-crossing RA is issued initially by aircraft 1 and is maintained until the conflict is over (once the high priority aircraft decides upon a non-crossing RA, its decision can not be reversed); 3) *Crossing_Execs*, executions where the TCAS protocol is initiated, a crossing RA is issued initially by aircraft 1 and is maintained until the conflict is over (once the aircraft cross in altitude and a sample action is scheduled, the advisory switches from being a crossing RA to a non-crossing RA); 4) *Reversing_Execs*, executions where the TCAS protocol is initiated, a crossing RA is issued initially by aircraft 1 and is reversed to a non-crossing RA before the aircraft cross in altitude and is maintained until the conflict is over (only possible to reverse out of a crossing RA). The sets *Conflict_Free_Execs*, *Non_Crossing_Execs*, *Crossing_Execs*, and *Reversing_Execs* are pairwise disjoint, and jointly comprise the set *Admissible_Execs*. We also define the set of safe executions of *WBS* as the set of executions in which the aircraft are sufficiently separated in altitude at closest horizontal approach, *i.e.*, *Safe_Execs* $= \{\alpha \in \mathit{Admissible\_Execs} \mid \forall\, s \in \alpha, (s.T = 0) \implies (|s.\Delta z| \geq \mathit{ALIM})\}$.

We assume that the TCAS system declares a conflict whenever there is a potential mid-air collision, *i.e.*, *Conflict_Free_Execs* $\subseteq$ *Safe_Execs*. We realize that this assumption is restricting, but we are interested in analyzing the performance of the TCAS system whenever it is engaged. Again, we defer the analysis of whether the TCAS system actually gets engaged in all potential mid-air collisions to future research.

4.4. Safety conditions

For an execution $\alpha \in$ *Conflict_Execs*, we define $\alpha.s_0 \in \alpha$ to be the state prior to which the high priority aircraft gets initially alerted to a potential mid-air collision threat. Also, by abusing notation, let $\alpha.T_0$ denote the time to closest horizontal approach from the state $\alpha.s_0$. Let D be an upper bound on the delay from the time in which the high priority aircraft gets alerted to a threat up to the time in which both aircraft implement consistent RAs, *i.e.*, $D = \overline{d}_c + \overline{d}_p$.

We assume that the bound D is larger than the bound d used by the TCAS algorithm to determine what type of RA to issue, *i.e.*, $D > d$. Moreover, we assume that the TCAS algorithm detects a conflict far enough in advance so as to have enough time to react, *i.e.*, the time of closest horizontal approach occurs more than D time units after the declaration of a mid-air collision threat by aircraft 1. In view of analyzing the correctness of the TCAS algorithm, this seems to be a reasonable assumption because we should not expect TCAS to be able to prevent collisions in cases where the system as a whole does not have sufficient time to decide upon and implement the RAs issued by TCAS.

4.4.1. Safety of non-crossing executions. In the case of non-crossing executions, we define a derived variable, P_{NC}, that denotes the minimum possible vertical separation of the aircraft at closest horizontal approach under the assumption that both aircraft implement a non-crossing advisory following an initial implementation delay of D time units. For $s \in \mathcal{S}$, let $s.P_{NC} = |s.\Delta z| + s.\underline{\Delta v}_{NC} D + 2\mathrm{NStr}(s.T - D)$, where $s.\underline{\Delta v}_{NC} = \min(2\mathrm{NStr}, \sigma s.v_{z1} + \max(\underline{v}_z, -\sigma s.v_{z2} - \overline{a}_z \overline{d}_c))$ with $\sigma = \mathrm{sign}(s.v_{z1} - s.v_{z2})$. Intuitively, the worst-case vertical separation at the time of closest horizontal approach is the initial altitude separation, minus potential losses during the implementation delay, *i.e.*, the time it takes both aircraft to agree to and to implement the non-crossing advisory, plus the separation that is gained by following the RA at nominal strength once it is implemented. Moreover, let *Non_Cross_Safe* $= \{s \in \mathcal{S} \mid s.P_{NC} \geq \mathit{ALIM}\}$ be the set of states of *WBS* from which the choice and implementation of a non-crossing RA is guaranteed to result in adequate separation in altitude at closest horizontal approach.

Lemma 3 *If $\alpha \in$ Non_Crossing_Execs and $\alpha.s_0 \in$ Non_Cross_Safe, then $\alpha \in$ Safe_Execs.*

4.4.2. Safety of crossing executions. In the case of crossing executions, we define a derived variable, P_C, that denotes the minimum possible vertical separation of the aircraft at closest horizontal approach under the assumption that both aircraft implement a crossing advisory following an initial implementation delay of D time units. For $s \in \mathcal{S}$, let $s.P_C = -|s.\Delta z| + s.\underline{\Delta v}_C D + 2\text{NStr}(s.T - D)$, where $s.\underline{\Delta v}_C = \min(2\text{NStr}, -\sigma s.v_{z1} + \max(\underline{v}_z, \sigma s.v_{z2} - \overline{a}_z \overline{d}_c))$ with $\sigma = \text{sign}(s.v_{z1} - s.v_{z2})$. Intuitively, the worst-case vertical separation at the time of closest horizontal approach is the initial altitude separation, minus potential losses during the implementation delay, *i.e.*, the time it takes both aircraft to agree to and to implement the crossing advisory, plus the separation that is gained by following the RA at nominal strength once it is implemented. Moreover, let *Cross_Safe* $= \{s \in \mathcal{S} \mid s.P_C \geq \mathit{ALIM}\}$ be the set of states of *WBS* from which the choice and implementation of a crossing RA that is carried out to completion is guaranteed to result in adequate separation in altitude at closest horizontal approach.

Lemma 4 *If $\alpha \in$ Crossing_Execs and $\alpha.s_0 \in$ Cross_Safe, then $\alpha \in$ Safe_Execs.*

4.4.3. Safety of reversing executions. In the case of reversing executions, a reversal can occur in two distinct parts of the execution, namely, prior to and on or after the time in which the crossing execution gets implemented. We analyze each of these cases separately.

First, we analyze the case in which the reversal occurs prior to the implementation of the crossing resolution. In this case, the reversal occurs prior to D time units after the first declaration of the threat and gets implemented by both aircraft within D of the time of the sense reversal of aircraft 1. Thus, the time elapsing from the state $\alpha.s_0$ to the latest possible time in which the non-crossing RA gets implemented is $2D$. We define a derived variable, $P_R^<$, that denotes the minimum possible vertical separation of the aircraft at closest horizontal approach. For all $s \in \mathcal{S}$, let $s.P_R^< = |s.\Delta z| + s.\underline{\Delta v}_R^<(2D) + 2\text{NStr}(s.T - 2D)$, where $s.\underline{\Delta v}_R^< = \min(2\text{NStr}, \max(\underline{v}_z, \sigma s.v_{z1} - \overline{a}_z D) + \max(\underline{v}_z, -\sigma s.v_{z2} - \overline{a}_z(D + \overline{d}_c)))$ with $\sigma = \text{sign}(s.v_{z1} - s.v_{z2})$, *i.e.*, σ is the non-crossing RA sense of aircraft 1 is state s. The intuitive understanding of the derived variable $P_R^<$ involves realizing that the worst-case would be to decide to reverse at the latest possible point in time, *i.e.*, D time units after the initial threat declaration, which would in turn allow the least amount of time for the new advisory to attain the necessary vertical separation at closest horizontal approach. It follows that the state s is safe in this type of an execution if $s.P_R^< \geq \mathit{ALIM}$.

Second, we analyze the case in which a reversal occurs subsequent to the implementation of the crossing advisory by both aircraft, *i.e.*, the reversal occurs in a state in which both aircraft are following the crossing resolution advisory. Let s be the state in which the first alert declaration is scheduled by aircraft 1, s' be the first state in which both aircraft are implementing the crossing advisory, and s'' be the state in which the sample action resulting in the reversal is scheduled. Note that the maximum time between the occurrence of state s and s' is D. Throughout this section we will refer to the time to closest horizontal approach from state s as T, instead of $s.T$.

A reversal is warranted only if the crossing advisory is unsafe and the non-crossing advisory is safe. Letting the crossing and non-crossing senses of aircraft 1 in state s be denoted by $-\sigma$ and σ, respectively, the conditions that would dictate the reversal in state s'' according to the TCAS specifications are $\mathit{SEP}_{12}(-\sigma, \text{NStr}) < \mathit{ALIM}$ and $\mathit{SEP}_{12}(\sigma, \text{NStr}) \geq \mathit{ALIM}$. Moreover, since we are assuming that the crossing advisory has already been implemented, it follows that $-\sigma v_{z1} \geq \text{NStr}$ and $\sigma v_{z2} \geq \text{NStr}$. Moreover, since reversals can only be considered while the difference in altitude opposes the current RA sense, it follows that $-\sigma \Delta z < 0$. Combining the above conditions, it can be shown that in order for an aircraft to reverse out of an implemented crossing RA, $-\sigma$, in state s'' it the case that $\sigma \Delta z \geq \mathit{ALIM}$, *i.e.*, the aircraft altitude separation in state s'' must be greater than or equal to *ALIM*. From this condition, we obtain conditions on the latest possible time, $\overline{T}_R$, at which a reversal can occur. For any state s in which the first alert declaration is scheduled, the latest point in time at which the reversal could occur corresponds to the latest point in time that the inequality $\sigma \Delta z \geq \mathit{ALIM}$ could be violated, *i.e.*, $\overline{T}_R$ is bounded by the inequality $-|\Delta z| + s.\underline{\Delta v}_C D + 2\text{NStr}(\overline{T}_R - D) \leq -\mathit{ALIM}$. Solving for $\overline{T}_R$ we get $\overline{T}_R \leq (-\mathit{ALIM} + |\Delta z| - s.\underline{\Delta v}_C D + 2\text{NStr}D)/(2\text{NStr})$. If the value of $\overline{T}_R$ turns out to be negative, then it follows that the reversal could never have been scheduled in the first place. In such cases, we assume that the execution is safe.

In order for such a state to be safe, the worst case trajectory must be safe; that is, given an altitude separation of *ALIM*, the worst case would be to follow a trajectory of minimum vertical velocity until the non-crossing RA gets implemented and then carry on with a nominal strength non-crossing RA. We define a derived variable, $P_R^{\geq}$, that denotes the minimum possible vertical separation of the aircraft at closest horizontal approach given that aircraft 1 reverses its sense $\overline{T}_R$ time units after its initial declaration of a threat. For all $s \in \mathcal{S}$, let $s.P_R^{\geq} = \mathit{ALIM} + s.\underline{\Delta v}_R^{\geq} D + 2\text{NStr}(T - \overline{T}_R - D)$, where $s.\underline{\Delta v}_R^{\geq} = \min(2\text{NStr}, \max(\underline{v}_z, \sigma s.v_{z1} - \overline{a}_z \overline{T}_R) + \max(\underline{v}_z, -\sigma s.v_{z2} - \overline{a}_z(\overline{T}_R + \overline{d}_c)))$ with $\sigma = \text{sign}(s.v_{z1} - s.v_{z2})$, *i.e.*, σ is the non-crossing RA sense of aircraft 1. Plugging in the value for $\overline{T}_R$ and simplifying, we get $s.P_R^{\geq} = 2\mathit{ALIM} - |\Delta z| + (s.\underline{\Delta v}_C + s.\underline{\Delta v}_R^{\geq})D + 2\text{NStr}(T - 2D)$. It follows that the state s is safe in this type of an execution if $s.P_R^{\geq} \geq \mathit{ALIM}$.

Thus, let *Reverse_Safe* $= \{s \in \mathcal{S} \mid (s.P_R^< \geq \mathit{ALIM}) \wedge ((s.P_R^{\geq} \geq \mathit{ALIM}) \vee (\overline{T}_R < D))\}$ be the set of states from which any type of reversing execution results in sufficient altitude separation at closest horizontal approach. Our approach is to take the conjunction of the safety properties for the two types of reversing executions. When doing so however, we must be cautious because the second condition is only valid if the reversal occurs after the crossing advisory gets implemented.

Lemma 5 *If* $\alpha \in$ *Reversing_Execs and* $\alpha.s_0 \in$ *Reverse_Safe, then* $\alpha \in$ *Safe_Execs.*

4.5. Safety of executions in summary

In the previous section, we derived safety conditions for each of the categories of executions of *WBS*. In particular, we defined sets of states from which the choice and execution of a non-crossing, crossing, and reversing execution, respectively, would result is sufficient altitude separation at closest horizontal approach. In this section, we provide three ways in which these safety conditions can be combined in order to provide overall safety conditions.

4.5.1. Conjunction of per-category safety properties. In this section, we define our overall safety property to be the conjunction of the per-category safety properties. Albeit simple, this approach is conservative since in order for an execution of *WBS* to be deemed safe, it must satisfy the safety conditions of all types of executions.

Theorem 6 *If* $\alpha \in$ *Conflict_Execs and* $\alpha.s_0 \in$ *Non_Crossing_Safe* $\cap$ *Crossing_Safe* $\cap$ *Reverse_Safe, then* $\alpha \in$ *Safe_Execs.*

4.5.2. Isolating non-crossing executions. In this section, we remove some of the conservatism of Theorem 6 by isolating the non-crossing execution advisories. The inherent bias in the TCAS system toward non-crossing RAs dictates that the majority of RAs issued by TCAS will be non-crossing RAs. Thus, by isolating the set of non-crossing executions and distinguishing them from the crossing and reversing executions, we obtain less conservative results.

We begin by defining a necessary condition for initially choosing a crossing advisory. In order for a crossing advisory to be chosen, at the point in time of the advisory declaration by the high priority aircraft, the TCAS algorithm should deem it appropriate. According to the conflict resolution automaton R_i, the declaration occurs through either a *Declare*$_{12}$ or a *Receive*$_{12}$(dir) action, where dir $\in$ Dir. In the case of a *Declare*$_{12}$ action, the crossing advisory is selected only when the estimated altitude separation at closest approach resulting from a crossing and a non-crossing advisory is sufficient and insufficient, respectively. In the case of a *Receive*$_{12}$(dir) action, where dir $\in$ Dir, a crossing advisory is chosen only in the case when the RA suggested by the low priority aircraft is a crossing RA and the high priority aircraft agrees with it. For any execution α of *WBS*, we let the state from which the high priority aircraft gets alerted to the threat by a *Declare*$_{12}$ or a *Receive*$_{12}$(dir) action, where dir $\in$ Dir, be denoted by $\alpha.s_0$ and the state following the scheduling of the action as $\alpha.s_0'$. Moreover, let $s.C = \exists\, \sigma \in \{1,-1\}$ such that $(\sigma s.\Delta z < 0) \wedge (Sep_{12}(\sigma, \mathrm{NStr}) \geq ALIM \wedge Sep_{12}(-\sigma, \mathrm{NStr}) < ALIM)$ be the derived variable that denotes whether the choice of a crossing RA is possible from the perspective of aircraft 1. Since $s.C$ is a necessary condition for the aircraft to engage in a crossing RA, the negation of this condition is a sufficient condition for choosing a non-crossing RA. Let *Cross_Impossible* $= \{s \in \mathcal{S} \mid s.C = \texttt{False}\}$ be the set of states of *WBS* from which it is impossible for aircraft 1 to choose a crossing RA.

Lemma 7 *If* $\alpha \in$ *Conflict_Execs and* $\alpha.s_0 \in$ *Cross_Impossible, then* $\alpha \in$ *Non_Crossing_Execs.*

Theorem 8 *If* $\alpha \in$ *Conflict_Execs and* $\alpha.s_0 \in$ $($*Cross_Impossible* $\cap$ *Non_Cross_Safe*$) \cup ($*Non_Cross_Safe* $\cap$ *Cross_Safe* $\cap$ *Reverse_Safe*$)$*, then* $\alpha \in$ *Safe_Execs.*

4.5.3. Aircraft close in altitude. In this section, we specify safety conditions for a set of executions that are defined parametrically with respect to the altitude separation of the aircraft at the point in time when the conflict is initially declared by aircraft 1. This approach was suggested to us by engineers actively involved in the design and analysis of the TCAS system. The intuition behind this approach is that crossing advisories will most likely be chosen when the aircraft are close in altitude, so it is very useful to consider and reason about the performance of TCAS is such executions.

If the aircraft are close in altitude when the threat gets declared, then the type of execution to be carried out is finalized by the time the aircraft would cross in altitude had a crossing advisory been chosen initially and carried out to completion. On one hand, if a non-crossing RA is declared initially by aircraft 1, the execution type is known immediately. On the other hand, if a crossing RA is declared initially by aircraft 1, the RA is either carried out to completion, or reversed before the aircraft cross in altitude. Thus, by the time the aircraft cross in altitude, it is known whether the execution is crossing or reversing. Let *Close-in-Alt_Execs* $= \{\alpha \in$ *Conflict_Execs* $\mid s = \alpha.s_0, |s.\Delta v_z| \leq K\}$ be the set of executions of *WBS* in which the aircraft are separated in altitude by at most $K ft$ when aircraft 1 is alerted to a threat.

The safety condition for executions in the set *Close-in-Alt_Execs* is obtained in a very similar fashion to the way the safety condition is obtained for the second type of reversing executions. In particular, the separation obtained by any type of execution is bounded from below by the separation obtained by a reversing execution in which aircraft 1 reverses its sense just as the aircraft cross in altitude. We denote the latest possible crossing time by $\overline{T}_C$. For any state $s \in \mathcal{S}$ in which the threat declaration is scheduled by aircraft 1, the latest point in time at which the aircraft could cross in altitude corresponds to the latest point in time that the inequality $\sigma\Delta z \geq 0$ could be violated, *i.e.*, $\overline{T}_C$ is bounded by the inequality $-K + s.\underline{\Delta v}_C D + 2\text{NStr}(\overline{T}_C - D) \leq 0$. Solving for $\overline{T}_C$ we get $\overline{T}_C \leq (K - s.\underline{\Delta v}_C D + 2\text{NStr}D)/(2\text{NStr})$. If $-K + s.\underline{\Delta v}_C D > 0$, then the above calculation for $\overline{T}_C$ will result in a negative value, which implies that the reversal would have occurred prior to D time units following the initial declaration of a threat by aircraft 1. Thus, in such cases as an upper bound on the delay in reversing we can use the value of D time units.

In order for a state s to be safe, the worst case altitude separation would be obtained by an execution in which the aircraft follow a trajectory of minimum vertical velocity until the reversal gets implemented and then carry on with nominal strength. We define a derived variable, P, that denotes the minimum possible vertical separation of the aircraft at closest horizontal approach given that aircraft 1 starts to implement a non-crossing advisory $\max(D, \overline{T}_C)$ time units after the initial threat declaration by aircraft 1. For all $s \in \mathcal{S}$, let $s.P = s.\underline{\Delta v}_R D + 2\text{NStr}(T - \max(D, \overline{T}_C) - D)$, where $s.\underline{\Delta v}_R = \max(\underline{v}_z, \sigma s.v_{z1} - \overline{a}_z \max(D, \overline{T}_C)) + \max(\underline{v}_z, -\sigma s.v_{z2} - \overline{a}_z(\max(D, \overline{T}_C) + \overline{d}_c))$ with $\sigma = \text{sign}(s.v_{z1} - s.v_{z2})$, *i.e.*, σ is the non-crossing RA sense of aircraft 1. It follows that the state s is safe in this type of an execution if $s.P \geq ALIM$. Thus, let $\textit{Close-in-Alt\_Safe} = \{s \in \mathcal{S} \mid s.P \geq ALIM\}$.

Theorem 9 *If $\alpha \in$ Close-in-Alt_Execs and $\alpha.s_0 \in$ Close-in-Alt_Safe, then $\alpha \in$ Safe_Execs.*

5. Conclusions

We demonstrate how high-level modeling techniques involving HIOA can be used to model and analyze complex safety-critical systems such as TCAS. In our presentation, we define HIOA models of all the core components of closed-loop system — components that involve both discrete and continuous behavior, define an idealized system involving two TCAS-equipped and "well-behaved" aircraft, and provide conditional yet absolute safety conditions for the closed loop system. We realize that many of the simplifying assumptions we make are restricting, but are intention is to illustrate how high-level modeling and analysis techniques which are amenable to formal analysis can be used to model such systems. We defer the analysis of more complicated TCAS behavior to future research.

References

[1] E. Dolginova and N. A. Lynch. Safety Verification for Automated Platoon Maneuvers: A Case Study. In O. Maler, editor, *Proc. International Workshop on Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *LNCS*, pages 154–170. Springer-Verlag, 1997.

[2] C. Heitmeyer and N. A. Lynch. The Generalized Railroad Crossing: A Case Study in Formal Verification of Real-Time Systems. In *Proc. 15th IEEE Real-Time Systems Symposium*, pages 120–131, Dec. 1994.

[3] N. Leveson. *SafeWare: System Safety and Computers*. Addison-Wesley, 1995.

[4] C. Livadas. Formal Verification of Safety-Critical Hybrid Systems. Master of Engineering Thesis, Dept. of Electrical Engineering and Computer Science, MIT, Sept. 1997.

[5] C. Livadas, J. Lygeros, and N. A. Lynch. High-Level Modeling and Analysis of TCAS. Technical Report, Laboratory for Computer Science, MIT, Dec. 1999.

[6] J. Lygeros, D. N. Godbole, and S. Sastry. A Verified Hybrid Controller for Automated Vehicles. In *35th IEEE Conference on Decision and Control (CDC'96)*, pages 2289–2294, Dec. 1996.

[7] J. Lygeros and N. A. Lynch. On the Formal Verification of the TCAS Conflict Resolution Algorithm. In *36th IEEE Conference on Decision and Control (CDC'97)*, pages 1829–1834, Dec. 1997.

[8] J. Lygeros, G. J. Pappas, and S. Sastry. An Approach to the Verification of the Center-TRACON Automation System. In T. A. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control (HSCC'98)*, volume 1386 of *LNCS*, pages 289–304. Springer-Verlag, 1998.

[9] N. A. Lynch, R. Segala, and F. Vaandrager. Hybrid Automata. Preprint/Work in Progress, July 1999.

[10] N. A. Lynch, R. Segala, F. Vaandrager, and H. B. Weinberg. Hybrid I/O Automata. In R. Alur, T. Henzinger, and E. Sontag, editors, *Proc. DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems, Hybrid Systems III: Verification and Control*, volume 1066 of *LNCS*, pages 496–510. Springer-Verlag, 1996.

[11] Radio Technical Commission for Aeronautics. Minimum operational performance standards for traffic alert and collision avoidance system (TCAS) airborn equipment. Technical Report RTCA/DO-185, RTCA, September 1990. Consolidated Edition.

[12] The MITRE Corporation. TCAS II collision avoidance subsystem requirements specification, September 1996.

[13] C. Tomlin, G. J. Pappas, and S. Sastry. Conflict Resolution for Air Traffic Management: A Study in Multi-Agent Hybrid Systems. *IEEE Transactions on Automatic Control*, 43(4), Apr. 1998.

[14] H. B. Weinberg, N. A. Lynch, and N. Delisle. Verification of Automated Vehicle Protection Systems. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III: Verification and Control*, volume 1066 of *LNCS*, pages 101–113. Springer-Verlag, 1996.