High-Level Modeling and Analysis of the Traffic Alert and Collision Avoidance System (TCAS)*

Carolos Livadas,[†] John Lygeros,[‡] and Nancy A. Lynch[§]

Abstract

In this paper, we demonstrate a high-level approach to modeling, analyzing, and verifying complex safety-critical systems through a case study on the Traffic Alert and Collision Avoidance System (TCAS) [1–3]; an avionics system that detects and resolves aircraft collision threats. Due to the complexity of the TCAS software and the hybrid nature of the closed-loop system, the traditional testing technique of exhaustive simulation does not constitute a viable verification approach. Moreover, the detailed specification of the system software employed to date as a means towards analysis and verification, neither help in intuitively understanding the behavior of the system, nor enable the analysis of the closed-loop system behavior. We advocate defining high-level hybrid system models that capture the behavior not only of the software, but also of the airplanes, sensors, pilots, *etc.* In particular, we show how the core components of TCAS can be captured by relatively simple Hybrid I/O Automata (HIOA) [4, 5], which are amenable to formal analysis. We then outline a methodology for establishing conditions under which TCAS guarantees sufficient separation in altitude for aircraft involved in collision threats. The contributions of this paper are the high-level models of the closed-loop TCAS system and the demonstration of the usefulness of high-level modeling, analysis, and verification techniques.

Keywords: Hybrid Systems, Automata, TCAS, Safety-Critical, Formal Verification.

---

*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.

[†]Carolos Livadas, Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square NE43-368, Cambridge, MA 02139, clivadas@theory.lcs.mit.edu

[‡]John Lygeros, Robotics Lab, 211-137 Cory Hall, Department of Electrical Engineering, University of California, Berkeley, CA 94720, lygeros@eecs.berkeley.edu

[§]Nancy A. Lynch, Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square NE43-365, Cambridge, MA 02139, lynch@theory.lcs.mit.edu

1 Introduction

The Traffic Alert and Collision Avoidance System (TCAS) [1–3] is an on-board aircraft conflict detection and resolution system used by all carrier aircraft in the US of more than 10 passengers. 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.

In practice, a safety-critical system such as TCAS is designed and implemented by undergoing extensive phases of testing through simulation. Unfortunately, this approach in verifying the correctness of a complex system's implementation 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 safety and performance guarantees of the system. Finally, and most importantly, the safety and performance guarantees obtained through extensive simulations are not absolute; in fact, it is not even possible to provide conditional performance guarantees, *e.g.*, under certain assumptions, the system is guaranteed to perform as required.

In an effort to alleviate the shortcomings of trying *a posteriori* to demonstrate a system's correctness through exhaustive testing, there has been considerable effort in applying formal specification and analysis techniques to evaluate the correctness of such systems. Due to its safety-critical nature and its great exposure, TCAS has been one of the few commercial systems for which formal modeling and analysis has been attempted. Based on informally stated high-level specifications salvaged from the early design stages and the actual software implementation of TCAS, a state-machine model of the behavior of the TCAS system has recently been compiled [6]. After applying software refinement techniques [7] to guarantee that the TCAS software is truly implementing its higher-level specifications, the specifications in turn are subsequently used as a basis for analyzing the behavior of the TCAS system. Unfortunately, it is our belief that such techniques, carried out either prior to, or after a system's design and implementation, begin specifying the system and modeling its behavior at relatively detailed levels. In so doing, the intuitive understanding of the behavior of the system is overshadowed by the details and technicalities present in the detailed low-level specifications. Moreover, such techniques focus their attention only on the software components 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.

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 components 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*. Composition enables the reasoning about a system's components in isolation and the subsequent use of the component-wise results when reasoning about the system as a whole. Model refinement enables the successive refinement of a system's model while guaranteeing that the lower-level models are indeed implementations of the higher-level models. In so doing, properties shown to be true of the behavior of more abstract models extend to more refined models without any proof obligations. 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 specified, 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 safety and performance guarantees. The methodology presented here has already been successfully applied to various safety-critical transportation systems, such as automated highways [8], personal rapid transit systems [9, 10], train gate controllers [11], and the Center-TRACON Automation System (CTAS) [12]. 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 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 (HIOA) [4,5]. The verification techniques we use involve a combination of techniques from control theory and distributed algorithms.

We proceed by briefly surveying the prior work in the area of applying formal analysis techniques to air traffic management systems and then present the hybrid I/O automaton model. In Section 4.1, 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 5, 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 6, we conclude and suggest future research directions. Earlier versions of this work has been presented in Refs. 13 and 14.

2 Prior Work

The safety-critical nature of air traffic management systems and the inadequacy of current system and software verification techniques have guided system designers toward formal approaches in specifying, modeling, and analyzing system behavior. To our knowledge, formal modeling and analysis techniques have been used to model and verify two systems in the area of air traffic control and management: the Traffic Alert and Collision Avoidance System (TCAS) — an on-board aircraft conflict detection and resolution system used by all carrier aircraft in the US of more than 10 passengers — and the Center-TRACON Automation System (CTAS) — a system used at several airports to assist air traffic controllers in scheduling the arrival and departure of aircraft in the vicinity of the airports.

In an attempt to design a formal specification language for complex software systems and while focusing on TCAS as their primary case study, Leveson *et al.* have developed the *Requirements State Machine Language* (RSML) [6], a formal modeling language designed for requirements specification of process control systems. This language was designed based on prior formalisms for modeling complex systems, such as *StateCharts* [15] *etc.*, and focuses primarily on providing simple system specifications that capture the essence of the systems at hand and are readable, understandable, and primarily usable by the designers and users of the system alike. The RSML specifications of TCAS produced by Leveson *et al.* [6] have actually become the official TCAS specifications. Recently, Leveson *et al.* [16] have been developing a process of producing system specifications that focus on capturing the high-level system goals and requirements as well as the design decisions and assumptions made during a system's design phase. The resulting specifications, which are coined *Intent Specifications* to indicate that they capture the intents of the system designers, are geared towards capturing the intuitive understanding of the system being specified while noting the assumptions and rationale behind design decisions. Moreover, their multi-level structure provides an easy description of the system through which high-level reasoning about the behavior of the system is possible. Leveson *et al.* argue that intent specifications are the preferred specification format that can be used to study, maintain, and revise the design of a complex system. In fact, they are designed so as to be easily readable and understandable by all the people involved in the design and use of such systems — in the case of TCAS, this would involve the designers of TCAS, air traffic controllers, aircraft pilots, as well as newcomers who may be asked to update and augment the system.

There have been several efforts in studying the CTAS system. From a formal modeling and analysis perspective, Lygeros *et al.* [12] produced a high-level HIOA-based model of the CTAS system. The correctness analysis of the system was however deferred to future research. In the area of software analysis, Jackson, Chapin *et al.* [17], studied the structure of the software involved in the CTAS system and discovered that, due to fear of software redesign and to the continuous addition of new functionality, the CTAS's software has become complicated and inefficient. After redesigning a core component of CTAS's software, they concluded that basic software engineering techniques and new technology can make the system much simpler, easier to reason about, maintain, and augment.

3 Hybrid System Modeling Formalism

The modeling formalism used in this paper is the *hybrid I/O automaton* (HIOA) model [4,5]. This model is an extension of the *timed I/O automaton* (TIOA) model [18,19] and allows the explicit treatment of continuous behavior. In this section, we introduce the HIOA model and describe how this model can be used to model, analyze, and verify hybrid systems. For a detailed presentation of the HIOA model, the reader is referred to Refs. 4 and 5.

In the HIOA modeling formalism, a system is modeled as a collection of hybrid I/O automata each of which specifies the discrete and continuous behavior of a particular component of the system. Each of these hybrid I/O automata is a (possibly) infinite state machine whose states are the valuations of a set of variables. The discrete behavior of a system is modeled by "jumps" in the state of its HIOA model, whereby the valuation of the system's state variables changes. These discrete jumps are specified as labeled transitions whose labels are the *actions* that carry out the transition from the initial to the final state of the jump. All possible jumps in a HIOA's state as a result of the execution of an action comprise the set of *discrete transitions* of the HIOA. The continuous behavior of a system is modeled by the continuous evolution of the valuation of the variables of the system model as time advances; that is, the continuous behavior of an HIOA model is specified by the set of *trajectories* of the state variables of the automaton. The external interface of an HIOA with its environment is dictated by the partition of its variables and its actions into three categories: *input*, *internal*, and *output*. Input variables and actions model the variables and actions of the environment that are exposed to the HIOA and over which the HIOA has no control. Internal variables and actions specify the internal functionality of the HIOA and are not exposed to the environment. In addition to modeling the behavior of the system, internal variables and actions are often used for bookkeeping purposes. Output variables and actions comprise the state and the discrete behavior of the system model that is exposed to the environment and over which the HIOA has control. Thus, the HIOA comprising the system model "communicate" through their shared input and output variables and actions.

In the following sections, we present a more detailed definition of hybrid I/O automata and give a brief comparison of HIOA and other hybrid system modeling formalisms.

3.1 Hybrid I/O Automata

A hybrid I/O 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 initial states, the discrete transitions, and the trajectories of a HIOA must satisfy several technical conditions which are omitted here.

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 initial states of A denote the states from which the evolution of the system is restricted to begin. When analyzing a HIOA A, it is often useful to define *derived* variables for A. Such variables are functionally dependent on the variables of the automaton A and, although useful in the analysis of A, are not essential in its definition.

Input variables and actions model the variables and actions of the environment that are exposed to the HIOA and over which the HIOA has no control. Thus, the input actions of A are always enabled; that is, an action a in Σ^{in} can be scheduled at any point in time. In order to model the scheduling of an action outside the HIOA A that is unobservable by A except possibly through discrete changes in the valuation of its input variables, the set of input actions Σ^{in} always contains the *environment* action, e. When this action is scheduled the valuation of the input variables of the automaton A may arbitrarily change, in effect modeling the environment's control over the input variables. In the special case when the valuation of the input variables does not change upon the scheduling of an environment action, the environment action is referred to as *stuttering*.

The set of discrete transitions $\mathcal{D}$ is comprised of all triples (s, a, s') (or equivalently $s \xrightarrow{a} s'$), each of which denote the discrete transition of the HIOA A from a state s to a state s' through the scheduling of the action $a \in \Sigma$. By convention, the set of discrete transitions of A is specified by collectively describing all discrete transitions involving each action a in Σ in *precondition-effect* format. This format is comprised of a *label*, a *precondition*, and an *effect clause*. The *label* corresponds to the label of the action a. The *precondition* is a predicate over the variables of A and specifies the conditions under which the action a is enabled; that is, the precondition defines the set of states in which the action a may be scheduled. However, the scheduling of an action a in Σ is not required whenever it is enabled. The valuation of the precondition of an action a in a state s is denoted by $s.\mathit{Pre}(a)$. The *effect clause* specifies the pseudo-code that must be applied to the pre-state of a discrete transition involving the action a so as to yield the post-state of the discrete transition. It follows that, in order for (s, a, s') to be a discrete transition of A, the precondition in the specification of the action a must be satisfied by the pre-state s, *i.e.*, $s.\mathit{Pre}(a)$ must evaluate to `True`. Moreover, the application of the pseudo-code in the effect clause of the specification of the action a to the pre-state s must yield the post-state s'. Note that since input actions are always enabled, they have no preconditions, *i.e.*, for any a in Σ^{in}, it is the case that $Pre(a) \equiv \texttt{True}$.

The set of trajectories $\mathcal{W}$ is a set of functions from intervals of time to valuations of the variables of A. For a trajectory w of A, $w \in \mathcal{W}$, and an instant in time t in the domain of w, $t \in dom(w)$, $w(t)$ represents the valuation of the variables of the HIOA t time units into the trajectory w. 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 $w(w.ltime)$. The set of trajectories $\mathcal{W}$ of A is specified by pseudo-code which describes the properties that any trajectory w involving the variables of A must satisfy in order to be a trajectory of A. By convention, the trajectory pseudo-code consists of a collection of predicates all of which must be satisfied throughout any trajectory w of A.

The evolution of the system as time advances is described by *hybrid executions* — finite or infinite alternating sequences of trajectories and actions. More formally, 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 $w_i.lstate \xrightarrow{a_{i+1}}_A w_{i+1}.fstate$). The set of all execution fragments of A is denoted by $execs(A)$. If the first state of α is an initial state, *i.e.*, $w_0.fstate \in \Theta$, 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. A hybrid execution α of A is *admissible* if $\alpha.ltime = \infty$.

The *hybrid trace* of a hybrid execution fragment is defined as the evolution of the input and output variables of the HIOA; that is, the externally visible part of the hybrid execution. More formally, the *hybrid trace* of a hybrid execution fragment α of A, denoted by $h\text{-}trace(\alpha)$, 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 $h\text{-}traces(A)$, is the set of hybrid traces that arise from all the finite and admissible hybrid executions of A.

Since the valuation of the variables of an automaton A can change instantaneously, the variables of A can have multiple valuations at the same instant in time. However, it is possible to specify particular occurrences of states by designating the trajectory within which the state occurs and the time elapsed from the beginning of the trajectory to the point in time of the particular occurrence of the state within the given trajectory. Thus, we introduce the notion of *superdense time* in an execution fragment α of A as a pair (i,t), where $t \leq w_i.ltime$. We totally order superdense times in α lexicographically; that is, superdense times, *e.g.*, (i,t) and (i',t'), are ordered first by the index of the trajectory, *e.g.*, i and i', and then by their trajectory times, *e.g.*, t and t'. Finally, an *occurrence of* a state s *in* α is specified by 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 said to be *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 $\Sigma_1^{out} = \Sigma_2^{out}$. In the HIOA model, the notion of an implementation relation is given by inclusion of the sets of hybrid traces; that is, provided that the A_1 and A_2 are comparable, A_1 implements A_2 if every external behavior of A_1 is allowed by A_2, *i.e.*, $h\text{-}traces(A_1) \subseteq h\text{-}traces(A_2)$. In this setting, A_1 and A_2 are referred to as the *implementation* and the *specification*, respectively. We often use the notation $A_1 \leq A_2$ to denote that the hybrid traces of A_1 are included in those of A_2; that is, $A_1 \leq A_2 \triangleq h\text{-}traces(A_1) \subseteq h\text{-}traces(A_2)$ and, thus, $A_1 \leq A_2$ implies that A_1 *implements* A_2.

The composition of two HIOA is defined as their synchronization on shared input/output variables and input/output actions. Two HIOA A_1 and A_2 can only be composed if they are *compatible*; that is, 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. It is important to note that the composition of two HIOA results in a HIOA. Moreover, composition respects the implementation relation, *i.e.*, supposing B is a HIOA, if the HIOA A_1 implements the HIOA A_2, then the composition of A_1 with B implements the composition of A_2 with B.

Most of the proofs in the HIOA framework use *invariant assertions* and *simulations*. In the case of invariant assertions, the proofs are by induction on the length of a hybrid execution of the HIOA. Such proofs show that a particular predicate on the state of the HIOA is satisfied in every state of the execution. A simulation is a mapping between the states of two HIOA and is used to prove that one HIOA implements another. The fact that the mapping is indeed a simulation is again done by induction on the length of a hybrid execution of the implementation. This induction matches up individual steps in the implementation with either single steps, or sequences of steps, in the specification; in effect, it is shown that each step of the implementation is allowed by the specification. Simulations are very useful because they promote reasoning about systems at a high level of abstraction. Suppose that we have shown that an abstract HIOA model of a system exhibits a particular property and that we want to show that the same property holds for a more refined HIOA model of the system. One approach is to start from scratch and try to generate a new proof of the property for the refined model. An alternative approach is to show, through a simulation, that the refined model of the system actually implements the abstract model. In so doing, all the properties that have been shown to hold for the abstract model extend to the refined model without any proof obligations. The latter approach can greatly reduce the amount of work needed to extend results from more abstract to more refined models. This technique often results in multiple models of the system spanning several *levels of abstraction*.

To date, the proofs in all our case studies of hybrid systems have been conducted by hand. However, it is our belief that software support can be employed to assist in the various proof obligations. Moreover, we believe that given a sufficiently detailed HIOA model of a system, the software for the discrete parts of the actual system could be generated automatically such that, by construction, the software is a truthful implementation of its more abstract specifications.

3.2 Related Modeling Formalisms

The recent interest in the formal verification of real-time and hybrid systems has resulted in a number of techniques to model and analyze their behavior. In particular, models that are analogous to the timed I/O automaton model [18,19] are the models of Alur and Dill [20], Lamport [21], and Henzinger, Manna, and Pnueli [22]. As is the case with the timed I/O automaton model, these models have also been extended to the hybrid setting; for instance, the timed transition model [22] has been extended to the phase transition model [23,24]. Hybrid I/O automata are also analogous to phase transition models [23–26], from which they were actually inspired — the discrete transitions and the trajectories of hybrid I/O automata correspond to the transitions and the activities of phase transition systems. The hybrid system model [25,26] is similar to the phase transition model with the distinction that, as in the hybrid I/O automaton model, discrete transitions are labeled, thus allowing the appropriate synchronization of composed automata. The distinction between the hybrid system model [25, 26] and the hybrid I/O automaton model lies in the latter's classification of the variables and discrete transitions into input, internal, and output.

4 The TCAS System

A common paradigm in managing and controlling safety-critical systems has been to separate the system control into operation and protection. The operation aspect of the control is geared towards the normal operation of the system at hand and is optimized for functionality and performance. For large and complex systems, this translates into large and complex control systems whose reliability is questionable. The protection aspect of the control is responsible for the control of the system in case of emergency; that is, it is responsible for preventing hazards. By keeping the complexity of the protection subsystem low, system designers hope to increase the reliability of the closed-loop system. In the case of air traffic management and control, the operation of the system is handled by the air traffic controllers who instruct the aircraft pilots to follow specific flight paths. However, due to the complexity of the system it was deemed appropriate to design a protection system that alerts pilots to nearby aircraft posing mid-air collision threats and proposes threat resolution maneuvers. This protection system is denoted the Traffic Alert and Collision Avoidance System (TCAS).

TCAS has evolved through a series of versions. The first version, TCAS I [1] (originally named the Beacon-Based Collision Avoidance System (BCAS)), issues a proximity warning or *Traffic Advisory* (TA) to alert a pilot to nearby aircraft that pose a mid-air collision threat. Airline regulations dictate that pilots should not undertake maneuvers in response to TAs, unless the aircraft posing the mid-air collision threat has been visually acquired. TCAS I has already been deployed and is mandatory for all carrier aircraft of 10 to 30 passengers operating within the US airspace.

The second version of TCAS, TCAS II [1–3], extends the functionality of its predecessor by proposing maneuvers that resolve the mid-air collision threats posed by nearby aircraft. In particular, TCAS II 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, however, the danger of collision increases TCAS II issues a *Resolution Advisory* (RA); that is, it proposes to the pilot a flight maneuver that is supposed to resolve the mid-air collision threat. Airline regulations dictate that pilots should abide by the RAs proposed by the TCAS system, unless after visually acquiring the aircraft posing the mid-air collision threat a better resolution strategy is available. The RAs issued by TCAS II are restricted to the vertical plane. In particular, resolution maneuvers instruct the pilots to refrain from climbing or descending faster than a given ascent or descent rate, respectively. When all aircraft involved in the mid-air collision threat are TCAS equipped, TCAS II uses a symmetry-breaking communication protocol so as to determine consistent resolution maneuvers for each aircraft. For example, in the case of two aircraft, one of the two aircraft should be instructed not to climb faster than a given ascent rate and the other not to descend faster than a given descent rate. Following the determination of a consistent set of maneuvers, the maneuvers are continuously presented to the pilots until the mid-air collision threat has been resolved; that is, until TCAS II deems that the aircraft no longer pose a mid-air collision threat to each other.

Following extensive field testing and feedback from pilots and air traffic controllers, TCAS II evolved through a series of minor revisions from its initial version to TCAS II-6.04A [2]; the version of TCAS that is currently standard for all carrier aircraft of more than 30 passengers operating within the US airspace. Many of the minor revisions were conducted due to oversensitivity of the system, which would frustrate air-traffic controllers and make pilots reluctant to using the TCAS system.

The functionality of TCAS II-6.04A was further extended to allow RA *reversals*. This version of TCAS, TCAS II-7 [3, 27], continuously monitors and reevaluates the effectiveness of its previously issued RA. If at any point in time TCAS II determines that, while the RA currently being issued fails to resolve the conflict, the reversed RA resolves the conflict, it is capable of reversing the sense of the previously issued RA. In order to avoid situations in which TCAS induces successive reversals, TCAS II-7 is only allowed to reverse once. After recently being tested through extensive simulation [28], TCAS II-7 has been adopted by the International Civil Aviation Organization (ICAO) as the international standard and is mandatory for all new carrier aircraft.

Future versions of TCAS are already at the conceptual and development stage. TCAS III was intended to augment the functionality of TCAS II-7 by allowing RAs both in the horizontal and vertical planes. However, the complexity of augmenting the already complex TCAS II-7 system and the then inaccurate sensing hardware resulted in the abandonment of TCAS III early in its conceptual stage. The recent advancement in sensor technology and the potential use of the Global Positioning System (GPS) for accurate position measurement, has revived the plans of horizontal

RAs in the latest version of TCAS, TCAS IV.

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. Studies indicate that pilots get uncomfortable when presented with advisories of very short duration or with a sequence of conflicting advisories. The passenger comfort requirement imposes a limit on the accelerations that can be employed in a maneuver. Finally, air traffic controllers dislike maneuvers that result in large and abrupt changes of altitude, as they conflict with their standard way of arranging air traffic. Indeed, in all version of the TCAS system, a substantial fraction of the TCAS functionality is devoted to such objectives.

4.1 Modeling TCAS Using HIOA

In this paper, we focus our attention on TCAS II-7; the version of TCAS II that is capable of RA reversals. Our model of the TCAS system is based on the low-level specifications of TCAS II-7 [3,27]. Following the functional breakdown in TCAS's specifications, our model of the closed-loop TCAS system is comprised of the components shown in Figure 1. While generating the component models, the input and output variables and actions of each of the component HIOA are chosen so as to correspond to the actual interfaces of the closed-loop system components. Since we want to evaluate the resolution aspects of TCAS, we model all aspects of the TCAS system relating to RAs and ignore those relating to TAs. Moreover, in order to keep the analysis tractable, we remove the functionality of TCAS that is devoted to human factors concerns. Finally, even though TCAS II-7 is capable of issuing TAs and RAs for conflicts involving multiple aircraft, our model is restricted to conflicts involving only two aircraft. In the following subsections, we describe the abstract models of each of the components of the closed-loop system shown in Figure 1. Once again, we stress that the models presented in this section are abstract models of the system components and are intended for reasoning about the behavior of the closed-loop system at a high level of abstraction. As greater familiarity is gained with the system and more specific questions need to be answered, the component models may be refined so as to model the behavior of the system in more detail. Throughout the following subsections, we propose various such refinements for the component models of the closed-loop TCAS system. The formal models of all components are presented in Appendixes A–F. Earlier versions of these models appeared in Refs. 13 and 14.

Modeling the closed-loop TCAS system using HIOA has several advantages. First, in contrast with prior modeling approaches, which focus on modeling and verifying only the software components of the closed-loop system, HIOA can be used to model all aspects of the closed-loop system; that is, in addition to the various components comprising the TCAS system, we model the behavior of the pilots, the dynamics of the aircraft, and the communication channels. Second, by modeling each of the functional components of the system as a separate HIOA, we obtain a model that captures the inherent modularity in the system and promotes the understanding of both the system behavior and the component interaction. Moreover, by taking advantage of the formal notion of composition in the HIOA model, results obtained by analyzing the behavior of the component models in isolation, extend directly to the closed-loop system. Finally, using the HIOA model we can take full advantage of model refinement; that is, properties shown to be true of more abstract models can be extended to more refined models by showing that the more refined models are indeed implementations of their more abstract counterparts. This can greatly reduce the amount of work needed to extend safety results from more abstract to more refined models. Moreover, since each of the components of the closed-loop system is modeled by a HIOA, we can selectively refine only the models of the components whose detailed behavior is essential in the safety and performance analysis. Essentially, by having a modular HIOA based model of the closed-loop system, we can focus our attention on the components whose behavior is more important.

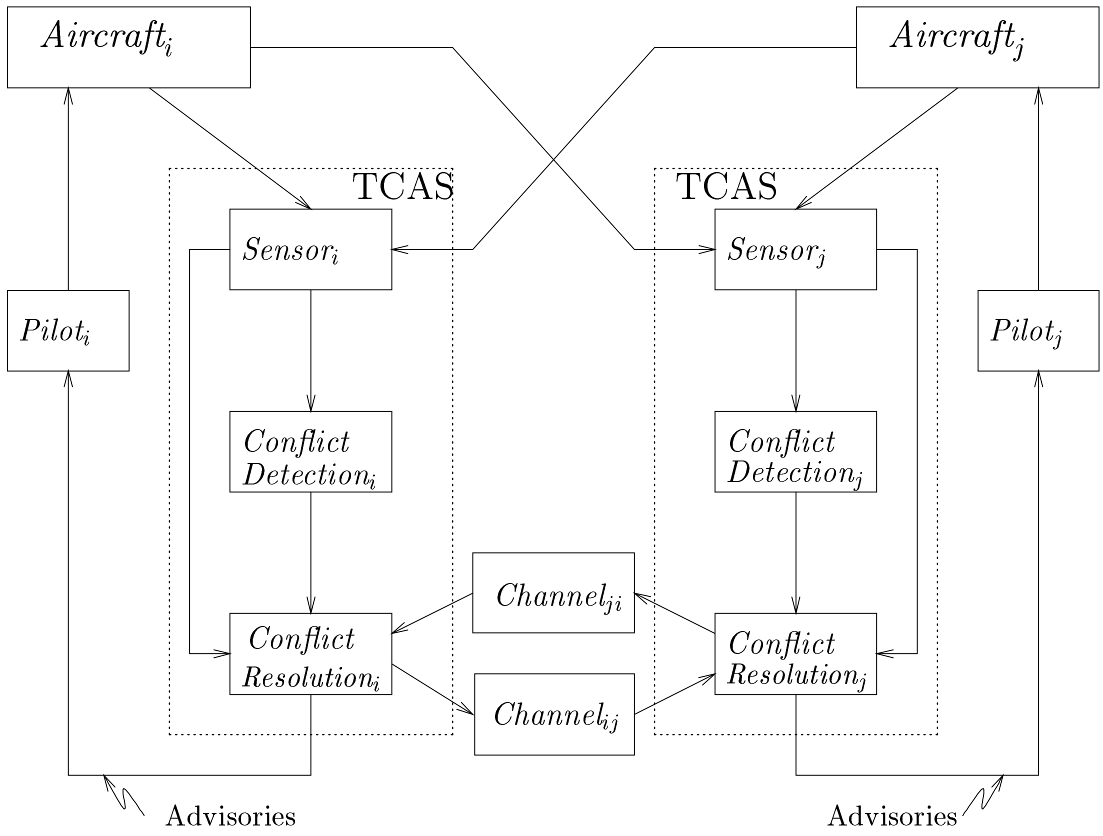

Figure 1: TCAS System Block Diagram

The following sections make use of the nomenclature, constants, data types, and parameters summarized in Table 1.

Table 1: Nomenclature, Constants, Data Types, and Parameters.

Nomenclature:

$\perp$ — the undefined element or variable valuation

Constants:

$\texttt{RMAX} = 12\ nmi$

$\texttt{RDTHR} = 10\ ft/s$

$\texttt{NomStr} = 1500\ ft/min$

$\texttt{IncStr} = 2500\ ft/min$

$\texttt{2NomStr} = 3000\ ft/min$

Data Types:

$\text{Dir} = \{\text{Climb}, \text{Descend}\}$

$\text{Dir}_\perp = \text{Dir} \cup \{\perp\}$

$\text{Strengths} = \{-2000, -1000, -500, 0, 1500, 2500\}(ft/min)$

$\text{Aircraft} = \{1, 2\}$

$\text{Others}_i = \text{Aircraft} \setminus \{i\}$, for all $i \in \text{Aircraft}$

Parameters:

`ALIM` — Minimum Allowable Vertical Separation (ft)

`DMOD` — Threat Minimum Range Threshold (nmi)

`H1` — Threat Minimum Divergence Threshold (nmi^2/s)

`TRTHR` — Threat Modified Tau Threshold (s)

`ZT` — Threat Altitude Threshold (ft)

4.1.1 Aircraft Model

The model of the aircraft is intended to capture the flight of the aircraft and the interface of the aircraft with the sensor automata used to measure each aircraft's state. In so doing, we must adopt flight dynamics for the aircraft, describe the avionics equipment the aircraft, and specify the interface of the aircraft automaton with the sensor automata. In view of simplicity, we adopt very simple aircraft dynamics and assume that the aircraft output their exact state to the sensor automata. We thus defer the introduction of sensor uncertainty to the sensor automata.

Each aircraft, $i \in \{1,2\}$, is modeled by the HIOA $A_i = (U_{A_i},\ X_{A_i},\ Y_{A_i},\ \Sigma^{in}_{A_i},\ \Sigma^{int}_{A_i},\ \Sigma^{out}_{A_i},\ \Theta_{A_i},\ \mathcal{D}_{A_i},\ \mathcal{W}_{A_i})$ (Appendix A). At this stage we assume there are no output or internal actions and no input action (other than the environment action), that is $\Sigma^{in}_{A_i} = \{e\}$, $\Sigma^{int}_{A_i} = \Sigma^{out}_{A_i} = \emptyset$. At a later stage appropriate actions can be added to model discrete changes in the physical system, such as malfunctions.

The avionics hardware with which each aircraft is equipped, is specified by the output variable $Equipment_i \in \{\text{None, Report, TCAS}\}$; an aircraft could have either no avionics hardware, or an altitude reporting transponder, or, finally, an altitude reporting transponder and TCAS. When an aircraft is equipped with an altitude reporting transponder, the aircraft can be identified by its unique transponder identifier, which is specified by the output variable $Mode\_S_i \in \mathbb{N}$. All the aircraft considered in our analysis are assumed to be equipped with an altitude reporting transponder; that is, a $Mode\_S$ variable is defined for all aircraft, and $Mode\_S_i \neq Mode\_S_j$, for all $i, j \in \{1,2\}$, $j \neq i$. The unique $Mode\_S$ identifier is used by TCAS for the purposes of symmetry braking; the smaller an aircraft's $Mode\_S$ identifier the higher the aircraft's priority. At this stage the variables $Mode\_S_i$ and $Equipment_i$ are assumed to have trivial dynamics: they simply maintain their initial value. At a later stage more complicated dynamics may be added to model malfunctions that may change the status of the aircraft.

The physical movement of the airplane i is summarized by the trajectories of its position and velocity. Let $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $v_i = (v_{xi}, v_{yi}, v_{zi}) \in \mathbb{R}^3$ and $a_i = (a_{xi}, a_{yi}, a_{zi}) \in \mathbb{R}^3$ be the position, velocity, and acceleration of the aircraft i with respect to some fixed reference frame on the ground. 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 $X_{A_i} = \emptyset$. The aircraft dynamics we use are very simple and ignore important aircraft characteristics such as the details of the aerodynamic forces, high frequency modes (due for example to flexion of the wings), the effect of structural controls (such as the flaps), and input constraints (such as those imposed by the aircraft engine). As the maneuvers required by TCAS are unlikely to excite high order dynamics, induce input saturation, *etc.*, these simple aircraft dynamics are deemed sufficient for our analysis.

4.1.2 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 TCAS, *e.g.*, the aircraft's radio and pressure altimeters (which measure the aircraft's altitude), the aircraft's radar (which measures the range of neighboring aircraft), and the *Mode_S* transponders of other aircraft (which advertise the altitude of their respective aircraft). The information that the sensors receive is quantized spatially and sampled temporally. In fact, the data provided from the measuring devices (altimeters, radar, and transponders) can sometimes be quantized roughly; for example, although the transponders of nearby aircraft quantize the altitudes they report at either 25*ft* or 100*ft* increments, the altitude and range measured by each aircraft is typically quantized finer. In order to "smooth" the received data and produce estimates of the range and altitude rates of all aircraft, the data provided by the measuring devices is filtered. These filters are effectively simplified versions of the Kalman filter and are known as the "α-β-γ tracker" for the range data, and the "α-β tracker" for the altitude data. If the quantization of the altitude data is coarse (100*ft*) a nonlinear filter known as the "level-occupancy-time algorithm" is used instead. In our model model of the aircraft sensors, we ignore the operation of these filters and the rules for switching between them.

The sensors of aircraft i are modeled by the HIOA $S_i = (U_{S_i},\ X_{S_i},\ Y_{S_i},\ \Sigma^{in}_{S_i},\ \Sigma^{int}_{S_i},\ \Sigma^{out}_{S_i},\ \Theta_{S_i},\ \mathcal{D}_{S_i},\ \mathcal{W}_{S_i})$ (Appendix B). The input variables of S_i are the positions and velocities of all aircraft, $U_{S_i} = \{p_j, v_j\}_{j\in\{1,2\}}$. 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 the neighboring aircraft, j. In other words, $Y_{S_i} = \{h_{ij}, \dot{h}_{ij}\}_{j\in\{1,2\}} \cup \{R_{ij}, \dot{R}_{ij}\}_{j\neq i}$. At this stage the sensor automaton is assumed to have no input or internal actions ($\Sigma^{in}_{S_i} = \Sigma^{int}_{S_i} = \emptyset$).

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_{Ai}, n_{ARi}, n_{Ri}, and n_{RRi} denote the width of the error intervals for h_{ij}, $\dot{h}_{ij}$, R_{ij}, and $\dot{R}_{ij}$ respectively, and $[a \pm b]$, for $b \geq 0$, denote the interval $[a-b, a+b]$. The output variables of the sensors are updated every T_{S_i} seconds, upon the occurrence of an output action $Sample_i$. We set $\Sigma_{S_i}^{out} = \{Sample_i\}$. An internal variable $t_{S_i} \in \mathbb{R}$ keeps track of the time that has elapsed since the last sample. Upon occurrence of $Sample_i$ the value of h_{ij} is reset to a new value in the interval $[z_j \pm n_{Ai}]$ (and similarly for the remaining output variables of the sensor).

We assume n_{Ai}, n_{ARi}, n_{Ri}, and n_{RRi} are constant; in a more refined model of the sensors, these quantities may constitute internal variables so as to model, for example, the effect of switching between filtering algorithms. In the current TCAS implementation data can be sampled at either a high ($T_{S_i} = 1$ second) or a low ($T_{S_i} = 5$ seconds) rate. In our analysis, we assume that $T_{S_i} = 1$ throughout. If changes in the sampling rate need to be modeled at a later stage, T_{S_i} can be considered as an internal variable whose value changes whenever the sampling rate does.

If in future analyses of the TCAS system a more refined model of the aircraft sensors is required, one may define and subsequently prove that the composition of more detailed automata representing the various sensory components implement the abstract sensor automaton S_i presented in this section.

4.1.3 Conflict Detection

The conflict detection automaton models the component of TCAS that is responsible for detecting whether neighboring aircraft pose a mid-air collision threat and whether previously declared threats have been resolved. The conflict detection automaton monitors the state of the aircraft provided by the sensor automaton and schedules actions that are used to declare and undeclare threats.

The conflict detection module is modeled by the HIOA $D_i = (U_{D_i}, X_{D_i}, Y_{D_i}, \Sigma_{D_i}^{in}, \Sigma_{D_i}^{int}, \Sigma_{D_i}^{out}, \Theta_{D_i}, \mathcal{D}_{D_i}, \mathcal{W}_{D_i})$ (Appendix C). 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, $U_{D_i} = Y_{S_i} \cup \{Threat_{ij}\}_{j \neq i}$. At this stage 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 of aircraft j with respect to aircraft i need to satisfy for aircraft j to be declared a threat by aircraft i.

$$\begin{aligned} Range\_Test_{ij} = & ((\dot{R}_{ij} > \mathtt{RDTHR}) \wedge \hat{R}_1) \\ & \vee ((\dot{R}_{ij} \leq \mathtt{RDTHR}) \wedge \hat{R}_2) \end{aligned}$$

with

$$\begin{aligned} \hat{R}_1 &= (R_{ij} \leq \mathtt{DMOD}) \wedge (R_{ij}\dot{R}_{ij} \leq \mathtt{H1}) \text{ , and} \\ \hat{R}_2 &= (R_{ij} \leq \mathtt{RMAX}) \wedge \left(\frac{R_{ij} - \frac{\mathtt{DMOD}^2}{R_{ij}}}{\min\{\dot{R}_{ij}, -\mathtt{RDTHR}\}} < \mathtt{TRTHR} \right) \end{aligned}$$

`RMAX` is a system constant equal to $12nmi$. `RDTHR` is a system constant equal to $10ft/s$. `DMOD` (or Threat Minimum Range Threshold, units of nmi), `H1` (or Threat Minimum Divergence Threshold, units of nmi^2/s), and `TRTHR` (or Threat Modified Tau Threshold, units of s) are system parameters whose values are determined by the pilot-selected sensitivity and the ground-station-commanded sensitivity of TCAS.

The $Altitude\_Test_{ij}$ estimates the vertical separation between the aircraft i and j at the "time of closest approach"; that is, the vertical separation of the aircraft i and j at the time when their spatial separation is minimum. TCAS's estimate of the time to closest approach, τ_{ij}, is given by:

$$\tau_{ij} = -\frac{R_{ij}}{\min\{\dot{R}_{ij}, -10\}}$$

Thus, the altitude test $Altitude\_Test_{ij}$ is satisfied if the projected vertical separation of the aircraft i and j is below the system parameter ZT (or Threat Altitude Threshold, units of ft). The parameter ZT depends on the aircraft altitude.

$$Altitude\_Test_{ij} = \left(\left| (h_{ii} - h_{ij}) - (\dot{h}_{ii} - \dot{h}_{ij})\tau_{ij} \right| \leq \mathtt{ZT} \right)$$

We assume that an intruding aircraft is declared a threat by TCAS as soon as it satisfies (or "passes") both the range and the altitude tests. In practice, a number of exceptions to this rule are introduced in the TCAS specifications, in order to reduce the number of false alarms and to improve the speed of detection in cases where additional information is available through communicated intents in conflicts involving TCAS equipped aircraft. In our analysis, we ignore all these exceptions. 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 this point in time, the $Undeclare_{ij}$ output action is scheduled.

4.1.4 Conflict Resolution

The conflict resolution automaton models the component of TCAS that is responsible for issuing resolution advisories. The conflict resolution automaton gets initially alerted to a threat through either the conflict detection automaton, or the intent messages sent by the neighboring aircraft. Following the threat detection and periodically thereafter, the conflict resolution automaton evaluates its resolution advisory options and issues updated resolution sense and strength advisories. Whenever there is a change in the sense of a RA, the conflict resolution automaton sends a message to the other aircraft involved in the threat communicating its RA intentions. In order to avoid successive RA reversals, the conflict resolution automaton can only change the sense of a RA once.

The conflict resolution module is modeled by the HIOA $R_i = (U_{R_i}, X_{R_i}, Y_{R_i}, \Sigma^{in}_{R_i}, \Sigma^{int}_{R_i}, \Sigma^{out}_{R_i}, \Theta_{R_i}, \mathcal{D}_{R_i}, \mathcal{W}_{R_i})$ (Appendix D). 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.*, $U_{R_i} = Y_{S_i} \cup \{Mode\_S_j, Equipment_j\}_{j \in \{1,2\}}$. The output variables of R_i are the boolean $Threat_{ij}$ variables, indicating that aircraft i considers aircraft j as a threat, and a resolution advisory for the pilot, consisting of a $Sense_i \in \{\mathrm{Climb}, \mathrm{Descend}, \perp\}$ and a $Strength_i \in$ Strengths (units of *ft/min*); negative strengths are referred to as *Vertical Speed Limits* (VSL), 1500 is referred to as the *nominal strength* (`NomStr`), and 2500 as the *increased strength* (`IncStr`). The $Sense_i$ indicates whether aircraft i should try to pass above ($Sense_i$ = Climb) or below ($Sense_i$ = Descend) the intruding aircraft. The $Strength_i$ determines whether this goal is to be achieved by asking the pilot to avoid "bad" maneuvers (take no action that will force the vertical speed to exceed a particular VSL) or by asking the pilot to actively pursue "good" maneuvers (increase the vertical speed to 1500 or 2500 *ft/min*). $Sense_i = \perp$ indicates that no action is needed. In summary, the output variables of R_i are $Y_{R_i} = \{Sense_i, Strength_i\} \cup \{Threat_{ij}\}_{j\in\{1,2\}}$. 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[1]. 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 and input action $Receive_{ij}(\text{dir})$, with $\text{dir} \in \{\text{Climb}, \text{Descend}\}$ 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 $\Sigma^{in}_{R_i} = \{Sample_i\} \cup \{Declare_{ij}, Receive_{ij}(\text{dir}), Undeclare_{ij}\}_{j\neq i}$. Aircraft i sends its intentions to aircraft j through an output action, $Send_{ij}(\text{dir})$, where $\text{dir} \in \{\text{Climb}, \text{Descend}\}$ 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.*, $a_j \equiv [0\ \ 0\ \ 0]^T$. It also assumes that the pilot of aircraft i will respond to the advisory after a delay of exactly d (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.

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

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 formally 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 $\sigma_i \in \{-1, 1\}$ and $Strength \in$ Strengths, consider the derived variable:

$$SEP_{ij}(\sigma_i, Strength) = \begin{cases} \sigma_i \left[(h_{ii} - h_{ij}) + (\dot{h}_{ii} - \dot{h}_{ij})\tau_{ij} \right] \\ \quad \text{if } (\tau_{ij} \leq d) \vee (\sigma_i \dot{h}_{ii} \geq Strength) \\ \sigma_i \left[(h_{ii} - h_{ij}) + (\dot{h}_{ii} - \dot{h}_{ij})d \right. \\ \quad \left. + (\sigma_i Strength - \dot{h}_{ij})(\tau_{ij} - d) \right] \\ \quad \text{otherwise} \end{cases}$$

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 `NomStr` = 1500 *ft/min* and 2`NomStr` = 3000 *ft/min*.

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 descent resolution provides adequate separation, the one that produces the largest separation is chosen[2]. If one of the two resolutions 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; that is, the advisory that would result in the aircraft not crossing each other in altitude. For example, if aircraft i is already higher that aircraft j, then aircraft i should prefer a Climb resolution advisory. If aircraft j has already issued an advisory, the complementary

[2]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.

sense (encoded by the received intent) is typically chosen. The only exception is if aircraft i has a lower *Mode_S* number, the received intent corresponds to a crossing maneuver and aircraft i deems that a non-crossing maneuver resolves the mid-air collision conflict.

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 lower priority (*i.e.*, 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 altitude separation, while the reversed advisory will. However, aircraft i can only reverse once; the internal variable $\mathit{Reversed}_i$ is used to enforce this requirement. The new intent is communicated to aircraft j which, due to its lower priority (*i.e.*, 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; that is, upon the scheduling of each Sample_i action. The choice of $\mathit{Strength}_i$ again depends on the predicted altitude separation of the aircraft at the estimated time to closest approach τ_{ij}. The new strength is chosen to make the derived boolean variable $\mathit{Strength\_Choice}_{ij}$ true. If $\mathit{Sense}_i = \perp$, then it is the case that $\mathit{Strength\_Choice}_{ij} = \text{True}$, otherwise $\mathit{Strength\_Choice}_{ij}$ is given by:

$$
\begin{aligned}
&\mathit{Strength\_Choice}_{ij} = \\
&\quad (\mathit{SEP}_{ij}(\sigma_i, -2000) \geq \texttt{ALIM} \\
&\quad\quad \Rightarrow \mathit{Strength}_i = -2000) \\
&\quad \wedge(\mathit{SEP}_{ij}(\sigma_i, -1000) \geq \texttt{ALIM} > \mathit{SEP}_{ij}(\sigma_i, -2000) \\
&\quad\quad \Rightarrow \mathit{Strength}_i = -1000) \\
&\quad \wedge(\mathit{SEP}_{ij}(\sigma_i, -500) \geq \texttt{ALIM} > \mathit{SEP}_{ij}(\sigma_i, -1000) \\
&\quad\quad \Rightarrow \mathit{Strength}_i = -500) \\
&\quad \wedge(\mathit{SEP}_{ij}(\sigma_i, 0) \geq \texttt{ALIM} > \mathit{SEP}_{ij}(\sigma_i, -500) \\
&\quad\quad \Rightarrow \mathit{Strength}_i = 0) \\
&\quad \wedge(\mathit{SEP}_{ij}(\sigma_i, 1500) \geq \texttt{ALIM} > \mathit{SEP}_{ij}(\sigma_i, 0) \\
&\quad\quad \Rightarrow \mathit{Strength}_i = 1500) \\
&\quad \wedge(\texttt{ALIM} > \mathit{SEP}_{ij}(\sigma_i, 1500) \\
&\quad\quad \Rightarrow \mathit{Strength}_i = 2500)
\end{aligned}
$$

Note that in all cases the weakest (and hopefully least disruptive) strength that guarantees adequate separation is chosen.

4.1.5 Pilot

The pilot automaton models the behavior of the pilot of the aircraft. Upon getting alerted to a threat, the pilot either implements the resolution advisory within a predefined implementation delay, or ignores the resolution advisory. For reasons of simplicity, pilots are assumed to have direct control over the aircraft's acceleration. Moreover, pilots are assumed to implement the resolution advisories by exerting a constant vertical acceleration until the vertical rate proposed by the resolution advisory is reached.

The pilot is modeled by the HIOA $P_i = (U_{P_i},\ X_{P_i},\ Y_{P_i},\ \Sigma_{P_i}^{in},\ \Sigma_{P_i}^{int},\ \Sigma_{P_i}^{out},\ \Theta_{P_i},\ \mathcal{D}_{P_i}\ ,\ \mathcal{W}_{P_i})$ (Appendix F). The input variables are the sense and strength of the RA issued by TCAS 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, 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 an 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, $[\underline{v}_{zi}, \overline{v}_{zi}]$. The width of these ranges reflects considerations such as passenger comfort, standard pilot practice, and the capabilities of the aircraft.

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. To ensure that all advisories can be followed, we impose the following assumption:

Assumption 1 $\underline{a}_{zi} \leq -a < 0 < a \leq \overline{a}_{zi}$, $[-2500, 2500] \subseteq [\underline{v}_{zi}, \overline{v}_{zi}]$ *and at the states* $v_{zi} \in [\underline{v}_{zi}, \overline{v}_{zi}]$.

4.1.6 Communication Channel

The communication channel automaton models the medium through which an aircraft can send messages to neighboring aircraft. We abstract away the details of the communication medium by assuming that messages are queues within the communication channel and are delivered to their destination within given propagation delay bounds.

Communication of intents is achieved through the communication channel HIOA C_{ij} (Appendix E). The automaton has an input action $\mathit{Send}_{ij}(\mathrm{dir})$, for $\mathit{dir} \in \mathit{Dir}$, 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 $\mathit{Receive}_{ji}(\mathrm{dir})$, for $\mathit{dir} \in \mathit{Dir}$. 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.

4.1.7 TCAS and the Closed-Loop System

We define the HIOA TCAS_i as the composition of S_i, D_i, and R_i with all variables in $Y_{S_i} \cup \{\mathit{Threat}_{ij}\}_{j \neq i}$ and all actions in $\Sigma^{out}_{S_i} \cup \Sigma^{out}_{D_i}$ hidden. The interface of the TCAS_i with the outside world is through the input variables $U_{TCAS_i} = U_{S_i} \cup \{\mathit{Mode\_S}_j, \mathit{Equipment}_j\}_{j \neq i}$, the output variables

$Y_{TCAS_i} = \{Sense_i, Strength_i\}$, the input actions $Receive_{ij}(\mathrm{dir})$, for $dir \in Dir$, and the output actions $Send_{ij}(\mathrm{dir})$. It is important to note that if the aircraft is not TCAS-equipped no advisories are produced and no messages are sent.

Definition 1 *The closed-loop system of two aircraft, denoted by PS, is modeled as the composition of A_i, $TCAS_i$, P_i, and C_{ij}, for $i, j \in \{1, 2\}$, $i \neq j$, i.e., $PS = \prod_{i,j \in \{1,2\}, i \neq j} A_i \times TCAS_i \times P_i \times C_{ij}$.*

5 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 our system: a pair of well-behaved and TCAS-equipped aircraft. We then determine the maximum time required for the TCAS system to issue consistent RAs and for the pilots to implement these advisories. 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.

Throughout this section, while formally modeling and analyzing the behavior of the closed-loop TCAS system, we make several assumptions some of which are quite restrictive. Indeed, many of these assumptions may merely be *simplifying assumptions*; that is, assumptions that were made in order to keep the analysis task tractable. However, some of these assumptions may be *intrinsic assumptions*; that is, assumptions that correspond to either undocumented assumptions made during the design phase of the TCAS system, or actual limits of the TCAS system. We argue that the generation of these assumptions is a very important byproduct of the formal analysis using HIOA. These assumptions expose various issues regarding the closed-loop TCAS system that need to be addressed in future research. On one hand, the simplifying assumptions need to be relaxed and, on the other hand, the intrinsic assumptions need to be examined and validated prior to claiming any safety and performance guarantees. An assumption that can neither be relaxed nor validated would indicate that the TCAS system is flawed and, thus, in need of redesign.

In proving the various properties of the closed-loop TCAS system, it was convenient to reason about the discrete and continuous behavior in isolation. For example, we determine the worst-case time bound in obtaining consistent resolution advisories and implementing them by only reasoning about the discrete behavior of the system. Conversely, we determine the projected vertical separation obtained by the various RAs by only reasoning about the continuous behavior of the system. An additional advantage of using the HIOA modeling formalism is that the various discrete and continuous properties are stated in formal statements which can be combined so as to state properties of the hybrid system behavior.

5.1 A Pair of Well-Behaved and TCAS-Equipped Aircraft

In this section, we define a simple and idealized closed-loop TCAS system, *WBS*. 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 attain the resolution strength suggested by the TCAS system. 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.

Definition 2 *A* **well-behaved pair system**, *WBS, is a TCAS system of 2 aircraft which* $Equipment_i =$ *TCAS*, $n_{Ai} = 0$, $n_{ARi} = 0$, $n_{Ri} = 0$, $n_{RRi} = 0$, $Follow_i = \texttt{True}$, $a_{xi} = a_{yi} = 0$, *for* $i \in \{1, 2\}$, *and* $a = \infty$.

For a pair of well-behaved and TCAS-equipped aircraft, *WBS*, the system state is denoted by s, the maximum and minimum pilot delays in implementing RAs are denoted by $\overline{d}_p = \max\{\overline{d}_1, \overline{d}_2\}$ and $\underline{d}_p = \min\{\overline{d}_1, \overline{d}_2\}$, respectively, the maximum and minimum communication delays are denoted by $\overline{d}_c = \max\{\overline{d}_{12}, \overline{d}_{21}\}$ and $\underline{d}_c = \min\{\overline{d}_{12}, \overline{d}_{21}\}$, respectively, and the difference in the aircraft position and velocity components are denoted by $\Delta x = x_1 - x_2$, $\Delta y = y_1 - y_2$, $\Delta z = z_1 - z_2$, $\Delta v_x = v_{x1} - v_{x2}$, $\Delta v_y = v_{y1} - v_{y2}$, and $\Delta v_z = v_{z1} - v_{z2}$. Since the sensors of the aircraft in *WBS* are assumed to be exact, it is the case that 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 exact, 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. To simplify the notation, we also assume that the maximum ascent/descent rates of both aircraft are equal; that is, $\overline{v}_{z1} = \overline{v}_{z2}$

and $\underline{v}_{z1} = \underline{v}_{z2}$. Thus, we define the upper bound on the magnitude of relative vertical speed as $\overline{\Delta v}_z = -\underline{\Delta v}_z = \overline{v}_{z1} - \underline{v}_{z2} = \overline{v}_{z2} - \underline{v}_{z1}$. Finally, we let $\mathcal{S}$ denote the set of states of *WBS* and *Admissible_Execs* denote the set of admissible executions of *WBS*, *i.e.*, $\mathit{Admissible\_Execs} = \{\alpha \in \mathit{execs}(\mathit{WBS}) \mid \alpha.\mathit{fstate} \in \Theta_{\mathit{WBS}} \text{ and } \alpha.\mathit{ltime} = \infty\}$.

We assume that the various parameters used by TCAS remain constant throughout any execution of *WBS*. As discussed in Section 4, these parameters depend on the altitude of the aircraft and on the sensitivity settings of the TCAS system.

Assumption 2 *The parameters* `ALIM`*,* `DMOD`*,* `H1`*,* `TRTHR`*, and* `ZT` *are constant throughout any execution of WBS.*

Without loss of generality, we assume that aircraft 1 is the high priority aircraft.

Assumption 3 $\mathit{Mode\_S}_1 < \mathit{Mode\_S}_2$

In view of only considering TCAS resolutions that utilize nominal resolutions, we assume that the actions $\mathit{Declare}_{ij}$, Sample_i, and $\mathit{Receive}_{ij}(\mathit{dir})$, for $i, j \in \{1, 2\}, i \neq j$ and $\mathit{dir} \in \mathit{Dir}$, only set the $\mathit{Strength}_1$ and $\mathit{Strength}_2$ variables to 1500 *ft/min*. Indeed, since TCAS periodically resets the strength of the RA based on reevaluating the projected vertical separation of the aircraft, this assumption limits the interaction of discrete and continuous behavior in TCAS. We defer the relaxation of this assumption to future research.

Assumption 4 $\mathit{Strength}_1 = \mathit{Strength}_2 = \mathtt{NomStr} = 1500$ *ft/min*

Also, we assume that once pilots get alerted to a threat, they do not oppose the RA suggested by TCAS. For example, following the issuing of a RA that instructs the pilot to climb, it is assumed that the pilot will not decrease the aircraft's vertical speed prior to implementing the RA. This assumption is in agreement with our prior assumption that the pilots follow the RAs proposed by the TCAS system.

Assumption 5 *Pilots do not oppose advisories, i.e., for any state* $s \in \mathcal{S}$, $s.\sigma_i s.a_i \geq 0$, *for* $i \in \{1, 2\}$.

For any state $s \in \mathcal{S}$, we define a derived variable $s.T$ that denotes the time to closest horizontal approach of the aircraft.

Definition 3 *For any state $s \in \mathcal{S}$, let $s.T$ be defined as follows:*

$$T = -\frac{\Delta x \Delta v_x + \Delta y \Delta v_y}{\Delta v_x^2 + \Delta v_y^2}$$

By taking the derivative of T with respect to time, and recalling that $a_{xi} = a_{yi} = 0$, for $i \in \{1, 2\}$, it can be shown that the time to closest horizontal approach is fixed in time.

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 refer to a particular state occurrence of the state s within α.

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.

Definition 4 *For $i, j \in \{1, 2\}, i \neq j$, $Conflict\_Imminent_{ij} = \{s \in \mathcal{S} \mid \exists dir \in Dir : (s.Pre(Declare_{ij}) = \texttt{True}) \vee (s.Pre(Receive_{ij}(dir) = \texttt{True})\}$*

Definition 5 $Conflict\_Imminent = Conflict\_Imminent_{12} \cup Conflict\_Imminent_{21}$

Definition 6 $Non\_Cross = \{s \in \mathcal{S} \mid (s.Sense_1 \neq \bot) \wedge (s.Crossing_1 = \texttt{False})\}$

Definition 7 $Cross = \{s \in \mathcal{S} \mid (s.Sense_1 \neq \bot) \wedge (s.Crossing_1 = \texttt{True})\}$

Lemma 1 $Non\_Cross \cap Cross = \emptyset$.

We define a history boolean variable, *Conflict_Over* $\in$ `Bool`, that records the termination of a conflict. Thus, *Conflict_Over* is `False` initially, and becomes `True` upon the occurrence of either an $Undeclare_{12}$ or $Undeclare_{21}$ action.

Definition 8 $Conflict\_Resolved = \{s \in \mathcal{S} \mid s.Conflict\_Over = \texttt{True}\}$

The following assumption states 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. We realize that this assumption is restricting, but we are interested in analyzing the performance of the TCAS system whenever it is engaged through the time of closest horizontal approach and are not concerned with the cases in which TCAS undeclares prematurely. We defer the analysis of whether the TCAS system actually undeclares prematurely to future research.

Assumption 6 *For any state $s \in \mathcal{S}$ of WBS, it is the case that both $s.Pre(Undeclare_{12})$ and $s.Pre(Undeclare_{21})$ imply that $s.T < 0$.*

5.2 Agreement Protocol

We first define the set of execution fragments, *Stable_Resolution_Frags*, that consists of all execution fragments of *WBS* in which threats are not undeclared and reversals do not occur, *i.e.*, for any execution fragment α in *Stable_Resolution_Frags*, none of the $Sample_1$ and $Sample_2$ actions in α affect the $Sense_1$ and $Sense_2$ variables, respectively, and the $Undeclare_{12}$ and $Undeclare_{21}$ actions are never scheduled within α.

In Table 2, we define sets of states of *WBS* that represent milestones of the protocol used by TCAS to obtain consistent RAs when two aircraft are involved in a conflict. The following lemma specifies that the milestones of Table 2 correspond to the incremental progress in the agreement protocol between the two aircraft involved in the conflict.

Lemma 2

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*.

Table 2: Milestone Sets of the TCAS RA Protocol.

$Local\text{-}Awareness = \{s \in \mathcal{S} \mid$

$(s.Threat_{12} \wedge s.Sense_1 \neq \perp)$

$\vee(s.Threat_{21} \wedge s.Sense_2 \neq \perp)\}$

$Local\text{-}Awareness\text{-}Sent = \{s \in \mathcal{S} \mid$

$(s.Threat_{12} \wedge s.Sense_1 \neq \perp)$

$\vee(s.Threat_{21} \wedge s.Sense_2 \neq \perp$

$\wedge s.Intent\_Sent_{21} \neq \perp)\}$

$Local\text{-}Resolution = \{s \in \mathcal{S} \mid$

$s.Threat_{12} \wedge s.Sense_1 \neq \perp\}$

$Local\text{-}Resolution\text{-}Sent = \{s \in \mathcal{S} \mid s.Threat_{12}$

$\wedge s.Sense_1 \neq \perp \wedge s.Intent\_Sent_{12} = \overline{s.Sense_1}\}$

$Global\text{-}Resolution = \{s \in \mathcal{S} \mid$

$s.Threat_{12} \wedge s.Threat_{21} \wedge s.Sense_2 \neq s.Sense_1$

$\wedge s.Sense_1 \neq \perp \wedge s.Intent\_Sent_{12} = \overline{s.Sense_1}$

$\wedge s.Sense_2 \neq \perp \wedge s.mset_{12} = \emptyset\}$

$Global\text{-}Resolution\text{-}Sent = \{s \in \mathcal{S} \mid$

$s.Threat_{12} \wedge s.Threat_{21} \wedge s.Sense_1 \neq s.Sense_2$

$\wedge s.Sense_1 \neq \perp \wedge s.Intent\_Sent_{12} = \overline{s.Sense_1}$

$\wedge s.Sense_2 \neq \perp \wedge s.Intent\_Sent_{21} = \overline{s.Sense_2}$

$\wedge s.mset_{12} = \emptyset\}$

$Global\text{-}Agreement = \{s \in \mathcal{S} \mid$

$s.Threat_{12} \wedge s.Threat_{21} \wedge s.Sense_1 \neq s.Sense_2$

$\wedge s.Sense_1 \neq \perp \wedge s.Intent\_Sent_{12} = \overline{s.Sense_1}$

$\wedge s.Sense_2 \neq \perp \wedge s.Intent\_Sent_{21} = \overline{s.Sense_2}$

$\wedge s.mset_{12} = \emptyset \wedge s.mset_{21} = \emptyset\}$

The following lemma specifies the time that is needed to progress through some of 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 advisory is reached within $2\overline{d}_c$, and the protocol terminates within $3\overline{d}_c$. It is important to note that once consistent RAs are obtained, 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.

Lemma 3 *For any finite execution fragment α of WBS in Stable_Resolution_Frags such that $\alpha.fstate \in$ Local-Awareness it is the case that:*

1. $\alpha.ltime > \overline{d}_c \implies \alpha.lstate \in$ *Local-Resolution,*
2. $\alpha.ltime > 2\overline{d}_c \implies \alpha.lstate \in$ *Global-Resolution,*
3. $\alpha.ltime > 3\overline{d}_c \implies \alpha.lstate \in$ *Global-Agreement, and*
4. $\alpha.ltime > 3\overline{d}_c + \overline{d}_p \implies (\neg s.Follow_1 \vee (\sigma s.v_{z1} \geq \texttt{NomStr})) \wedge (\neg s.Follow_2 \vee (-\sigma s.v_{z2} \geq \texttt{NomStr}))$, *where $s = \alpha.lstate$ and $\sigma = 1$, if $s.Sense_1 = Climb$, and $\sigma = -1$ otherwise.*

5.3 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. It is important to note that 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. In this case, it is worth noting that 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. Recall that it is only possible to reverse out of a crossing RA.

These execution categories of *WBS* are formally defined below.

Definition 9 $\mathit{Conflict\_Free\_Execs} = \{\alpha \in \mathit{Admissible\_Execs} \mid \forall\, s \in \alpha, s \notin \mathit{Conflict\_Imminent}\}$

Definition 10 $\mathit{Non\_Crossing\_Execs} = \{\alpha \in \mathit{Admissible\_Execs} \mid \exists\, s_1 \in \alpha, s_1 \in \mathit{Non\_Cross} : \forall\, s_2 \in \alpha, s_2 \leq s_1,$ *it is the case that* $s_2 \notin \mathit{Cross}\}$

Definition 11 $\mathit{Crossing\_Execs} = \{\alpha \in \mathit{Admissible\_Execs} \mid \exists\, s_1 \in \alpha, s_1 \in \mathit{Cross} : \forall\, s_2 \in \alpha, s_1 \leq s_2,$ *it is the case that*
$(s_2 \in \mathit{Cross}) \vee (s_2.\mathit{Sense}_1 s_2.\Delta z \geq 0) \vee (s_2 \in \mathit{Conflict\_Resolved})\}$.

Definition 12 $\mathit{Reversing\_Execs} = \{\alpha \in \mathit{Admissible\_Execs} \mid \exists\, s_1 \in \alpha, s_1 \in \mathit{Cross} : \exists\, s_2 \in \alpha, s_1 \leq s_2,$ *such that*
$(s_2 \in \mathit{Non\_Cross}) \wedge (s_1.\mathit{Sense}_1 s_2.\Delta z < 0) \wedge (s_2 \notin \mathit{Conflict\_Resolved})\}$.

Definition 13 $\mathit{Conflict\_Execs} = \mathit{Non\_Crossing\_Execs} \cup \mathit{Crossing\_Execs} \cup \mathit{Reversing\_Execs}$

Lemma 4 *The sets Conflict_Free_Execs, Non_Crossing_Execs, Crossing_Execs, and Reversing_Execs are pairwise disjoint.*

Lemma 5 $\mathit{Admissible\_Execs} = \mathit{Conflict\_Free\_Execs} \cup \mathit{Conflict\_Execs}$

We proceed by defining the set of safe executions of *WBS*. The safe executions are defined to be the executions in which the aircraft are sufficiently separated in altitude at closest horizontal approach.

Definition 14 $\mathit{Safe\_Execs} = \{\alpha \in \mathit{Admissible\_Execs} \mid \forall\, s \in \alpha, (s.T = 0) \Rightarrow (|s.\Delta z| \geq \mathtt{ALIM})\}$

The following assumption states that the set of conflict-free executions are safe. In this paper, we assume that the TCAS system declares a conflict whenever there is a potential mid-air collision. 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.

Assumption 7 $\mathit{Conflict\_Free\_Execs} \subseteq \mathit{Safe\_Execs}$

5.4 Safety Conditions

For an execution $\alpha \in \mathit{Conflict\_Execs}$, we define $\alpha.s_0 \in \alpha$ such that $\alpha.s_0 \in \mathit{Conflict\_Imminent}_{12}$ and for all $s \in \alpha$, $s < \alpha.s_0$, it is the case that $s \notin \mathit{Conflict\_Imminent}_{12}$; that is, $\alpha.s_0$ is 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 $\alpha.s_0$. We also keep track of the time elapsing from the initial declaration of a threat by the high priority aircraft through an auxiliary variable t_{RA}. Although discrete transitions do not affect the value of t_{RA}, along trajectories, it is the case that $\dot{t}_{RA} = 1$. Let D be an upper bound on the delay from the time in which the high priority aircraft gets alerted to the 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.

Assumption 8 $D > d$

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.

Assumption 9 *For any execution $\alpha \in \mathit{Conflict\_Execs}$, it is the case that $\alpha.T_0 > D$.*

5.4.1 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 $s \in \mathcal{S}$, we define the derived variable P_{NC} as follows:

$$s.P_{NC} = |s.\Delta z| + s.\underline{\Delta v}_{NC} D + 2\texttt{NomStr}(s.T - D)$$

where $s.\underline{\Delta v}_{NC} = \min(2\texttt{NomStr}, \sigma s.v_{z1} + \max(\underline{v}_z, -\sigma s.v_{z2} - \overline{a}_z \overline{d}_c))$ 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.

We proceed by defining 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.

Definition 15 *Non_Cross_Safe* $= \{s \in \mathcal{S} \mid s.P_{NC} \geq \texttt{ALIM}\}$.

Lemma 6 *If $\alpha \in$ Non_Crossing_Execs and $s_1 \in \alpha$ such that $D < s_1.t_{RA} < \alpha.T_0$, then it is the case that $\sigma s_1.\Delta z \geq \sigma s_0.\Delta z + s_0.\underline{\Delta v}_{NC} D$, where $\sigma = \mathrm{sign}(s_0.v_{z1} - s_0.v_{z2})$ and $s_0.\underline{\Delta v}_{NC} = \min(2\texttt{NomStr}, \sigma s_0.v_{z1} + \max(\underline{v}_z, -\sigma s_0.v_{z2} - \overline{a}_z \overline{d}_c))$.*

Lemma 7 *If $\alpha \in$ Non_Crossing_Execs and $s_1, s_2 \in \alpha$ such that $D < s_1.t_{RA} < \alpha.T_0$ and $s_2.t_{RA} = \alpha.T_0$, then it is the case that $\sigma s_2.\Delta z \geq \sigma s_1.\Delta z + 2\texttt{NomStr}(s_2.t_{RA} - s_1.t_{RA})$, where $\sigma = \mathrm{sign}(s_0.v_{z1} - s_0.v_{z2})$.*

Corollary 8 *If $\alpha \in$ Non_Crossing_Execs and $s_2 \in \alpha$ such that $s_2.t_{RA} = \alpha.T_0$, then it is the case that $\sigma s_2.\Delta z \geq \sigma s_0.\Delta z + s_0.\underline{\Delta v}_{NC} D + 2\texttt{NomStr}(\alpha.T_0 - D)$, where $\sigma = \mathrm{sign}(s_0.v_{z1} - s_0.v_{z2})$ and $s_0.\underline{\Delta v}_{NC} = \min(2\texttt{NomStr}, \sigma s_0.v_{z1} + \max(\underline{v}_z, -\sigma s_0.v_{z2} - \overline{a}_z \overline{d}_c))$.*

Lemma 9 *If $\alpha \in$ Non_Crossing_Execs and $\alpha.s_0 \in$ Non_Cross_Safe, then $\alpha \in$ Safe_Execs.*

5.4.2 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 $s \in \mathcal{S}$, we define the derived variable P_C as follows:

$$s.P_C = -|s.\Delta z| + s.\underline{\Delta v}_C D + 2\texttt{NomStr}(s.T - D)$$

where $s.\underline{\Delta v}_C = \min(2\texttt{NomStr}, -\sigma s.v_{z1} + \max(\underline{v}_z, \sigma s.v_{z2} - \overline{a}_z \overline{d}_c))$ 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.

We proceed by defining 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.

Definition 16 *Cross_Safe* $= \{s \in \mathcal{S} \mid s.P_C \geq \texttt{ALIM}\}$.

Lemma 10 *If $\alpha \in$ Crossing_Execs and $s_1 \in \alpha$ such that $D < s_1.t_{RA} < \alpha.T_0$, then it is the case that $-\sigma s_1.\Delta z \geq -\sigma s_0.\Delta z + s_0.\underline{\Delta v}_C D$, where $\sigma = \text{sign}(s_0.v_{z1} - s_0.v_{z2})$ and $s_0.\underline{\Delta v}_C = \min(2\texttt{NomStr}, -\sigma s_0.v_{z1} + \max(\underline{v}_z, \sigma s_0.v_{z2} - 2\overline{a}_z \overline{d}_c))$.*

Lemma 11 *If $\alpha \in$ Crossing_Execs and $s_1, s_2 \in \alpha$ such that $D < s_1.t_{RA} < \alpha.T_0$ and $s_2.t_{RA} = \alpha.T_0$, then it is the case that $-\sigma s_2.\Delta z \geq -\sigma s_1.\Delta z + 2\texttt{NomStr}(s_2.t_{RA} - s_1.t_{RA})$, where $\sigma = \text{sign}(s_0.v_{z1} - s_0.v_{z2})$.*

Corollary 12 *If $\alpha \in$ Crossing_Execs and $s_2 \in \alpha$ such that $s_2.t_{RA} = \alpha.T_0$, then it is the case* *that* $-\sigma s_2.\Delta z \geq -\sigma s_0.\Delta z + s_0.\underline{\Delta v}_C D + 2\texttt{NomStr}(\alpha.T_0 - D)$, *where* $\sigma = \text{sign}(s_0.v_{z1} - s_0.v_{z2})$ *and* $s_0.\underline{\Delta v}_C = \min(2\texttt{NomStr}, -\sigma s_0.v_{z1} + \max(\underline{v}_z, \sigma s_0.v_{z2} - 2\overline{a}_z\overline{d}_c))$.

Lemma 13 *If* $\alpha \in$ *Crossing_Execs and* $\alpha.s_0 \in$ *Cross_Safe, then* $\alpha \in$ *Safe_Execs.*

5.4.3 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. Note that up to $2D$ time units after the first threat declaration, the aircraft could be engaging in a trajectory that opposes the non-crossing RA sense. For all $s \in \mathcal{S}$, the derived variable $P_R^<$ is defined as follows:

$$s.P_R^< = |s.\Delta z| + s.\underline{\Delta v}_R^<(2D) + 2\texttt{NomStr}(s.T - 2D)$$

where $s.\underline{\Delta v}_R^< = \min(2\texttt{NomStr}, \max(\underline{v}_z, \sigma s.v_{z1} - \overline{a}_z D) + \max(\underline{v}_z, -\sigma s.v_{z2} - \overline{a}_z(D + \overline{d}_c)))$ 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.

In this case, the intuitive understanding of the function $P_R^<$ 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 the following inequality is satisfied:

$$s.P_R^< \geq \texttt{ALIM}$$

Thus, in all reversing executions where the reversal occurs earlier than the point in time when the initial crossing execution gets implemented, the above inequality guarantees that the aircraft will attain the necessary vertical separation by the time of closest horizontal approach.

Lemma 14 *Let $\alpha \in Reversing\_Execs$ be an execution of WBS in which the reversal occurs prior to D time units after the first threat declaration by aircraft 1. If $s_0.P_R^< \geq \texttt{ALIM}$, then $\alpha \in Safe\_Execs$.*

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 $s.T$.

Let 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 as follows:

$$SEP_{12}(-\sigma, \texttt{NomStr}) < \texttt{ALIM}$$
$$\Rightarrow \sigma\Delta z - \sigma\Delta v_z\tau_{12} < \texttt{ALIM}, \text{ and}$$

$$SEP_{12}(\sigma, \texttt{NomStr}) \geq \texttt{ALIM}$$
$$\Rightarrow \begin{cases} \sigma\Delta z + \sigma\Delta v_z d + (\texttt{NomStr} - \sigma v_{z2})(\tau_{12} - d) \geq \texttt{ALIM} \\ \quad \text{if } \tau_{12} > d, \text{or} \\ \sigma\Delta z + \sigma\Delta v_z\tau_{12} \geq \texttt{ALIM} \\ \quad \text{otherwise} \end{cases}$$

i.e., a reversal is warranted only if the crossing advisory is unsafe and the non-crossing advisory is safe. Rearranging each of the inequalities corresponding to $SEP_{12}(\sigma, \texttt{NomStr}) \geq \texttt{ALIM}$, we get:

$$\sigma\Delta z \geq \texttt{ALIM} - \sigma\Delta v_z d - (\texttt{NomStr} - \sigma v_{z2})(\tau_{12} - d)$$
$$\quad \text{if } \tau_{12} > d, \text{or}$$
$$\sigma\Delta z \geq \texttt{ALIM} - \sigma\Delta v_z\tau_{12}$$
$$\quad \text{otherwise}$$

Since we are assuming that the crossing advisory has already been implemented, it follows that $-\sigma v_{z1} \geq \texttt{NomStr}$ and $\sigma v_{z2} \geq \texttt{NomStr}$. 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, we obtain the pair of conservative inequalities, for state s'':

if $\tau_{12} > d$ then:

$$\begin{aligned}\sigma\Delta z &\geq \texttt{ALIM} - \sigma\Delta v_z d - (\texttt{NomStr} - \sigma v_{z2})(\tau_{12} - d)\\ &\geq \texttt{ALIM} + 2\texttt{NomStr}d - (\texttt{NomStr} - \sigma v_{z2})(\tau_{12} - d)\\ &\geq \texttt{ALIM} + 2\texttt{NomStr}d\\ &> \texttt{ALIM}\end{aligned}$$

else:

$$\begin{aligned}\sigma\Delta z &\geq \texttt{ALIM} - \sigma\Delta v_z \tau_{12}\\ &\geq \texttt{ALIM} + 2\texttt{NomStr}\tau_{12}\\ &\geq \texttt{ALIM}\end{aligned}$$

Thus, in order for an aircraft to reverse out of an implemented crossing RA, $-\sigma$, it the case that $\sigma\Delta z \geq \texttt{ALIM}$, *i.e.*, the aircraft altitude separation must be greater than or equal to $\texttt{ALIM}$. 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 $\sigma\Delta z \geq \texttt{ALIM}$ could be violated, *i.e.*, $\overline{T}_R$ is bounded by the inequality:

$$-|\Delta z| + s.\underline{\Delta v}_C D + 2\texttt{NomStr}(\overline{T}_R - D) \leq -\texttt{ALIM}$$

Solving for $\overline{T}_R$ we have:

$$\overline{T}_R \leq \frac{-\texttt{ALIM} + |\Delta z| - s.\underline{\Delta v}_C D + 2\texttt{NomStr}D}{2\texttt{NomStr}}$$

It is important to note that 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, *i.e.*, we have reached a contradiction. For simplicity, we assume that if $\overline{T}_R$ turns out to be negative, then 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, $P_{\overline{R}}^{\geq}$, 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 $s \in \mathcal{S}$, the derived variable $P_{\overline{R}}^{\geq}$ is defined as follows:

$$s.P_{\overline{R}}^{\geq} = \mathtt{ALIM} + s.\underline{\Delta v}_{\overline{R}}^{\geq} D + 2\mathtt{NomStr}(T - \overline{T}_R - D)$$

where $s.\underline{\Delta v}_{\overline{R}}^{\geq} = \min(2\mathtt{NomStr}, \max(\underline{v}_z, \sigma s.v_{z1} - \overline{a}_z \overline{T}_R) + \max(\underline{v}_z, -\sigma s.v_{z2} - \overline{a}_z(\overline{T}_R + \overline{d}_c)))$ with $\sigma = \mathrm{sign}(s.v_{z1} - s.v_{z2})$, *i.e.*, σ is the non-crossing RA sense of aircraft 1.

Plugging in the value for $\overline{T}_R$ and simplifying, we get:

$$\begin{aligned} s.P_{\overline{R}}^{\geq} =& 2\mathtt{ALIM} - |\Delta z| + (s.\underline{\Delta v}_C + s.\underline{\Delta v}_{\overline{R}}^{\geq}) D \\ &+ 2\mathtt{NomStr}(T - 2D) \end{aligned}$$

It follows that the state s is safe in this type of an execution if the following inequality is satisfied:

$$s.P_{\overline{R}}^{\geq} \geq \mathtt{ALIM}$$

Thus, in all reversing executions where the reversal occurs no earlier than the point in time in which the initial crossing execution gets implemented, the above inequality guarantees that the aircraft will attain the necessary vertical separation by the time of closest horizontal approach.

Lemma 15 *Let $\alpha \in$ Reversing_Execs be an execution of WBS in which the reversal occurs no earlier than D time units after the first threat declaration by aircraft 1. If $s_0.P_{\overline{R}}^{\geq} \geq \mathtt{ALIM}$, then $\alpha \in$ Safe_Execs.*

We continue by defining a set of states from which any type of reversing execution would result 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.

Definition 17 *Reverse_Safe* $= \{s \in \mathcal{S} \mid (s.P_R^{<} \geq \texttt{ALIM}) \wedge ((s.P_R^{\geq} \geq \texttt{ALIM}) \vee (\overline{T}_R < D))\}$.

Lemma 16 *If $\alpha \in$ Reversing_Execs and $\alpha.s_0 \in$ Reverse_Safe, then $\alpha \in$ Safe_Execs.*

5.5 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.

5.5.1 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. The following theorem states that if the initial state of a conflict satisfies all per-category safety conditions, then the execution is a safe execution.

Theorem 17 *If $\alpha \in$ Conflict_Execs and $\alpha.s_0 \in$ Non_Crossing_Safe $\cap$ Crossing_Safe $\cap$ Reverse_Safe, then $\alpha \in$ Safe_Execs.*

Albeit simple, Theorem 17 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.

5.5.2 Isolating Non-Crossing Executions

In this section, we remove some of the conservatism of Theorem 17 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 $Receive_{12}(\text{dir})$ action, where $\text{dir} \in \text{Dir}$. 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 $Receive_{12}(\text{dir})$ action, where $\text{dir} \in \text{Dir}$, 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 $Receive_{12}(\text{dir})$ action, where $\text{dir} \in \text{Dir}$, be denoted by $\alpha.s_0$ and the state following the scheduling of the action as $\alpha.s'_0$. Moreover, we define the derived variable C to denote whether the choice of a crossing RA is possible from the perspective of aircraft 1, which is the high priority aircraft:

$$
\begin{aligned}
s.C = \exists\, \sigma \in \{1, -1\} &\text{ such that} \\
&(\sigma s.\Delta z < 0) \\
&\wedge (Sep_{12}(\sigma, \texttt{NomStr}) \geq \texttt{ALIM} \\
&\wedge Sep_{12}(-\sigma, \texttt{NomStr}) < \texttt{ALIM})
\end{aligned}
$$

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.

Definition 18 $\mathit{Cross\_Impossible} = \{s \in \mathcal{S} \mid s.C = \texttt{False}\}$

Lemma 18 *If $\alpha \in \mathit{Conflict\_Execs}$ and $\alpha.s_0 \in \mathit{Cross\_Impossible}$, then $\alpha \in \mathit{Non\_Crossing\_Execs}$.*

Theorem 19 *If $\alpha \in \mathit{Conflict\_Execs}$ and $\alpha.s_0 \in (\mathit{Cross\_Impossible} \cap \mathit{Non\_Cross\_Safe}) \cup (\mathit{Non\_Cross\_Safe} \cap \mathit{Cross\_Safe} \cap \mathit{Reverse\_Safe})$, then $\alpha \in \mathit{Safe\_Execs}$.*

5.5.3 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.

We proceed by defining the set of executions of *WBS* in which the aircraft are separated in altitude by at most Kft when aircraft 1 is alerted to a threat.

Definition 19 $\mathit{Close\text{-}in\text{-}Alt\_Execs} = \{\alpha \in \mathit{Conflict\_Execs} \mid s = \alpha.s_0, |s.\Delta v_z| \leq K\}$

The safety condition for executions in *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:

$$-K + s.\underline{\Delta v}_C D + 2\texttt{NomStr}(\overline{T}_C - D) \leq 0$$

Solving for $\overline{T}_C$ we have:

$$\overline{T}_C \leq \frac{K - s.\underline{\Delta v}_C D + 2\texttt{NomStr} D}{2\texttt{NomStr}}$$

If $-K + s.\underline{\Delta v}_C D > 0$, then the above calculation for $\overline{T}_C$ will result in a negative value. However, such cases imply 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 we can use the value of D time units as an upper bound on the delay in reversing.

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 $s \in \mathcal{S}$, the derived variable P is defined as follows:

$$s.P = s.\underline{\Delta v}_R D + 2\texttt{NomStr}(T - \max(D, \overline{T}_C) - D)$$

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 the following inequality is satisfied:

$$s.P \geq \texttt{ALIM}$$

We proceed by defining the set of states of *WBS* from which a close in altitude encounter results in adequate separation in altitude at closest horizontal approach.

Definition 20 *Close-in-Alt_Safe* $= \{s \in \mathcal{S} \mid s.P \geq \texttt{ALIM}\}$.

Theorem 20 *If $\alpha \in$ Close-in-Alt_Execs and $\alpha.s_0 \in$ Close-in-Alt_Safe, then $\alpha \in$ Safe_Execs.*

6 Conclusions and Future Research

We demonstrate how high-level modeling techniques involving the HIOA model can be used to model and analyze complex safety-critical hybrid systems such as the Traffic Alert and Collision Avoidance System (TCAS). In our presentation, we define HIOA models of the core components of closed-loop system involving aircraft, pilots, TCAS system components, and the communication channels used by the aircraft to communicate among themselves. Then, we define an idealized system involving two TCAS-equipped and "well-behaved" aircraft — that is, aircraft that do not accelerate in the horizontal plane and whose pilots follow the RAs issued by their TCAS system. We proceed by splitting the aircraft encounters into categories and providing safety conditions for each such category. Finally, we combine the per-category safety conditions into overall safety conditions. The contributions of this paper are the high-level models of the TCAS system, which may be used as a basis for studying a wide range of TCAS properties, and the demonstration of the usefulness and practicality of high-level modeling and analysis techniques.

Clearly, this paper constitutes the first step toward a comprehensive analysis and verification of TCAS. The analysis presented in this paper needs to be extended by relaxing several of the restricting assumptions of the pair of well-behaved and TCAS-equipped aircraft and extending the model to describe more detailed and realistic views of the closed-loop system components. In the case of the pair of well-behaved and TCAS-equipped aircraft, we need to consider cases in which multiple aircraft are simultaneously involved in a conflict, pilots accelerate in the x and y directions, and pilots thwart TCAS's resolution advisories. In the case of the TCAS model, we ought to progressively remove the various simplifications in the aircraft dynamics, the TCAS algorithm, and the pilot response. When relaxing the various assumptions and removing the various simplifications made in our analysis, model refinement may be used to extend the results of the high-level models to their more refined counterparts; that is, by showing that the more refined system model actually implements the high-level model, the properties shown in this paper extend to the more refined model. Finally, it would be interesting to see whether, given a sufficiently detailed system model, the software for the discrete parts of the system could be generated automatically such that, by construction, the software generated is a truthful implementation of the higher level specifications.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments and suggestions as to how to improve the content and the presentation of this paper.

References

[1] Radio Technical Commission for Aeronautics (SC-147), "Traffic Alert and Collision Avoidance System (TCAS) I Functional Guidelines," Tech. Rep. RTCA/DO-184, RTCA, May 1983, Specifications of TCAS I.

[2] Radio Technical Commission for Aeronautics (SC-147), "Minimum Operational Performance Standards for Traffic Alert and Collision Avoidance System II (TCAS II) Airborne Equipment," Tech. Rep. RTCA/DO-185, RTCA, Sept. 1990, Consolidated Edition, Specifications of TCAS II-6.04a.

[3] Radio Technical Commission for Aeronautics (SC-147), "Minimum Operational Performance Standards for Traffic Alert and Collision Avoidance System II (TCAS II) Airborne Equipment," Tech. Rep. RTCA/DO-185A, RTCA, Dec. 1997, Specifications of TCAS II version 7.0.

[4] Nancy A. Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg, "Hybrid I/O Automata," in *Proc. DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems, Hybrid Systems III: Verification and Control*, R. Alur, T. Henzinger, and E. Sontag, Eds. 1996, vol. 1066 of *Lecture Notes in Computer Science*, pp. 496–510, Springer-Verlag, The DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems took place in New Brunswick, New Jersey, in October 1995.

[5] Nancy A. Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg, "Hybrid Automata," Preprint/Work in Progress. Preliminary versions appeared as Refs. 29 and 4, Apr. 1999.

[6] Nancy G. Leveson, Mats P.E. Heimdahl, Holly Hildreth, and Jon D. Reese, "Requirements Specification for Process-Control Systems," *IEEE Transactions on Software Engineering*, vol. SE-20, no. 9, pp. 684–707, Sept. 1994.

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

[8] Ekaterina Dolginova and Nancy A. Lynch, "Safety Verification for Automated Platoon Maneuvers: A Case Study," in *Proc. International Workshop on Hybrid and Real-Time Systems (HART'97)*, Oded Maler, Ed. 1997, vol. 1201 of *Lecture Notes in Computer Science*, pp. 154–170, Springer-Verlag, The International Workshop on Hybrid and Real-Time Systems took place in Grenoble, France, in March 1997.

[9] H. B. Weinberg, Nancy A. Lynch, and Norman Delisle, "Verification of Automated Vehicle Protection Systems," in *Hybrid Systems III: Verification and Control*, R. Alur, T. Henzinger, and E. Sontag, Eds. 1996, vol. 1066 of *Lecture Notes in Computer Science*, pp. 101–113, Springer-Verlag.

[10] Carolos Livadas, "Formal Verification of Safety-Critical Hybrid Systems," Master of Engineering Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, Sept. 1997.

[11] Constance Heitmeyer and Nancy A. Lynch, "The Generalized Railroad Crossing: A Case Study in Formal Verification of Real-Time Systems," in *Proc. 15th IEEE Real-Time Systems Symposium (RTSS'94)*, San Juan, Puerto Rico, Dec. 1994, pp. 120–131, IEEE Computer Society.

[12] John Lygeros, George J. Pappas, and Shankar Sastry, "An Approach to the Verification of the Center-TRACON Automation System," in *Hybrid Systems: Computation and Control (HSCC'98)*, Thomas A. Henzinger and Shankar Sastry, Eds. 1998, vol. 1386 of *Lecture Notes in Computer Science*, pp. 289–304, Springer-Verlag, The First International Workshop, Hybrid Systems: Computation and Control (HSCC'98) took place in Berkeley, California, in April 1998.

[13] John Lygeros and Nancy A. Lynch, "On the Formal Verification of the TCAS Conflict Resolution Algorithm," in *36th IEEE Conference on Decision and Control (CDC'97)*, San Diego, CA, Dec. 1997, pp. 1829–1834.

[14] Carolos Livadas, John Lygeros, and Nancy A. Lynch, "High-Level Modeling and Analysis of TCAS," in *Proc. 20th IEEE Real-Time Systems Symposium (RTSS'99)*, Phoenix, Arizona, Dec. 1999, pp. 115–125, IEEE Computer Society.

[15] David Harel, "StateCharts: A Visual Formalism for Complex Systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, June 1987.

[16] Nancy G. Leveson, "Intent Specifications: An Approach to Building Human-Centered Specifications," *IEEE Transactions on Software Engineering*, To appear.

[17] Daniel Jackson and John Chapin, "Simplifying Air-traffic Control: A Case Study in Software Design," Unpublished work, July 1999.

[18] Rainer Gawlick, Roberto Segala, Jørgen Søgaard-Andersen, and Nancy A. Lynch, "Liveness in Timed and Untimed Systems," in *Proc. 21st International Colloquium on Automata, Languages and Programming (ICALP'94)*, Serge Abiteboul and Eli Shamir, Eds. 1994, vol. 820 of *Lecture Notes in Computer Science*, pp. 166–177, Springer-Verlag, The 21st International Colloquium on Automata, Languages and Programming (ICALP'94) took place in Jerusalem, Israel, in July 1994. Full version appeared as Ref. 30.

[19] Nancy A. Lynch and Frits Vaandrager, "Forward and Backward Simulations — Part II: Timing-Based Systems," *Information and Computation*, vol. 128, no. 1, pp. 1–25, July 1996, Preliminary version appeared as Ref. 31.

[20] Rajeev Alur and David L. Dill, "A Theory of Timed Automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994, Preliminary version appeared as Ref. 32.

[21] Leslie Lamport, "The Temporal Logic of Actions," *Transactions on Programming Languages and Systems*, vol. 16, no. 3, pp. 872–923, May 1994, Also appeared as Ref. 33.

[22] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli, "Temporal Proof Methodologies for Timed Transition Systems," *Information and Computation*, vol. 112, no. 2, pp. 273–337, Aug. 1994, Preliminary versions of Part I and Part II appeared as Refs. 34 and 35, respectively.

[23] Oded Maler, Zohar Manna, and Amir Pnueli, "From Timed to Hybrid Systems," in *Proc. REX Workshop "Real-Time: Theory in Practice"*, J.W. de Bakker, K. Huizing, W.P. de Roever, and G. Rozenberg, Eds., vol. 600 of *Lecture Notes in Computer Science*, pp. 447–484. Springer-Verlag, 1992, The REX Workshop "Real-Time: Theory in Practice" took place in Mook, The Netherlands, in June 1991.

[24] Zohar Manna and Amir Pnueli, "Verifying Hybrid Systems," in *Hybrid Systems*, Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, Eds. 1993, vol. 736 of *Lecture Notes in Computer Science*, pp. 4–35, Springer-Verlag.

[25] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho, "Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems," in *Hybrid Systems*. 1993, vol. 736 of *Lecture Notes in Computer Science*, pp. 209–229, Springer-Verlag, Extended version appeared as Ref. 26.

[26] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine, "The Algorithmic Analysis of Hybrid Systems," *Theoretical Computer Science*, vol. 138, no. 1, pp. 3–34, Feb. 1995, Preliminary version appeared as Ref. 36.

[27] W. Dwight Love, "Preview of TCAS II Version 7," *Air Traffic Control Quarterly*, vol. 6, no. 4, pp. 231–247, Dec. 1998.

[28] Anne C. Drumm, "Personal Communication," MIT Lincoln Laboratory, Nov. 1997.

[29] Nancy A. Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg, "Hybrid I/O Automata," Technical Memo MIT/LCS/TM-544, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, Dec. 1995.

[30] Rainer Gawlick, Roberto Segala, Jørgen Søgaard-Andersen, and Nancy A. Lynch, "Liveness in Timed and Untimed Systems," Technical Report MIT/LCS/TR-587, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, Dec. 1993.

[31] Nancy A. Lynch and Frits Vaandrager, "Forward and Backward Simulations — Part II: Timing-Based Systems," Technical Memo MIT/LCS/TM-487.c, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, Apr. 1995.

[32] Rajeev Alur and David L. Dill, "Automata for Modeling Real-Time Systems," in *Proc. 17th International Colloquium on Automata, Languages and Programming (ICALP'90)*. 1990, vol. 443 of *Lecture Notes in Computer Science*, pp. 322–335, Springer-Verlag.

[33] Leslie Lamport, "The Temporal Logic of Actions," Research Report 79, Digital Equipment Corporation Systems Research Center, Palo Alto, California, Dec. 1991.

[34] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli, "Timed Transition Systems," in *Proc. REX Workshop "Real-Time: Theory in Practice"*, J.W. de Bakker, K. Huizing, W.P. de Roever, and G. Rozenberg, Eds., vol. 600 of *Lecture Notes in Computer Science*, pp. 226–251. Springer-Verlag, 1992, The REX Workshop "Real-Time: Theory in Practice" took place in Mook, The Netherlands, in June 1991.

[35] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli, "Temporal Proof Methodologies for Real-Time Systems," in *Proc. 18th Annual Symposium on Principles of Programming Languages*. 1991, pp. 353–366, ACM Press, New York.

[36] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine, "The Algorithmic Analysis of Hybrid Systems," in *Proc. 11th International Conference on Analysis and Optimization of Systems, Discrete-Event Systems*. 1994, vol. 199 of *Lecture Notes in Control and Information Sciences*, pp. 331–351, Springer-Verlag.

A Aircraft Automaton A_i

Variables:

Input:

$a_i \in \mathbb{R}^3$

Output:

$Mode\_S_i \in \mathbb{N}$, initially arbitrary

$Equipment_i \in \{\text{None}, \text{Report}, \text{TCAS}\}$, initially arbitrary

$p_i \in \mathbb{R}^3$, initially arbitrary with $z_i \geq 0$

$v_i \in \mathbb{R}^3$, initially arbitrary

Actions:

Input:

e, the environment action

Discrete Transitions:

e:

Effect: Arbitrarily reset the input variables.

Trajectories:

Input variables follow arbitrary trajectories.

$Mode\_S_i$ and $Equipment_i$ remain constant.

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

B Sensor Automaton S_i

Variables:

Input:

$p_j \in \mathbb{R}^3$ for all $j \in$ Aircraft

$v_j \in \mathbb{R}^3$ for all $j \in$ Aircraft

Internal:

$t_{S_i} \in \mathbb{R}^+$, initially 0

Output:

$R_{ij} \in \mathbb{R}^+$, for all $j \in$ Others$_i$, initially 0

$\dot{R}_{ij} \in \mathbb{R}$, for all $j \in$ Others$_i$, initially 0

$h_{ij} \in \mathbb{R}^+$, for all $j \in$ Aircraft, initially 0

$\dot{h}_{ij} \in \mathbb{R}$, for all $j \in$ Aircraft, initially 0

Actions:

Input:

e, the environment action

Output:

$Sample_i$

Discrete Transitions:

e:

Effect: Arbitrarily reset the input variables.

$Sample_i$:

Precondition:

$t_{S_i} = T_{S_i}$

Effect:

$t_{S_i} := 0$

$h_{ij} :\in \left[z_j \pm \frac{n_{Ai}}{2}\right]$

$\dot{h}_{ij} :\in \left[z_j \pm \frac{n_{ARi}}{2}\right]$

$R_{ij} :\in \left[\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \pm \frac{n_{Ri}}{2}\right]$

$\dot{R}_{ij} :\in \left[\frac{(x_i-x_j)(v_{xi}-v_{xj})+(y_i-y_j)(v_{yi}-v_{yj})+(z_i-z_j)(v_{zi}-v_{zj})}{\sqrt{(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2}} \pm \frac{n_{RRi}}{2}\right]$

Trajectories:

Input variables follow arbitrary trajectories.

$\dot{t}_{S_i} = 1$

Output variables remain constant.

Trajectories stop once the $Sample_i$ action is enabled.

C Conflict Detection Automaton D_i

Variables:

Input:

$Threat_{ij} \in$ `Bool` for all $j \in \text{Others}_i$

$R_{ij} \in \mathbb{R}^+$, for all $j \in \text{Others}_i$

$\dot{R}_{ij} \in \mathbb{R}$, for all $j \in \text{Others}_i$

$h_{ij} \in \mathbb{R}^+$, for all $j \in$ Aircraft

$\dot{h}_{ij} \in \mathbb{R}$, for all $j \in$ Aircraft

Derived:

$Range\_Test_{ij} \in$ `Bool` (see text)

$Altitude\_Test_{ij} \in$ `Bool` (see text)

Actions:

Input:

e, the environment action

Output:

$Declare_{ij}$, for all $j \in \text{Others}_i$

$Undeclare_{ij}$, for all $j \in \text{Others}_i$

Discrete Transitions:

e:

Effect: Arbitrarily reset the input variables.

$Declare_{ij}$:

Precondition: $(\neg Threat_{ij}) \wedge Range\_Test_{ij} \wedge Altitude\_Test_{ij}$

$Undeclare_{ij}$:

Precondition: $Threat_{ij} \wedge \neg Range\_Test_{ij}$

Trajectories:

Input variables follow arbitrary trajectories.

Trajectories stop once any output action is enabled.

D Conflict Resolution Automaton R_i

Variables:

Input:

$Mode\_S_j \in \mathbb{N}$ for all $j \in$ Aircraft

$Equipment_j \in \{\text{None}, \text{Report}, \text{TCAS}\}$ for all $j \in$ Aircraft

$R_{ij} \in \mathbb{R}^+$, for all $j \in \text{Others}_i$

$\dot{R}_{ij} \in \mathbb{R}$, for all $j \in \text{Others}_i$

$h_{ij} \in \mathbb{R}^+$, for all $j \in$ Aircraft

$\dot{h}_{ij} \in \mathbb{R}$, for all $j \in$ Aircraft

Internal:

$Reversed_i \in$ `Bool`, initially False

$Crossing_i \in$ `Bool`, initially False

$Intent\_Sent_{ij} \in \mathrm{Dir}_{\perp}$, for all $j \in \mathrm{Others}_i$, initially $\perp$

Output:

$Sense_i \in \mathrm{Dir}_{\perp}$, initially $\perp$

$Threat_{ij} \in \mathtt{Bool}$, for all $j \in \mathrm{Others}_i$, initially False

$Strength_i \in$ Strengths, initially 0

Derived:

$SEP(\mathrm{dir}, \mathrm{str}) \in \mathbb{R}$ with dir $\in$ Dir, str $\in$ Strengths (see text)

$OK(\mathrm{dir}) \in \mathtt{Bool}$ with dir $\in$ Dir

$OK(\mathrm{dir}) := (SEP(\mathrm{dir}, 1500) \geq \mathtt{ALIM})$

$Indep\_Choice_{ij} \in$ Dir

if $(OK(\mathrm{Climb}) \wedge OK(\mathrm{Descend}) \wedge (h_{ii} \geq h_{ij}))$ then $Indep\_Choice_{ij} = \mathrm{Climb}$

if $(OK(\mathrm{Climb}) \wedge OK(\mathrm{Descend}) \wedge (h_{ii} < h_{ij}))$ then $Indep\_Choice_{ij} = \mathrm{Descend}$

if $(OK(\mathrm{Climb}) \wedge \neg OK(\mathrm{Descend}))$ then $Indep\_Choice_{ij} = \mathrm{Climb}$

if $(\neg OK(\mathrm{Climb}) \wedge OK(\mathrm{Descend}))$ then $Indep\_Choice_{ij} = \mathrm{Descend}$

if $(\neg OK(\mathrm{Climb}) \wedge \neg OK(\mathrm{Descend}) \wedge (SEP(\mathrm{Climb}, 1500) \geq SEP(\mathrm{Descend}, 1500)))$ then

$Indep\_Choice_{ij} = \mathrm{Climb}$

if $(\neg OK(\mathrm{Climb}) \wedge \neg OK(\mathrm{Descend}) \wedge (SEP(\mathrm{Climb}, 1500) < SEP(\mathrm{Descend}, 1500)))$ then

$Indep\_Choice_{ij} = \mathrm{Descend}$

$Strength\_Choice_{ij} \in \mathtt{Bool}$ (see text)

Actions:

Input:

e, the environment action

$Declare_{ij}$, for all $j \in$ Aircraft

$Undeclare_{ij}$, for all $j \in$ Aircraft

$Receive_{ij}(\mathrm{dir})$, for all $j \in \mathrm{Others}_i$, with dir $\in$ Dir

$Sample_i$

Output:

$Send_{ij}(\mathrm{dir})$, for all $j \in \mathrm{Others}_i$, with dir $\in$ Dir

Discrete Transitions:

e:

Effect: Arbitrarily reset the input variables.

$Declare_{ij}$:

Effect:

if $Equipment_i$ = TCAS then

if $\neg Threat_{ij}$ then

$Threat_{ij}$:= True

$Sense_i$:= $Indep\_Choice_{ij}$

if $(Sense_i = \text{Climb} \wedge h_{ii} < h_{ij} - 100ft) \vee (Sense_i = \text{Descend} \wedge h_{ii} > h_{ij} + 100ft)$

then $Crossing_i$:= True

Choose $Strength_i$ so that $Strength\_Choice_{ij}$ = True.

$Undeclare_{ij}$:

Effect:

if $Threat_{ij}$ then

$Sense_i := \perp$

$Threat_{ij}$:= False

$Reversed_i$:= False

$Crossing_i$:= False

$Intent\_Sent_{ij} := \perp$

$Receive_{ij}$(dir):

Effect:

if $Equipment_i$ = TCAS then

if $(Mode\_S_i > Mode\_S_j)$ then

if $\neg Threat_{ij}$ then $Threat_{ij}$:= True

$Sense_i$:= dir

if $(Sense_i = \text{Climb} \wedge h_{ii} < h_{ij} - 100ft) \vee (Sense_i = \text{Descend} \wedge h_{ii} > h_{ij} + 100ft)$

then $Crossing_i$:= True

Choose $Strength_i$ so that $Strength\_Choice_{ij}$ = True.

if $\neg Threat_{ij} \wedge (Mode\_S_i < Mode\_S_j)$ then

$Threat_{ij} :=$ True

if (dir = Climb $\wedge\ h_{ii} \geq h_{ij}$) then $Sense_i :=$ Climb

elseif (dir = Descend $\wedge\ h_{ii} \leq h_{ij}$) then $Sense_i :=$ Descend

else $Sense_i := Indep\_Choice_{ij}$

if ($Sense_i =$ Climb $\wedge\ h_{ii} < h_{ij} - 100ft$) $\vee$ ($Sense_i =$ Descend $\wedge\ h_{ii} > h_{ij} + 100ft$)

then $Crossing_i :=$ True

Choose $Strength_i$ so that $Strength\_Choice_{ij} =$ True.

$Sample_i$:

Effect:

if $Equipment_i =$ TCAS then

if $Threat_{ij}$ 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 ($Equipment_j = TCAS \wedge Mode\_S_i < Mode\_S_j \wedge \neg Reversed \wedge Crossing_i$) then

if ($Sense_i =$ Descend $\wedge\ OK$(Climb) $\wedge \neg OK$(Descend)) then

$Sense_i :=$ Climb

$Reversed_i :=$ True

if ($Sense_i =$ Climb $\wedge \neg OK$(Climb) $\wedge\ OK$(Descend)) then

$Sense_i :=$ Descend

$Reversed_i :=$ True

if ($Sense_i =$ Climb $\wedge\ h_{ii} > h_{ij} - 100ft$) $\vee$ ($Sense_i =$ Descend $\wedge\ h_{ii} < h_{ij} + 100ft$)

then $Crossing_i :=$ False

Choose $Strength_i$ so that $Strength\_Choice_{ij} =$ True.

$Send_{ij}$(dir):

Precondition:

($Equipment_i =$ TCAS $\wedge\ Equipment_j =$ TCAS)$\wedge$

([$Sense_i =$ Climb $\wedge\ Intent\_Sent_{ij} \neq$ Descend $\wedge$ dir = Descend] $\vee$

[$Sense_i =$ Descend $\wedge\ Intent\_Sent_{ij} \neq$ Climb $\wedge$ dir = Climb])

Effect:

$Intent\_Sent_{ij} :=$ dir

Trajectories:

Input variables follow arbitrary trajectories.

Internal and output variables remain constant.

Trajectories stop once any $Send_{ij}(\text{dir})$ action, for dir $\in$ Dir, is enabled.

E Communication Channel Automaton C_{ij}

Variables:

Internal:

$t_{C_{ij}} \in \mathbb{R}^+$, initially 0

$mset_{ij}$ = queue of $(\text{dir}, \underline{t}, \overline{t})$, with dir $\in$ Dir, $\underline{t}, \overline{t} \in \mathbb{R}^+$, initially $\emptyset$

Actions:

Input:

e, the environment action

$Send_{ij}(\text{dir})$, with dir $\in$ Dir

Output:

$Receive_{ji}(\text{dir})$, with dir $\in$ Dir

Discrete Transitions:

e:

Effect: None

$Send_{ij}(\text{dir})$:

Effect: Append $(\text{dir}, t_{C_{ij}} + \underline{d}_{ij}, t_{C_{ij}} + \overline{d}_{ij})$ to the tail of $mset_{ij}$.

$Receive_{ji}(\text{dir})$:

Precondition: $t_{C_{ij}} \in [\underline{t}, \overline{t}]$, where $(\text{dir}, \underline{t}, \overline{t})$ is the head of $mset_{ij}$

Effect: Pop $(\text{dir}, \underline{t}, \overline{t})$ from the head of $mset_{ij}$.

Trajectories:

$mset_{ij}$ remains constant

$\dot{t}_{C_{ij}} = 1$

Trajectories stop once $t_{C_{ij}} \geq \bar{t}$, where $(\mathrm{dir}, \underline{t}, \bar{t})$ is the head of $mset_{ij}$

F Pilot Automaton P_i

Variables:

Input:

$Sense_i \in \mathrm{Dir}_\perp$

$Strength_i \in \mathrm{Strengths}$

$\dot{h}_{ii} \in \mathbb{R}$

Internal:

$t_{P_i} \in \mathbb{R}^+$, initially 0

$Q\_Size_i \in \mathbb{N}$, initially 0

$Current\_Sense_i \in \mathrm{Dir}_\perp$, initially $\perp$

$Current\_Strength_i \in \mathrm{Strengths}$, initially 0

$Last\_Sense_i \in \mathrm{Dir}_\perp$, initially $\perp$

$Last\_Strength_i \in \mathrm{Strengths}$, initially 0

Adv_Q queue of $(i, sns, str, \underline{t}, \bar{t})$, initially empty,

with $i \in \mathbb{N}$, $sns \in \mathrm{Dir}_\perp$, $str \in \mathrm{Strengths}$, $\underline{t} \in \mathbb{R}^+$, $\bar{t} \in \mathbb{R}^+$

$Follow_i \in \texttt{Bool}$, initially True

Output:

$a_i \in \mathbb{R}^3$, initially $[0 \;\; 0 \;\; 0]^T$

Derived:

$\sigma \in \{-1, 0, 1\}$, such that

$$\sigma := \begin{cases} 1 & \text{if } Current\_Sense_i = \text{Climb} \\ 0 & \text{if } Current\_Sense_i = \perp \\ -1 & \text{if } Current\_Sense_i = \text{Descend} \end{cases}$$

Actions:

Input:

e, the environment action

Internal:

$New\_Advisory_i$

$Implement\_Advisory_i$

Discrete Transitions:

e:

Effect: arbitrarily reset the input variables

$New\_Advisory_i$:

Precondition:

$Last\_Sense_i \neq Sense_i \vee Last\_Strength_i \neq Strength_i$

Effect:

$Q\_Size_i := Q\_Size_i + 1$

Add $(Q\_Size_i, Sense_i, Strength_i, t_{P_i} + \underline{d}_i, t_{P_i} + \overline{d}_i)$ to Adv_Q

$Last\_Sense_i := Sense_i$

$Last\_Strength_i := Strength_i$

$Implement\_Advisory_i$:

Precondition:

There exists $(i, sns, str, \underline{t}, \overline{t}) \in Adv\_Q$ such that $t_{P_i} \in [\underline{t}, \overline{t}]$

Effect:

Remove $(i', sns', str', \underline{t}', \overline{t}')$ with $i' \leq i$ from Adv_Q

Replace $(i', sns', str', \underline{t}', \overline{t}')$ with $(i' - i, sns', str', \underline{t}', \overline{t}')$ in Adv_Q, for all $i' > i$

$Q\_Size_i := Q\_Size_i - i$

$Current\_Sense_i := sns$

$Current\_Strength_i := str$

$Follow_i :\in \{\text{True}, \text{False}\}$

Trajectories:

Input variables follow arbitrary trajectories.

All internal variables except t_{P_i} remain constant.

$\dot{t}_{P_i} = 1$

Output variables satisfy:

$$a_{xi}(t) \in [\underline{a}_{ix}, \overline{a}_{ix}]$$

$$a_{yi}(t) \in [\underline{a}_{iy}, \overline{a}_{iy}]$$

$$a_{zi}(t) \in \begin{cases} [\underline{a}_{zi}, \overline{a}_{zi}] & \text{if } \left[\mathit{Current\_Sense}_i = \perp \vee (\neg \mathit{Follow}_i) \vee \sigma \dot{h}_{ii} \geq \mathit{Current\_Strength}_i\right] \\ & \quad \wedge \dot{h}_{ii} \in (\underline{v}_{zi}, \overline{v}_{zi}) \\ [0, \overline{a}_{zi}] & \text{if } \left[\mathit{Current\_Sense}_i = \perp \vee (\neg \mathit{Follow}_i) \vee \sigma \dot{h}_{ii} \geq \mathit{Current\_Strength}_i\right] \\ & \quad \wedge \dot{h}_{ii} \leq \underline{v}_{zi} \\ [\underline{a}_{zi}, 0] & \text{if } \left[\mathit{Current\_Sense}_i = \perp \vee (\neg \mathit{Follow}_i) \vee \sigma \dot{h}_{ii} \geq \mathit{Current\_Strength}_i\right] \\ & \quad \wedge \dot{h}_{ii} \geq \overline{v}_{zi} \\ [\sigma a, \sigma a] & \text{if } \mathit{Current\_Sense}_i \neq \perp \wedge \mathit{Follow}_i \wedge \sigma \dot{h}_{ii} < \mathit{Current\_Strength}_i \end{cases}$$

Trajectories stop once the $\mathit{New\_Advisory}_i$ action is enabled.

Trajectories stop once there exists $(i, sns, str, \underline{t}, \overline{t}) \in \mathit{Adv\_Q}$ with $t_{P_i} \geq \overline{t}$.