Automatic verification of temporal properties of Air Traffic Management procedures using hybrid systems

M.D. Di Benedetto, A. D'Innocenzo, A. Petriccone
Department of Electrical Engineering and Computer Science, Center of Excellence DEWS. University of L'Aquila, Italy.

***Abstract*—In an Air Traffic Management context possible catastrophic events can take place because of an error of a single agent, e.g. pilot or flight controller, involved in a procedure such as landing sequence. In this paper we provide a mathematical framework based on hybrid systems theory for formal modeling of air traffic management applications. This framework can be used to model the dynamics of complex multi-agent systems in normal operative conditions, as well as in abnormal scenarios. The hybrid system framework allows description and detection of these errors and their effects on the evolution of procedures, as well as automatic verification of important properties such as safety and liveness. The *ASEP-In Trail Procedure* will be analyzed to illustrate the proposed methodology. We will describe the use of Uppaal, a tool that can be used to model ASEP-ITP and to verify automatically the properties of interest including safety.**

***Index Terms*—Air Traffic Management Procedures Modeling, Hybrid Systems, In Trail Procedure, Automatic Verification, Uppaal.**

I. Introduction

THE volume of air traffic in the oceanic airspace is increasing rapidly so that a major efficiency overhaul to manage air traffic flows is warranted to maintain normal operation. The new procedures needed to satisfy this requirement have to increase capacity without affecting safety. Safety has to be analyzed considering that operations are the result of interactions between many entities of various types and at multiple locations. Furthermore air traffic management systems are characterized by a mixed environment with human-controlled and computer-controlled subsystems the behaviors of which evolve following completely different logics that cannot be represented using the same class of mathematical models. This complexity can easily be modeled by means of agents in the context of hybrid system theory. Each decision taken by a single agent, either human operator or computer program, influences the actions of all other agents involved. An hazardous decision induced by a wrong situational awareness can then be yield a catastrophic event. When modeling this kind of multi-agent systems all the decision making processes of each agent and their interactions have to be taken into account to identify non-nominal situations and act accordingly to prevent them to evolve into accidents.

In this paper we propose to apply a methodology for formal reasoning based on hybrid systems theory that provides a powerful framework to develop multi-agents models. Using this methodology it is possible to link the changes of the physical systems behavior with the actions made by each agent. These actions can be either correct decisions or erroneous ones, mostly caused by situational awareness errors, taken by human operators, such as pilots and controllers. In this context each decision can represent an instantaneous change inside the continuous dynamics of an agent.

Using hybrid models it is possible to describe the behavior of single agent by means of discrete states. However, the computation costs related to the automatic verification of properties of hybrid systems are prohibitive for even small size systems. Reachability verification [10], [11], observability verification [12] and model checking [14] for hybrid systems have been intensively studied in the literature. To reduce the complexity of formal methods such as model checking, abstraction has been commonly used. Abstraction consists of building a simplified model with fewer states and/or simpler dynamics, which is equivalent to the original hybrid automaton with respect to the property of interest. System equivalence is usually defined using the notions of language equivalence and bisimulation [13]. In [15] we proposed a procedure to verify temporal properties of a hybrid automaton using a timed automaton abstraction. Timed automata can generally be abstracted into finite state systems [13], which makes automatic verification of properties decidable. Model checking tools for timed automata are available, e.g. *Uppaal* [16]. We apply the same plane of action in this paper: we use rectangular automata to model the ASEP-ITP procedure, translate it into an equivalent timed automaton, and use Uppaal to automatically verify temporal properties of the procedure. Using the methodology introduced in [18], it is possible to use such mathematical framework to also analyze observability properties of the system.

The paper is organized as follows. In Section 2, a particular procedure followed in air traffic control, the *In Trail Procedure (ITP)*, is presented to illustrate our hybrid system framework. In Section 3, the hybrid model of the airborne procedure is defined. In Section 4, we introduce timed automata and Uppaal, a formal verification tool for this kind of systems. In Section 5, we show how to translate a rectangular automaton into a timed automaton and use Uppaal to automatically verify temporal properties of the procedure. Section 6 provides some concluding remarks.

II. Description of the In Trail Procedure

The In Trail Procedure (ITP) is part of the *Airborne Separation Assistance Systems (ASAS)* area. ASAS embraces the goal of improving flight management by introducing a stronger interaction between pilots and controllers. The In Trail Procedure (ITP) here considered is envisioned as an *Airborne Separation (ASEP) Application* which is one of the four ASAS application categories. ASEP applications involve the transfer of responsibilities for the separation from the controller to the flight crew during the execution of the procedure. This can happen when the flight crew does have the most appropriate surveillance equipments (i.e. ADS-B and ASAS equipment) and is therefore able to monitor separation and act if necessary.

The ASEP-ITP [1], [2] described hereafter is a procedure that aims at improving flight efficiency along oceanic routes where procedural control is performed. The procedure provides a safe and practical method for air traffic controller to approve, and flight crew to conduct, climb and descent through different flight levels with less stringent applicability conditions than today's operations.

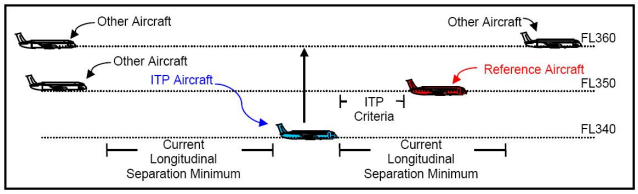

Fig. 1. Example of ITP geometry

A. ITP Criteria

The ASEP-ITP allows climb or descent through only one flight level for a maximum of 2000 feet in RVSM airspace (and 4000 feet in non-RVSM) and the ITP speed/distance criteria are designed so that under nominal conditions the proposed 5NM separation minimum is preserved throughout the ITP manoeuvre. The proposed ITP speed/distance criteria are the following:

- initiation ITP distance of no less than 10 NM and positive ground speed differential of no more that 20 kts, or
- ITP distance of no less than 15 NM and positive ground speed differential of no more that 30 kts.

The ITP encompasses a set of six vertical geometries: leading climb (as shown in Figure 1), leading descent, following climb, following descent, combined leading-following climb and combined leading-following descent. These geometries are designed on the basis of the relative position of the ITP aircraft and one or two reference aircraft.

The ITP aircraft must maintain a minimum 300 ft/min of climb or descent and constant cruise Mach number throughout the ITP manoeuvre. The reference aircraft must be non-manoeuvering and it is not expected to manoeuvre during the ITP. Given these conditions, it can be shown that a 4000 ft flight level change would result in a reduction in the initial distance of 4.5 NM assuming a positive ground speed differential of 20 kts. To ensure that the ITP separation minimum of 5NM will be guaranteed during the flight level change under these conditions, the initial distance between the aircraft must exceed 9.5 NM. So using 10 NM of initial distance the separation minimum is guaranteed. In the same way it could be proved that with positive ground speed differential of more than 20 but less than 30 kts, an initial distance of 15 NM ensures that ITP separation minimum is respected.

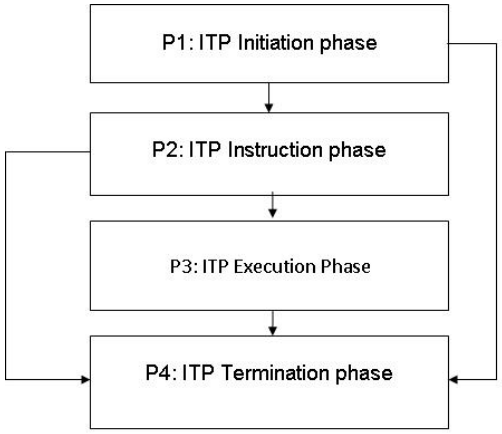

Fig. 2. ASEP-ITP phases diagram

A compact view of the ASEP-ITP phases is illustrated in Figure 2, and is now described.

B. ITP Initiation phase

In this phase, the flight crew determines that an ITP may be needed to obtain a higher or lower Flight Level. The crew identifies the Requested Flight Level and Intermediate Flight Levels, and verifies that their aircraft performance allows the ITP Flight Level change. The crew: checks for Same Direction Potentially Blocking Aircraft; verifies qualified ADS-B data from Potentially Blocking Aircraft; and assesses ITP Speed/Distance Criteria.

C. ITP Instruction Phase

ATC will always determine if a standard (i.e., non-ITP) Flight Level change can be approved. If a standard Flight Level change can be approved, this clearance would be issued instead of an ITP clearance. The controller checks for the presence of blocking aircraft at the Intervening Flight Levels and the availability of the Requested Flight Level. The controller checks that the Reference Aircraft are the only blocking aircraft and ensures that the request is consistent with the ITP. Once the controller has checked that the Positive Mach Differential is no greater than 0.04 Mach and that the Reference Aircraft are not expected to maneuver, the controller issues the ITP clearance. Once the flight crew accepts the clearance, they become responsible for performing the ITP maneuver.

D. ITP Execution Phase

The flight crew initiates the Flight Level change, maintaining a constant Mach number and a minimum 300 fpm (or greater if required by regulation) climb or descent rate throughout the ITP maneuver. The flight crew reports when the aircraft is established at the new Flight Level.

E. ITP Termination Phase

The ITP is completed when the ITP flight crew reports established at the new flight level. If the ITP aircraft cannot successfully complete the ITP once the climb or descent has been initiated, an abnormal termination occurs.

III. Hybrid Model of the ITP procedure

In this section the hybrid model of the ASEP-ITP is illustrated. The following description provides a general view of the hybrid systems. Thus, only the basic definitions are presented in order to facilitate the understanding of the ITP hybrid model proposed. A particular class of non-deterministic hybrid systems is represented by the *rectangular automata* and is the one that will be used in the hybrid model of ASEP-ITP. Considered the space $\mathbb{R}^n$ with variables $x_1, \cdots, x_n$, a rectangular set B of dimension n is the product of n intervals $B_i \subseteq \mathbb{R}$ of the real line, where each B_i is a bounded or unbounded interval.

Definition 1. ***(Rectangular Automaton [3])****: A rectangular automaton is a tuple* $H = (Q \times X, Q_0 \times X_0, U, Y, \varepsilon, E, \Psi, \eta, Inv, G, R)$ *where:*

- $Q = \{q_1, q_2, ..., q_N\}$ *is a set of* ***discrete states****.*
- $X \in \mathbb{R}^n$ *is a set of* ***continuous states****.*
- $Q_0 \subseteq Q$ *is a set of* ***initial discrete states****.*
- $X_0 \subseteq X$ *is a set of* ***initial continuous states****.*
- $U \subseteq \mathbb{R}^m$ *is a set of* ***continuous control input.***
- $Y \subseteq \mathbb{R}^p$ *is the set of* ***continuous observable output****.*
- $\{\varepsilon_q\}_{q \in Q}$ *associates to each discrete state* $q \in Q$ *the continuous time-invariant dynamics* $\varepsilon_q : \dot{x} = F_q(x)$ *with output* $y = g_q(x)$*.*
- $E \subseteq Q \times Q$ *is a* ***collection of edges****, where each edge* $e \in E$ *is a ordered pair of discrete states, the first component of which is known as source and is denoted by* $s(e)$*, while the second is the target and is denoted by* $t(e)$*.*
- Ψ *is the finite set of discrete output symbols* $\varepsilon, \psi_1, \psi_2, ..., \psi_r$ *where* ε *is the empty string that corresponds to unobservable output.*
- $\eta : E \to \Psi$ *is the output function, that associates to each edge a discrete output symbol.*
- $\{Inv_q\}_{q \in Q}$ *associates to each discrete state* $q \in Q$ *an invariant set* $Inv_q \subseteq X$*.*
- $\{G_e\}_{e \in E}$ *associates to each edge* $e \in E$ *a guard set* $G_e \subseteq Inv_{s(e)}$*.*
- $\{R_e\}_{e \in E}$ *associates to each edge* $e \in E$ *a reset map* $R_e : Inv_{s(e)} \to 2^{Inv_{t(e)}}$*, from* $Inv_{s(e)} \subset X$ *to the power set (i.e. the set of all the subsets) of* $Inv_{t(e)}$*.*

that satisfies the following constraints:

- *For every discrete state* $q \in Q$*, the set of initial continuous states* $X_0 \subseteq X$ *and the invariant set* $Inv_q \subseteq X$ *are rectangular sets.*
- *For every discrete state* $q \in Q$*, there is a rectangular set* B^q *such that the continuous time invariant dynamics* $\varepsilon_q : \dot{x} = F_q(x) \in B^q$ *for all* $x \in \mathbb{R}^n$*.*
- *For every edge* $e \in E$*, the set* $Guard_e$ *is a rectangular set, and there is a rectangular set* B^e *and a subset* $J^e \subseteq \{1, \cdots, n\}$ *such that for all* $x \in \mathbb{R}^n$ *the reset map is* $R_e = \{(x_1, \cdots, x_n^{'}) \in \mathbb{R}^n |$ *for all* $1 \leq i \leq n$*, if* $i \in J^e$ *then* $x_i^{'} \in B_i^e$ *else* $x_i^{'} = x_i\}$*.*

Therefore, in a rectangular automaton, the derivative of each variable stays between two fixed bounds, which can be different in different discrete states. Then in each discrete state $q \in Q$ the continuous dynamics can be defined as $\dot{x}_i \in B_i^q \subseteq B^q$ for all $1 \leq i \leq n$. With each discrete jump across an edge e, the value of the variable x_i either does not change if $i \notin J^e$, or resets non-deterministically to a new value within some fixed constant interval $B_i^e \subseteq B^e$ if $i \in J^e$.

The hybrid model proposed below embeds a set Σ of discrete input signals, and each edge $e \in E$ is associated to a symbol $\sigma \in \Sigma$ that triggers the discrete transition between the states linked by e. These inputs can be considered as discrete disturbance or control inputs which model the communication among the agents.

The ASEP-ITP can be decomposed in various subsystems representing the agents involved in the procedure, each with hybrid dynamics modeling its specific operations. It should be remarked that to exploit the descriptive power of hybrid system each agent must be considered by itself and subsequently the effects of their actions on the dynamics of other agents can be considered composing such models. The agents considered are:

- Air crew flying of ITP aircraft
- Oceanic controller

The approach used for selecting the agents does not provide the modeling of the reference aircraft as an agent. The main reason is that the flight crew of the reference aircraft does not have the awareness of existence of an ITP manoeuvre in which it is involved. In fact, there is no communication between the controller or the flight crew of the ITP aircraft and the flight crew of the reference aircraft. Furthermore any hazardous actions of the reference aircraft can be considered inside the hybrid dynamics of other agents.

The model proposed considers the simplest case of ASEP-ITP execution where the ITP aircraft requests a climb through one flight level, with only one leading reference aircraft involved and without other blocking aircraft. Furthermore, no wind is assumed. The continuous dynamics used in this approach are intentionally simplified. In fact due to the configuration of the traffic flows in the oceanic airspace (i.e organized parallel tracks system) it is possible to focus on longitudinal and vertical dynamics without considering the lateral dynamics.

Before defining the hybrid models, the following variables are introduced:

1) z_i initial flight level of the aircraft
2) z_f requested flight level of the ITP aircraft
3) x_0 the initial longitudinal position of the ITP aircraft

4) x_{r0} longitudinal position of the reference aircraft at the moment of the criteria evaluation
5) v_{G0} the ground speed of the ITP aircraft at the moment of the criteria evaluation
6) v_{Gr0} the ground speed of the reference aircraft at the moment of the criteria evaluation
7) M_i assigned Mach number for the ITP aircraft
8) a speed of sound, assumed as a constant value
9) v_{x0} the initial longitudinal airspeed of the ITP aircraft defined as $v_{x0} = M_i a$
10) $v_{z,max}$ maximal vertical speed of the ITP aircraft

Furthermore the following interesting areas of the airspace can be identified:

1) A safe region in which the ITP aircraft performing the ITP manoeuvre respects the ITP minimum distance separation. The safe zone is defined as $\Omega_S = \{(x, z) : x \in [-\infty, x_r - 5], z \in (z_i, z_f)\}$.
2) Thus, an unsafe zone can be defined as follows: $\Omega_U = \{(x, z) : x \in [x_r - 5, +\infty], z \in (z_i, z_f)\}$.

The agent $\mathcal{H}_p$ *Pilot Flying of ITP Aircraft* can be described using a model based on Definition 1. The following are the objects of the system:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}\}$ is the set of discrete states each associated with a node inside the graph depicted in Figure 3;
- $X = \{(x, z, \dot{x}, \dot{z}) : x \in \mathbb{R}_0^+, z \in \mathbb{R}^+, \dot{x} \in [v_{x0} - \Delta K, v_{x0} + \Delta K], \dot{z} \in [0, v_{z,max}]\}$ is the set of the continuous state values where x represents the longitudinal position of the aircraft expressed in nautical miles, z the altitude of the aircraft expressed in hundred of feet (i.e. flight level inside the International Standard Atmosphere), $\dot{x}$ is the airspeed along the longitudinal position expressed in Knots and $\dot{z}$ is the airspeed along the vertical position (i.e., the vertical speed) expressed in feet per minute;
- The initial discrete state is q_1, the initial continuous state is given by $(x_0, z_i, v_{x0}, 0)$;
- The continuous dynamics are the followings:
  - $F_{q_1} = \{\dot{x} \in [v_{x0} - \Delta K_c, v_{x0} + \Delta K_c], \dot{z} = 0, \ddot{x} \in [-0.001, 0.001], \ddot{z} = 0\}$ where $\Delta K_c = 0.001a$
  - $F_{q_7} = \{\dot{x} \in [v_{x0} - \Delta K_m, v_{x0} + \Delta K_m], \dot{z} \in [300, v_{z,max}], \ddot{x} \in [-0.005, 0.005], \ddot{z} \in [-0.001, 0.001]\}$ where $\Delta K_m = 0.01a$
  - $F_{q_8} = \{\dot{x} \in [v_{x0} - \Delta K_m, v_{x0} + \Delta K_m], \dot{z} \in [300, v_{z,max}], \ddot{x} \in [-0.005, 0.005], \ddot{z} \in [-0.001, 0.001]\}$ where $\Delta K_m = 0.01a$
  - $F_{q_9} = \{\dot{x} \in [v_{x0} - \Delta Ma, v_{x0} + \Delta Ma], \dot{z} \in [50, v_{z,max}], \ddot{x} \in [-0.005, 0.005], \ddot{z} \in [-0.001, 0.001]\}$ where $\Delta M = 0.04$
  - $F_{q_{10}} = \{\dot{x} \in [(M_i - \Delta M)a, (M_i + \Delta M)a], \dot{z} = 0, \ddot{x} \in [-0.001, 0.001], \ddot{z} = 0\}$ where $\Delta M = 0.04$
  - $F_{q_{12}} = \{\dot{x} \in [v_{x0} - \Delta K_m, v_{x0} + \Delta K_m], \dot{z} = 0, \ddot{x} \in [-0.001, 0.001], \ddot{z} = 0\}$ where $\Delta K_m = 0.01a$
  - $F_{q_{13}} = \{\dot{x} \in [v_{x0} - \Delta Ma, v_{x0} + \Delta Ma], \dot{z} \in [300, v_{z,max}], \ddot{x} \in [-0.01, 0.01], \ddot{z} \in [-0.005, 0.005]\}$ where $\Delta M = 0.04$
  - $F_{q_i} = F_{q_1}$ for $i = 2, 3, 4, 5, 6, 11$
- $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_8, \sigma_9\} \bigcup \{\varepsilon\}$ is the set of discrete inputs, where σ_1 represents the verification of ITP pre-conditions, σ_2 represents the reassessment failed after a clearance reception, σ_3 represents the ITP criteria are not verified, σ_4 means the ITP criteria verified, σ_5 represents the clearance denied, σ_6 means the clearance issued, σ_7 means detection of an abnormal event, σ_8 represents a situational awareness error, σ_9 is an ASAS conflict detection communication, ε is the verification of an internal event;
- $\Psi = \{\psi_1, \psi_2, \psi_3, \psi_4, \psi_5, \psi_6, \psi_7\} \cup \{\varepsilon\}$ is the set of discrete outputs, where ψ_1 means the clearance rejected by the crew, ψ_2 represents the clearance request, ψ_3 represents the setting of flight parameters for the climb, ψ_4 means the abnormal termination communication by the crew to the controller, ψ_5 means the report established at the new flight level, ψ_6 is the reversion to cruise operation, ψ_7 is the setting of flight parameters to solve an ASAS conflict detection, ε is associated with an unobservable transition;
- $E \subseteq Q \times Q$ is the set of transitions given by the graph depicted in Figure 3. A label $\sigma \in \Sigma$ is associated to each edge as shown in Figure 3;
- $\eta : E \to \Psi$ the discrete output function defined by the graph depicted in Figure 3;
- The domains of the discrete states are the following:
  - $Inv_{q_1} = \{(x, z, \dot{x}, \dot{z}) : x \in \mathbb{R}_0^+, z = z_i, \dot{x} \in [v_{x0} - \Delta K_c, v_{x0} + \Delta K_c], \dot{z} = 0\}$ where $\Delta K_c = 0.001a$
  - $Inv_{q_7} = \{(x, z, \dot{x}, \dot{z}) : \{(x, z) \in \Omega_S\} \bigcup \{x \in \mathbb{R}_0^+ \cap z = z_i\}, \dot{x} \in [v_{x0} - \Delta K_m, v_{x0} + \Delta K_m], \dot{z} \in [300, v_{z,max}]\}$ where $\Delta K_m = 0.01a$
  - $Inv_{q_8} = \{(x, z, \dot{x}, \dot{z}) : (x, z) \in \Omega_S \cup \Omega_U \bigcup \{x \in \mathbb{R}_0^+ \cap z = z_i\}, \dot{x} \in [v_{x0} - \Delta K_m, v_{x0} + \Delta K_m], \dot{z} \in [300, v_{z,max}]\}$ where $\Delta K_m = 0.01a$
  - $Inv_{q_9} = \{(x, z, \dot{x}, \dot{z}) : (x, z) \in \Omega_S \cup \Omega_U, \dot{x} \in [v_{x0} - \Delta Ma, v_{x0} + \Delta Ma], \dot{z} \in [50, v_{z,max}]\}$ where $\Delta M = 0.04$
  - $Inv_{q_{10}} = \{(x, z, \dot{x}, \dot{z}) : x \in \mathbb{R}_0^+, z = z_f, \dot{x} \in [(M_i - \Delta M)a, (M_i + \Delta M)a], \dot{z} = 0\}$ where $\Delta M = 0.04$
  - $Inv_{q_{12}} = \{(x, z, \dot{x}, \dot{z}) : x \in \mathbb{R}_0^+, z = z_f, \dot{x} \in [v_{x0} - \Delta K_m, v_{x0} + \Delta K_m], \dot{z} = 0\}$ where $\Delta K_m = 0.01a$
  - $Inv_{q_{13}} = \{(x, z, \dot{x}, \dot{z}) : (x, z) \in \Omega_S, \dot{x} \in [v_{x0} - \Delta Ma, v_{x0} + \Delta Ma], \dot{z} \in [300, v_{z,max}]\}$ where $\Delta M = 0.04$
  - $Inv_{q_i} = Inv_{q_1}$ for $i = 2, 3, 4, 5, 6, 11$
- The guards are the empty set for all the discrete transitions except for:
  - $G(q_7, q_9) = \{\dot{x} > v_{x0} + \Delta K_m \cup \dot{x} < v_{x0} - \Delta K_m \cup \dot{z} < 300\}$ where $\Delta K_m = 0.01a$
  - $G(q_8, q_9) = G(q_7, q_9)$

- $G(q_7, q_{12}) = \{z = z_f\}$
- $G(q_8, q_{12}) = G(q_9, q_{12}) = G(q_{13}, q_{12}) = G(q_7, q_{12})$
- $G(q_{12}, q_{10}) = \{\dot{x} > v_{x0} + \Delta K_m \cup \dot{x} < v_{x0} - \Delta K_m\}$ where $\Delta K_m = 0.01a$

- The reset function is always the identity function excepted for:
  - $R(q_8, q_{13}) = \{x_{q_{13}} = x_{q_8}, z_{q_{13}} = z_{q_8}, \dot{x} \in [v_{x0} - \Delta Ma, v_{x0} + \Delta Ma], \dot{z} \in [300, v_{z,max}]\}$ where $\Delta M = 0.04$
  - $R(q_9, q_{13}) = R(q_8, q_{13})$
  - $R(q_7, q_{12}) = \{x_{q_{12}} = x_{q_7}, z_{q_{12}} = z_{q_7}, \dot{x}_{q12} = \dot{x}_{q7}, \dot{z}_{q12} = 0\}$
  - $R(q_8, q_{12}) = R(q_9, q_{12}) = R(q_{13}, q_{12}) = R(q_7, q_{12})$
  - $R(q_7, q_{11}) = \{x_{q_{11}} = x_{q_7}, z_{q_{11}} = z_i, \dot{x}_{q11} = \dot{x}_{q7}, \dot{z}_{q11} = 0\}$
  - $R(q_8, q_{11}) = R(q_9, q_{11}) = R(q_{13}, q_{11}) = R(q_7, q_{11})$

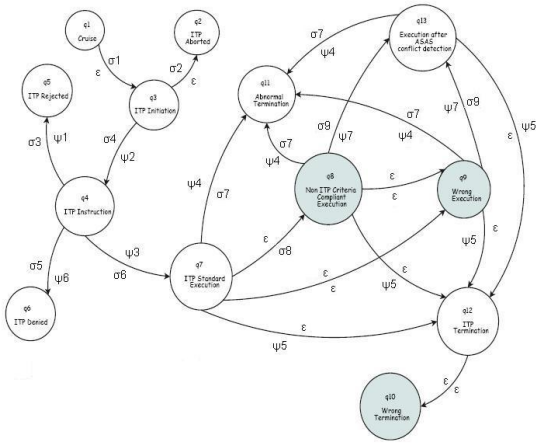

Fig. 3. Direct graph of pilot flying of ITP aircraft agent.

IV. UPPAAL AND TIMED AUTOMATA

Uppaal [16] is a toolbox for verification of real-time systems jointly developed by Uppsala University and Aalborg University. The tool is designed to verify systems that can be modeled as networks of timed automata extended with integer variables, structured data types, and channel synchronization. This model-checker is based on the theory of timed automata and its modeling language offers additional features such as bounded integer variables and urgency. A timed automaton is essentially a finite automaton (that is a graph containing a finite set of nodes or locations and a finite set of labeled edges) extended with real-valued variables. Such an automaton may be considered as an abstract model of a timed system. The query language of Uppaal, used to specify properties to be checked, is a subset of CTL (Computation Tree Logic, [17]).

A. The Modeling Language

A timed automaton is a finite-state machine extended with clock variables, in Uppaal a system is modeled as a network of several such timed automata in parallel. The model is further extended with bounded discrete variables that are part of the state. These variables are used as in programming languages: they are read, written, and are subject to common arithmetic operations.

A state of the system is defined by the locations of all automata, the clock constraints, and the values of the discrete variables. Every automaton may fire an edge separately or synchronize with another automaton, which leads to a new state. We give the basic definitions of the syntax and semantics for timed automata: C is a set of clocks and B(C) is the set of conjunctions over simple conditions of the form $x \bowtie c$ or $x - y \bowtie c$, where $x, y \in C$, $c \in \mathbf{N}$ and $\bowtie \in \{<, \leq, =, >, \geq\}$.

Definition 2. ***(Timed Automaton)****: A timed automaton is a tuple (L, l_0, C, A, E, I) where L is a set of locations, $l_0 \in L$ is the initial location, C is the set of clocks, A is a set of actions and co-actions, $E \subseteq L \times A \times B(C) \times 2^C \times L$ is a set of edges between locations with an action, a guard and a set of clocks to be reset, $I : L \rightarrow B(C)$ assigns invariants to locations.*

We now define the semantics of a timed automaton. A clock valuation is a function $u : C \rightarrow \boldsymbol{R}_{\geq 0}$ from the set of clocks to the non-negative reals. Let $\boldsymbol{R}^C$ be the set of all clock valuations and $u_0(x) = 0$ for all $x \in C$. We will abuse the notation by considering guards and invariants as sets of clock valuations, writing $u \in I(l)$ to mean that u satisfies $I(l)$.

Definition 3. ***(Semantics of Timed Automaton)****: Let (L, l_0, C, A, E, I) be a timed automaton. The semantics is defined as a labeled transition system $\langle S, s_0, \rightarrow \rangle$ where $S \subseteq L \times \mathbf{R}^C$ is the set of states, $s_0 = (l_0, u_0)$ is the initial state, $\rightarrow \subseteq S \times \{\mathbf{R}_{\geq 0} \cup A\} \times S$ is the transition relation such that:*

- $(l, u) \rightarrow^d (l, u+d)$ *if* $\forall d' : 0 \leq d' \leq d \Rightarrow u + d' \in I(l)$;
- $(l, u) \rightarrow^a (l', u')$ *if there exists* $e = (l, a, g, r, l') \in E$ *s.t.* $u \in g, u' = [r \mapsto 0]u, u' \in I(l)$;

where for $d \in \mathbf{R}_{\geq 0}$, $u + d$ maps each clock x in C to the value $u(x) + d$, and $[r \mapsto 0]u$ denotes the clock valuation which maps each clock in r to 0 and agrees with u over $C \backslash r$.

The Uppaal modeling language extends timed automata with the following additional features:

- *Templates*: automata are defined with a set of parameters that can be of any type;
- *Binary synchronization*: channels are declared as `chan c`. An edge labeled with `c!` synchronizes with another labeled `c?` ;
- *Broadcast channels*: are declared as `broadcast chan c`. In a broadcast synchronization one sender `c!` can synchronize with an arbitrary number of receivers `c?`; any receiver than can synchronize in the current state must do so. If there are no receivers, then the sender can still execute the `c!` action, i.e. broadcast sending is never blocking;

Expressions in Uppaal range over clocks and integer variables. Expressions are used with the following labels:

- *Guard*: is a particular expression satisfying the following conditions: it is side-effect free;it evaluates to a boolean; only clocks, integer variables,and constants are referenced(or arrays of these types); clocks and clock differences are only compared to integer expressions; guards over clocks are essentially conjunctions.
- *Synchronization*: a synchronization label is either on the form *Expression!* or *Expression?* or is an empty label. The expression must be side-effect free, evaluate to a channel, and only refer to integers, constants and channels;
- *Assignment*: an assignment label is a comma separated list of expressions with a side-effect; expressions must only refer to clocks, integer variables, and constants and only assign integer values to clocks;
- *Invariant*: an invariant is an expression that satisfies the following conditions: is side-effect free; only clock, integer variables, and constants are referenced; it is a conjunction of conditions of the form $x < e$ or $x <= e$, where x is a clock reference and e evaluates to an integer.

B. The Query Language

The main purpose of a model checker is verify the model w.r.t. a requirement specification. Like the model, the requirement specification must be expressed in a formally well-defined and machine readable language. Several such logics exist in the scientific literature, and Uppaal uses a simplified version of CTL [17].

Like in CTL, the query language consists of path formulae and state formulae. State formulae describe individual states, whereas path formulae quantify over paths or traces of the model. Path formulae can be classified into reachability, safety and liveness. A state formula is an expression that can be evaluated for a state without looking at the behavior of the model. For instance, this could be a simple expression, like $i == 7$, that is true in a state whenever i equals 7. The syntax of state formulae is a superset of that of guards, i.e., a state formula is a side-effect free expression, but in contrast to guards, the use of disjunctions is not restricted. It is also possible to test whether a particular process is in a given location using an expression on the form P.q_1, where P is a process and q_1 is a location.

- **Reachability Properties**: Reachability properties are the simplest class of properties. They ask whether a given state formula, φ, *possibly* can be satisfied by any reachable state. *Does a path exist, starting from the initial state, such that φ is eventually satisfied?* Reachability properties are often used while designing a model to perform sanity checks. We express that some state satisfying φ should be reachable using the path formula $E \diamond \varphi$, and in Uppaal we write this property using the syntax $E <> \varphi$.
- **Safety Properties**: Safety properties are of the form *something bad will never happen*. A variation of this property is that *something will possibly never happen*. For instance, when playing a game, a safe state is one in which we can still win the game, hence we will possibly not loose. In Uppaal these properties are formulated positively, e.g. *something good is invariantly true*. Let φ be a state formula, we express that φ should be true in all reachable states with the path formulae $A[]\ \varphi$, whereas $E[]\ \varphi$ means that there should exist a maximal[1] path such that φ is always true.
- **Liveness Properties**: Liveness properties are of the form *something will eventually happen*, e.g. in a model of a communication protocol we may require that any message that has been sent should eventually be received. Liveness is expressed with the path formula $A <> \varphi$, meaning that φ is eventually satisfied. The most useful form is the *leads to or response property*, which can be expressed as $\varphi \rightarrow \psi$, namely whenever φ is satisfied, then eventually ψ will be satisfied; in the communication protocol example, whenever a message is sent then eventually it will be received.

V. ASEP-ITP Model in Uppaal

To this point we can represent the hybrid model of the ASEP-ITP as a timed automaton, so that we can verify of the properties.

A. From Rectangular Automata to Timed Automaton

It is possible to translate a rectangular automaton into a timed automaton, obtaining an equivalent system that preserves all the temporal properties of the original system. This is due to the fact that both rectangular automata and timed automata admit a finite bisimulation [13]. This implies that, given any rectangular automaton, it is possible to construct a bisimilar (and thus equivalent) timed automaton. The main issue is translating the dynamics and guards of the rectangular automaton into clocks and guards of the timed automaton. However, since rectangular automata are characterized by very simple dynamics, such computation can be performed in a closed form, as will be illustrated in the next section, in the definition of the timed automaton that models the Pilot Flying of ITP Aircraft.

Remark 1. *For instance, we stress that the translation from rectangular automata to timed automata also preserves observability properties, that is the rectangular automaton is observable if and only if the timed automaton is observable. The implication is in fact symmetric, because of the equivalence of the two systems.*

B. Pilot flying of ITP aircraft Agent

The hybrid system of the pilot can be defined like a timed automaton in the following way:

$$Pilot = (L, l_0, C, A, E, I)$$

where:

- L is the set of following locations: *Cruise*, *ITP Aborted*, *ITP Initiation*, *Wait*, *ITP Instruction*, *ITP Rejected*, *ITP*

[1] A maximal path is a path that is either infinite or where the last state has no outgoing transitions.

Denied, *ITP Standard Execution*, *Abnormal Termination*, *NITPC CExe*(Non ITP Criteria compliant execution), *ITP Termination*, *Wrong Execution*, *Exe ASAS Conf*(Execution after ASAS conflict detection), *Wrong Termination*.

- $l_0 = \{Cruise\}$ is the initial location;
- $C = \{t_1, t_2\}$ is the set of clocks and it is used in the guard of some locations. In particular we have that:
  - $t_1 = \frac{z_{fin} - z_{in}}{\dot{z}_{min}} = \frac{40000ft - 36000ft}{300fpm} = 13$ minutes
  - $t_2 = \frac{z_{fin} - z_{in}}{\dot{z}_{max}} = \frac{40000ft - 36000ft}{1000fpm} = 4$ minutes

  supposing that the aircrafts carry out a change of the level of flight passing from 36000 ft to 40000 ft with a minimal speed of 300 fpm and a maximum of 1000 fpm;
- A is a set of actions and co-actions, defined in the Figure 4;
- E is a set of edges between locations, defined in Figure 4;
- $I = \{x \leq 10\}$ assigns invariants to locations, in particular to ITP Instruction.

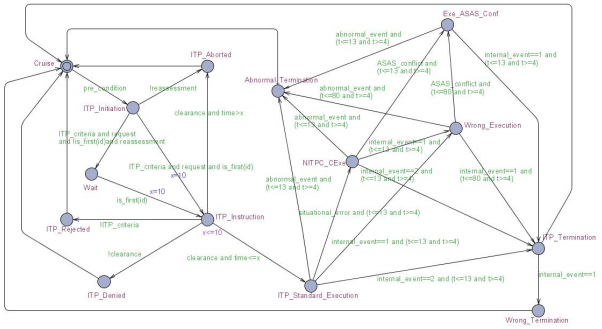

Fig. 4. Model of the pilot under Uppaal

C. Scenario: Wrong Execution of ITP with ASAS conflict detection and Wrong Termination

We apply our verification procedure to a specific scenario of the ITP, that models an error in the procedure. Initially, the flight is in its cruise phase and the pilot consider the will to execute an ASEP-ITP. If the pre-conditions are verified we enter in the ITP Initiation phase. In this phase the pilot checks the ITP Criteria. The ITP speed distance criteria are checked using the data v_{G0}, v_{Gr0} and $x_0 - x_{r0}$. Basing on his situational awareness the crew verifies that all the criteria are satisfied. Thus he communicates the request of an ITP clearance to the controller. The controller also verifies the ITP Criteria and gives to the aircrew the clearance to execute the manoeuvre. The pilot starts the manoeuvre respecting the performances envelope (Standard ITP Execution) but during the execution of the manoeuvre the pilot does not keep the Mach number constant, exceeding the bound of 0.01 Mach error (Wrong Execution). The ASAS technical system is functioning properly so it detects the possibility of a conflict and generates an alert. The technical system also assists the pilot providing information on how to resume the standard execution. In this phase (Execution after ASAS conflict detection) the pilot can temporarily change the vertical speed or the Mach number to solve the possible conflict. The pilot must revert to the initial Mach number after the conflict alarm is solved. We assume that the pilot forgets to revert the speed of the aircraft to the initial Mach number. When the aircraft stabilizes in the requested flight level (ITP termination) the pilot communicates to the ATC the establishment at that flight level. Without having any information the flight is in an hazardous situation. In fact, the immediate verification of a guard condition which checks if the Mach has changed, brings to a (Wrong Execution). The graph of Figure 5 enhances this path.

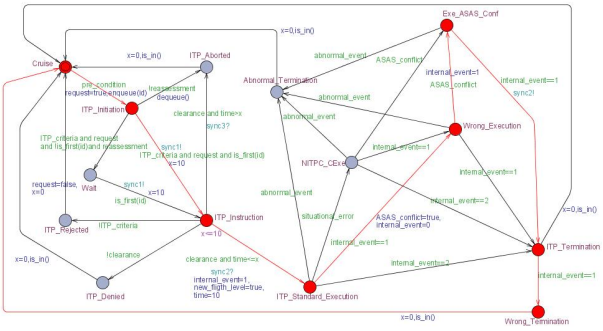

Fig. 5. Wrong Execution of ITP with ASAS conflict detection

Once described this scenario, we can analyze the Reachability properties of the pilot. In order to perform automatic verification with Uppaal, we need to formalize these properties using the appropriate syntax:

- $E <>$ *ASEP_Pilot.Cruise*: does there exist a path starting from the initial state, such that Cruise is eventually satisfied along that path? **Yes**, the property is satisfied;
- $E[\,]$ *ASEP_Pilot.Cruise*: does there exist a maximal path such that Cruise is always satisfied? **Yes**, the property is satisfied;
- $E <>$ *ASEP_Pilot.ITPInstruction*: does there exist a path starting at the initial state, such that ITP Instruction is eventually satisfied along that path? **Yes**, the property is satisfied;
- $E[\,]$ *ASEP_Pilot.ITPInstruction*: does there exist a maximal path such that ITP Instruction is always satisfied? **No**, the property is not satisfied.

Remark 2. *The property* $E <>$ ASEP_Pilot.Cruise *means that the location Cruise can be reached, namely such that the node Cruise will eventually be touched by a path.*

Remark 3. *The property* $E[\,]$ ASEP_Pilot.Cruise *means that the system can reach a safe state without passing through dangerous situations. This type of property leads to determine whether a safe state can be reached without passing through unsafe states.*

The same reasoning applies to the other properties for ITP Instruction.

VI. Conclusion

We presented a hybrid system framework for safety modeling in air traffic management. The need to develop new

sophisticated modeling methodologies originates from new challenges in safety and from the increasing inherent complexity in the airborne procedures. A specific procedure, the ASEP-ITP, was investigated to show how this framework can be used to represent a complex multi-agent application in which a wide set of possible abnormal scenarios may occur. Possible catastrophic events can take place due to e.g. unnoticed misunderstanding between agents involved. We demonstrated that our hybrid system framework can be used to describe and detect abnormal conditions and to understand and quantify their effects on the evolution of the procedure. In particular, we used Uppaal, a formal verification tool for timed automata, to verify important properties of the procedure such as reachability, which are essential in determining its safety. The use of appropriate abstractions extends the applicability of formal verification to realistic examples.

ACKNOWLEDGMENTS

This work was partially supported by European Commission under STREP project n.TREN/07/FP6AE/S07.71574/037180 iFLY. The authors are grateful to Pascal Lezaud and Thierry Miquel for hosting Marco Colageo and Antonio Di Francesco at ENAC for a stage, during which the description of the ASEP-ITP procedure was developed.

REFERENCES

[1] "D6.1b Qualitative Risk Assessment for ASEP-ITP", ASSTAR Projects, 01 February 2007 v.1.0

[2] "In-Trail Procedure in Procedural Airspace (ATSA-ITP) Application Description", Package I Requirements Focus Group, 21 June 2007 v8.0

[3] R. Alur, T.A. Henzinger, G. Laferriere and G.J. Pappas, "Discrete Abstractions of Hybrid Systems", Proceedings of the IEEE, vol. 88, NO. 7, July 2000

[4] A. Dijkstra, "Resilience Engineering and Safety Management Systems in Aviation", KLM Royal Dutch Airlines / TU Delft

[5] E. Hollnagel, O. Goteman, "The Functional Resonance Accident Model"

[6] M.D. Di Benedetto, S. Di Gennaro, A. D'Innocenzo, "Error Detection Within a Specific Time Horizon", public delivarable D7.4, project IST-2001-32460 HYBRIDGE, January 26,2005, http://www.nlr.nl/public/hosted-sites/hybridge

[7] M. D. Di Benedetto, S. Di Gennaro, A. DInnocenzo, "Situation Awareness Error Detection" , Public Deliverable D7.3, Project IST200132460 HYBRIDGE, August 18, 2004, http://www.nlr.nl/public/hostedsites/hybridge.

[8] Johan Bengtsson - Wang Yi - Uppsala University, *Timed Automata: Semantics, Algorithms and Tools*, Uppsala University, 2003.

[9] Gerd Behrmann - Alexandre David - Kim G. Larsen, *A Tutorial on Uppaal*, Department of Computer Science, Aalborg University, Denmark, 2004.

[10] A. Girard. Reachability of uncertain linear systems using zonotopes. In M. Morari and L. Thiele, editors, Hybrid Systems: Computation and Control, volume 3414 of Lecture Notes in Computer Science, pages 291305. Springer Verlag, 2005.

[11] Zhi Han and B. H. Krogh. Reachability analysis of largescale affine systems using lowdimensional polytopes. In J. Hespanha and A. Tiwari, editors, Hybrid Systems: Computation and Control, volume 3927 of Lecture Notes in Computer Science, pages 287301. Springer Verlag, 2006.

[12] A. DInnocenzo, M. D. Di Benedetto, and S. Di Gennaro. Observability of hybrid automata by abstraction. In J. Hespanha and A. Tiwari, editors, Hybrid Systems: Computation and Control, volume 3927 of Lecture Notes in Computer Science, pages 169183. Springer Verlag, 2006.

[13] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. Proccedings of the IEEE, 88(2):971984, July 2000.

[14] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. IEEE Transactions on Software Engineering, 22:181201, 1996.

[15] A. DInnocenzo, A. A. Julius, G. J. Pappas, M. D. Di Benedetto, and S. Di Gennaro. Verification of temporal properties on hybrid automata by simulation relations. In Proceedings of the 46th IEEE Conference on Decision and Control. New Orleans, Louisiana, USA., 1214 December 2007.

[16] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. International Journal on Software Tools for Technology Transfer, 1(1):134152, December 1997.

[17] E.M. Clarke, O. Grumberg, and D.A. Peled. Model Checking. The MIT Press, Cambridge, Massachusetts, 2002.

[18] Marco Colageo, Antonio Di Francesco. ICRAT 2008 - 3rd International Conference on Research in Air Transportation, Fairfax, Virginia, USA. June 01-04 2008.