Proceedings of the
46th IEEE Conference on Decision and Control
New Orleans, LA, USA, Dec. 12-14, 2007

Verification of temporal properties on hybrid automata by simulation relations

A. D'Innocenzo[♭], A.A. Julius[♯], G.J. Pappas[♯], M.D. Di Benedetto[♭], S. Di Gennaro[♭]

Abstract— **Model checking can be used to verify temporal properties of hybrid automata. However, model checking is not decidable in general. We overcome this difficulty by considering a durational graph abstraction, for which model checking is decidable. The contribution of this paper is to show that, given a hybrid automaton and the durational graph abstraction, there exists a *simulation relation* between the two systems. This approach allows checking properties such as safety, but also timing properties of hybrid automata. We apply our framework to a relevant case study in the context of air traffic management (*ATM*). For an extended version of this paper refer to [15].**

I. Introduction

The role of embedded systems in safety critical systems has constantly increased in the past few years creating the need of stronger verification methods than the ones customarily used by the industry today. Formal verification has recorded a number of successes in aiding the design of hybrid/embedded systems [12]. However, as the complexity of the to-be-verified systems increases, the computation costs related to the verification procedures become prohibitive. Reachability verification [7], [16], [18], observability verification [8], [13], [14], [23] and model checking [5], [19] for hybrid systems are intensely studied in the literature. A model checking tool for hybrid systems is *HyTech* [19]. To abate the complexity of model checking, abstraction has been commonly used. Abstraction consists of building a simplified model with fewer states and/or simpler dynamics that is equivalent to the original hybrid automaton with respect to the property of interest, i.e., if the property holds in the abstraction, it also holds in the original system and vice versa. System equivalence is usually defined by the notions of language equivalence and bisimulation [4], [25].

A procedure to translate a hybrid automaton into a rectangular automaton was proposed in [26]. In a recent paper [14] we proposed a procedure to check observability of a hybrid automaton using a timed automaton abstraction. The abstract model belongs to a subclass of timed automata, which is called *durational graph*. Timed automata are a special class of hybrid automata [3], [2], in which the continuous variables are *clocks*, increasing with a constant slope. Resets are restricted to clock resets to 0. Timed automata can generally be abstracted into finite state systems [4], which makes model checking decidable. Model checking tools for timed automata are available, e.g. *KRONOS* [28], *UPPAAL* [22].

This work was partially supported by the HYCON Network of Excellence (c.n. FP6-IST-511368), by MIUR (Project SCEF - PRIN05), and by NSF Presidential Early CAREER (PECASE, Grant 0132716).

[♭] Department of Electrical Engineering and Computer Science, Center of Excellence DEWS. University of L'Aquila, Italy. email: (adinnoce,dibenede,digennar)@ing.univaq.it

[♯] Department of Electrical and Systems Engineering. University of Pennsylvania, USA. email: (agung@seas,pappasg@ee).upenn.edu

The contribution of this paper is to show that, given a hybrid automaton and the durational graph obtained by the algorithm that we proposed in [14], there exists a *simulation relation* between the two systems. To achieve this, we naturally embed the hybrid automaton and the durational graph in transition systems, and show that these systems satisfy a simulation relation. This implies that the abstraction inherits the universal fragment of temporal logic properties of the hybrid automaton, namely properties that must be satisfied by all executions of the system. The variant of temporal logic that we use in this paper is CTL (Computation tree logic [12]) and TCTL (Timed CTL [1]). The additional discussion we present in this paper is motivated because temporal properties have empty intersection with the observability property defined in [14]. In constructing the timed automaton, we abstract the information about the continuous dynamics with a clock dynamics that measures the amount of time spent in each location. Constructing the guards of the timed automaton involves computation of the time spent in the location. For simple dynamics, this can be done analytically [6], [21]. For more complex dynamics, there are computational tools that can help [9], [10], [11], [27], [16]. The algorithm proposed here is more general than the algorithm proposed in [14], since it includes invariant sets. A further contribution of the paper is to implement and apply our procedure to verify safety and efficient scheduling in the context of air traffic management.

The paper is organized as follows. In Section II we give basic definitions of hybrid automata, timed automata, transition systems, simulation and bisimulation relation. In Section III we review the algorithm developed in [14] to construct a durational graph $\mathcal{G}$ given a hybrid automaton $\mathcal{H}$. In Section IV we prove that our abstraction $\mathcal{G}$ inherits a class of the temporal logic properties (i.e. the universal fragment) of $\mathcal{H}$. In Section V we apply the theoretical results on a relevant real *ATM* case study: how to schedule aircraft landing. Conclusions and outline for future work follow in the last section.

II. Basic definitions

Systems that have both discrete and continuous aspects in their dynamics are called hybrid systems. One prominent theoretical framework that is used to model hybrid systems is proposed by Lygeros [24], where the discrete part consists of a labeled oriented graph, and the continuous part is described by a dynamical continuous system associated to each discrete state. The interaction between the continuous and discrete part is described by invariant, guard, and reset conditions. We consider here hybrid automata, that are hybrid systems with autonomous dynamics.

1-4244-1498-9/07/$25.00 ©2007 IEEE.

Definition 1 (Hybrid automaton): A hybrid automaton is a tuple $\mathcal{H} = (Q \times X, Q_0 \times X_0, \mathcal{E}, E, \Omega, \omega, Inv, G, R)$ such that: $Q \times X$ is the hybrid state space, where Q is a finite set of discrete states $\{q_1, q_2, \cdots, q_N\}$, and $X \subseteq \mathbb{R}^n$ is the continuous state space; $Q_0 \times X_0 \subseteq Q \times X$ is the set of initial discrete and continuous conditions; $\{\mathcal{E}_q\}_{q \in Q}$ associates to each discrete state the autonomous continuous time–invariant dynamics $\mathcal{E}_q : \dot{x} = f_q(x)$. Given an initial condition x_0, we define the solution at time t according to f_q by $x(t) = x_{f_q}(t, x_0)$; $E \subseteq Q \times Q$ is a collection of edges; each edge $e \in E$ is an ordered pair of discrete states, the first component of which is the source and is denoted by $s(e)$, while the second is the target and is denoted by $t(e)$; Ω is a space of logic propositions. We associate to each discrete state $q \in Q$ an element of Ω, by the function $\omega : Q \to \Omega$. This allows to define properties that characterize a discrete state of the system by means of a logic proposition, similarly to the classical framework of Kripke structures [12]; $\{Inv_q\}_{q \in Q}$ associates to each discrete state an invariant set $Inv_q \subseteq X$; $\{G_e\}_{e \in E}$ associates to each edge a guard set $G_e \subseteq Inv_{s(e)}$; $\{R_e\}_{e \in E}$ associates to each edge a reset map $R_e : Inv_{s(e)} \to 2^{Inv_{t(e)}}$.
Note that a hybrid automaton is generally nondeterministic: the continuous state evolves following deterministic dynamics, and the discrete state evolution depends only on the continuous state, according to guards, possibly with nondeterministic behaviors in the discrete transitions and reset. Guards are enabling conditions: when the continuous state hits a guard, a transition might or might not occur. On the other hand, a transition is forced to occur when the continuous state exits the invariant set and at least one guard is enabled. In the following, we denote the set of incoming edges in q by $inc(q) \triangleq \{e \in E : t(e) = q\}$.

Referring to [24], we recall the definitions of hybrid time basis and hybrid execution of a hybrid system. A *hybrid time basis* $\tau \triangleq \{I_k\}_{k \geq 0}$ is a finite or infinite sequence of intervals $I_k = [t_k, t'_k]$ such that properties stated in [24] hold. The number of intervals is the cardinality $|\tau|$ of the time basis. A *hybrid execution* is a triple $\chi = (\tau, q, x)$, where τ is a hybrid time basis, and q, x describe the evolution of the discrete and continuous state by means of functions $q : \tau \to Q$ piecewise continuous, and $x : \tau \to X$. Functions q, x are defined on the hybrid time basis τ, take values on the hybrid state space, and satisfy the continuous and discrete dynamics and their interactions (invariant, guard and reset). In this paper, we consider *non blocking* hybrid automata, i.e. systems such that all hybrid executions are defined for all time instants. We do not exclude that many transitions (possibly infinite) might occur in a finite time interval, i.e. we allow *Zeno executions*.

We call *durational graph* a *timed automaton* [3] characterized by only one clock that is reset to 0 for all edges:

Definition 2 (Durational graph): A durational graph is a hybrid automaton $(Q \times X, Q_0 \times X_0, \mathcal{E}, E, \Omega, \omega, Inv, G, R)$ such that: $X = \mathbb{R}_+ \cup \{0\}$ is the continuous state space of the clock variable v; for each $q_0 \in Q_0$, the initial condition is given by $(q_0, 0)$; for each $q \in Q$, the continuous dynamics are defined by $\mathcal{E}_q : \dot{v} = 1$; for each $q \in Q$, the set Inv_q is a rectangular set[1]; for each $e \in E$, the set G_e is a rectangular set and $R_e(v) = \{0\}$.
By Definition 2, a durational graph is uniquely identified by a tuple $\mathcal{G} = (Q, Q_0, E, \Omega, \omega, Inv, G)$.

We use the framework of transition systems to model both hybrid automata and durational graphs [25] in the same mathematical framework. We apply the canonical definition of *simulation relation* given for transition systems to relate the hybrid automaton and the durational graph abstraction.

Definition 3 (Transition system): A (labeled) transition system with observations is a tuple $\mathcal{T} = (Q, Q_0, \Sigma, E, \Omega, \omega)$ that consists of a possibly infinite set Q of states, a possibly infinite set $Q_0 \subseteq Q$ of initial states, a possibly infinite set Σ of labels, a transition relation $E \subseteq Q \times \Sigma \times Q$, a possibly infinite set Ω of observations and an observation map $\omega : Q \to \Omega$.
In what follows, we use the notation $q \xrightarrow{\sigma} q'$ to denote that $(q, \sigma, q') \in E$, where $q, q' \in Q$ and $\sigma \in \Sigma$. Let $\mathcal{T}_1 = (Q_1, Q_0^1, \Sigma_1, E_1, \Omega_1, \omega_1)$ and $\mathcal{T}_2 = (Q_2, Q_0^2, \Sigma_2, E_2, \Omega_2, \omega_2)$ be two labeled transition systems with the same set of labels ($\Sigma_1 = \Sigma_2 = \Sigma$) and the same set of observations ($\Omega_1 = \Omega_2 = \Omega$).

Definition 4 (Simulation relation): A relation $\Gamma \subseteq Q_1 \times Q_2$ is called a simulation relation of $\mathcal{T}_1$ by $\mathcal{T}_2$, if for all $(q_1, q_2) \in \Gamma$:

(i) $\omega_1(q_1) = \omega_2(q_2)$,
(ii) for all $q_1 \xrightarrow{\sigma} q'_1$, there exists $q_2 \xrightarrow{\sigma} q'_2$ such that $(q'_1, q'_2) \in \Gamma$.

A relation Γ is called a bisimulation when it is both a simulation of $\mathcal{T}_1$ by $\mathcal{T}_2$, and a simulation of $\mathcal{T}_2$ by $\mathcal{T}_1$.

Definition 5 (Simulation): $\mathcal{T}_2$ simulates $\mathcal{T}_1$ (denoted $\mathcal{T}_1 \preceq \mathcal{T}_2$) if there exists Γ, a simulation relation of $\mathcal{T}_1$ by $\mathcal{T}_2$, such that for any $q_1 \in Q_0^1$, there exists $q_2 \in Q_0^2$ such that $(q_1, q_2) \in \Gamma$.
If any initial state of $\mathcal{T}_1$ can be related to any initial state of $\mathcal{T}_2$ and conversely, then $\mathcal{T}_1$ and $\mathcal{T}_2$ simulate each other, and we say that $\mathcal{T}_1$ and $\mathcal{T}_2$ are bisimilar.

III. Translating hybrid to timed automata

Since we are interested in verifying temporal properties of a system, we need an abstract model that embeds time; for this reason, we use a class of timed automata. We propose here a slightly modified version (that also considers the invariant sets) of the algorithm developed in [14], to construct a durational graph that is an abstraction of a hybrid automaton. Given a hybrid automaton $\mathcal{H}$ as in Definition 1, we define $X_0(q_0) \triangleq \{x_0 \in X_0 : (x_0, q_0) \in X_0 \times Q_0\}$ the set of initial continuous conditions associated to the initial discrete state $q_0 \in Q_0$. The non blocking assumption requires that $X_0(q_0) \subseteq Inv_{q_0}$. Moreover, we define $Range(R_e) \triangleq \{x \in X : \exists x' \in G_e, x \in R_e(x')\}$ the range of the reset associated to edge $e \in E$. The non blocking assumption requires that $\forall q \in Q, \forall e \in inc(q), Range(R_e) \subseteq Inv_q$: i.e., the reset "lands" in the invariant of the target location. Define now a relation $\gamma \subseteq Q \times \big(Q \times (E \cup Q_0)\big)$ as follows:

$$\gamma \triangleq \{(q, (q, l)) : q \in Q \text{ and either } l \in inc(q) \text{ or } l = q \in Q_0\}$$

For $l \in E \cup Q_0$, let

$$\Re_l \triangleq \begin{cases} X_0(q_0) & \text{if} \quad l = q_0 \in Q_0 \\ Range(R_e) & \text{if} \quad l = e \in E \end{cases}$$

[1] a rectangular set in $\mathbb{R}^n$ is any subset that can be defined by a finite union of cartesian products of intervals.

Algorithm 1: Consider $\mathcal{H} = (Q \times X, Q_0 \times X_0, \mathcal{E}, E, \Omega, \omega, Inv, G, R)$. We define a durational graph $\mathcal{G} = (Q', Q'_0, E', \Omega', \omega', Inv', G')$ as follows:

1) $Q' \triangleq \{(q,l) : (q,(q,l)) \in \gamma\}$;
2) $Q'_0 \triangleq \{(q,l) \in Q' : q \in Q_0, l = q\}$;
3) $E' \triangleq \big\{\big((q_1,l_1),(q_2,l_2)\big) : l_2 = (q_1,q_2)$ and either $l_1 \in inc(q_1)$ or $l_1 = q_1 \in Q_0\big\}$;
4) $\Omega' = \Omega$, and $\forall (q,l) \in Q', \omega'((q,l)) = \omega(q)$;
5) $\forall q' = (q,l) \in Q'$, define:

$$Inv'_{q'} \triangleq \{t \in \mathbb{R}_+ \cup \{0\} : \exists x_0 \in \Re_l, \; x_{f_q}(\tau, x_0) \in Inv_q, \forall \tau \in [0,t]\}. \quad (1)$$

6) $\forall e' = \big((q_1,l_1),(q_2,l_2)\big) \in E'$, define:

$$G'_{e'} \triangleq \{t \in \mathbb{R}_+ \cup \{0\} : \exists x_0 \in \Re_{l_1}, \; x_{f_{q_1}}(t, x_0) \in G_{l_2}\}. \quad (2)$$

The idea of the algorithm is to split each discrete state depending on the number of incoming edges, and to define invariant and guard sets of the durational graph by means of the dwell time in each discrete state of the hybrid automaton. The main issue is the computation of the invariant and guard sets $I_G = \{G'_{e'}\}_{e' \in E'}, I_{Inv} = \{Inv'_{q'}\}_{q' \in Q'}$. If the dynamics $f_q(x)$ are linear, the exact computation is possible when the system has a particular structure [6], [21]. Whenever the exact computation is not possible, one can compute the over approximations I^*_G, I^*_{Inv} e.g. using the Matlab routine proposed in [27]. The algorithm is very fast, even for high dimensional continuous state spaces, but there is no analysis of the over approximation error. It is also possible to use the result in [10], which provides a procedure to compute a sequence of polytopes (a *flow pipe*) that are over approximations of the reach sets $\{Reach_{[T_k,T_{k+1}]}(X_0)\}_{k \geq 0}$ for arbitrary small *sampling time* T. By checking for each $k \geq 0$ if the intersection set $Reach_{[T_k,T_{k+1}]}(X_0) \cap X_F$ is empty, it is possible to determine rectangular time intervals $I^*_G \supseteq I_G$ and $I^*_{Inv} \supseteq I_{Inv}$ with arbitrary precision. The weak point of this approach is the increase in the computation time. Another procedure, similar to [10], is presented in [16], where an algorithm to compute a sequence of zonotopes[2] is proposed. This algorithm is very fast, because the computation of the *flow pipe* and of the intersections with X_F is considerably faster for zonotopes than for polytopes [17]. Moreover, the algorithm computes the reach set in the presence of a control input that takes values in a bounded set. Recent results that can also be used for computing I^*_G, I^*_{Inv} are given in [18], [20]. If the dynamics are nonlinear, we can use the approach of [10]. In the case study of section V, we will use the framework developed in [20].

IV. Main result

The following remark shows that we can model check a hybrid automaton $\mathcal{H}$ on a conservative abstraction $\mathcal{G}$.

Remark 1: It is well known that simulation relations preserve all properties expressible in the universal fragment of CTL or TCTL.

In [14] it was proven that $\mathcal{G}$ preserves observability. The following remark justifies the additional discussion we present in this paper.

[2]A zonotope is a centrally symmetric polytope, defined by the Minkowski sum of its line segment generators $s_1, \cdots, s_m \in \mathbb{R}^n$.

Remark 2: The temporal logics CTL and TCTL have empty intersection with the observability property as defined in [14], since they do not allow one to express the equivalence of two output strings.

We prove here that $\mathcal{G}$ inherits the universal fragment of temporal properties of $\mathcal{H}$. To this aim, we consider the transition systems $\mathcal{T}^{\mathcal{H}}, \mathcal{T}^{\mathcal{G}}$ that naturally embed $\mathcal{H}, \mathcal{G}$. Then, we prove that $\mathcal{T}^{\mathcal{H}} \preceq \mathcal{T}^{\mathcal{G}}$ according to Definition 5.

Definition 6 (Transition system model of $\mathcal{H}$): Given a hybrid automaton $\mathcal{H}$, we define a transition system $\mathcal{T}^{\mathcal{H}}$ as follows:

- $Q^{\mathcal{T}^{\mathcal{H}}} = Q^{\mathcal{H}} \times X^{\mathcal{H}}$ is the state space: a state of $Q^{\mathcal{T}^{\mathcal{H}}}$ is a pair (q,x), with $q \in Q^{\mathcal{H}}$ and $x \in X^{\mathcal{H}}$;
- $Q_0^{\mathcal{T}^{\mathcal{H}}} = Q_0^{\mathcal{H}} \times X_0^{\mathcal{H}}$ is the set of initial states;
- $\Sigma^{\mathcal{T}^{\mathcal{H}}} = \mathbb{R}_+ \cup \{0\}$ is the set of labels (time flow);
- $E^{\mathcal{T}^{\mathcal{H}}}$ is the transition relation defined as follows:
  1) $(q,x) \xrightarrow{t} (q,x')$ for $t \neq 0$ if:
  $$x' = x_{f_q}(t,x) \wedge \forall \tau \in [0,t], x_{f_q}(\tau,x) \in Inv_q^{\mathcal{H}}. \quad (3)$$
  2) $(q,x) \xrightarrow{t} (q',x')$ for $t = 0$ if:
  $$e = (q,q') \in E \wedge x \in G_e^{\mathcal{H}} \wedge x' \in R_e^{\mathcal{H}}(x). \quad (4)$$
- $\Omega^{\mathcal{T}^{\mathcal{H}}} = \Omega^{\mathcal{H}}$ is the set of observations;
- $\omega^{\mathcal{T}^{\mathcal{H}}}$ is the observation map, where $\omega^{\mathcal{T}^{\mathcal{H}}}(q,x) = \omega^{\mathcal{H}}(q)$.

Definition 7 (Transition system model of $\mathcal{G}$): Given a durational graph $\mathcal{G}$, we define a transition system $\mathcal{T}^{\mathcal{G}}$ as follows:

- $Q^{\mathcal{T}^{\mathcal{G}}} = Q^{\mathcal{G}} \times \mathbb{R}_+ \cup \{0\}$ is the state space: a state of $Q^{\mathcal{T}^{\mathcal{G}}}$ is a pair (q,x), with $q \in Q^{\mathcal{G}}$ and $x \in \mathbb{R}_+ \cup \{0\}$;
- $Q_0^{\mathcal{T}^{\mathcal{G}}} = Q_0^{\mathcal{G}} \times \{0\}$ is the set of initial states;
- $\Sigma^{\mathcal{T}^{\mathcal{G}}} = \mathbb{R}_+ \cup \{0\}$ is the set of labels;
- $E^{\mathcal{T}^{\mathcal{G}}}$ is the transition relation defined as follows:
  1) $(q,x) \xrightarrow{t} (q,x')$ for $t \neq 0$ if:
  $$x' = x + t \wedge \forall \tau \in [0,t], x + \tau \in Inv_q^{\mathcal{G}}. \quad (5)$$
  2) $(q,x) \xrightarrow{t} (q',x')$ for $t = 0$ if:
  $$e = (q,q') \in E^{\mathcal{G}} \wedge x \in G_e^{\mathcal{G}} \wedge x' \in R_e^{\mathcal{G}}(x). \quad (6)$$
- $\Omega^{\mathcal{T}^{\mathcal{G}}} = \Omega^{\mathcal{G}}$ is the set of observations;
- $\omega^{\mathcal{T}^{\mathcal{G}}}$ is the observation map, where $\omega^{\mathcal{T}^{\mathcal{G}}}(q,v) = \omega^{\mathcal{G}}(q)$.

We recall that Algorithm 1 takes as input a hybrid automaton $\mathcal{H}$, and produces as output a durational graph $\mathcal{G}$ and a relation $\gamma \subseteq Q^{\mathcal{H}} \times Q^{\mathcal{G}}$. The following theorem states that $\mathcal{T}^{\mathcal{G}}$ simulates $\mathcal{T}^{\mathcal{H}}$.

Theorem 1: $\mathcal{T}^{\mathcal{H}} \preceq \mathcal{T}^{\mathcal{G}}$.

By Remark 1 it follows that if $\mathcal{T}^{\mathcal{G}}$ simulates $\mathcal{T}^{\mathcal{H}}$, then $\mathcal{G}$ inherits the universal fragment of temporal properties of $\mathcal{H}$.

V. Verification of safety and scheduling efficiency in the landing procedure

In this section, we analyze an important procedure in the context of air traffic management: the landing scheduling procedure. This procedure involves two aircraft (*A/C 1* and *A/C 2*) incoming from two different air traffic routes A_1 and A_2, as shown in Figure 1. A conflict resolution is completely assigned to the approaching air traffic controller (*ATC*). For

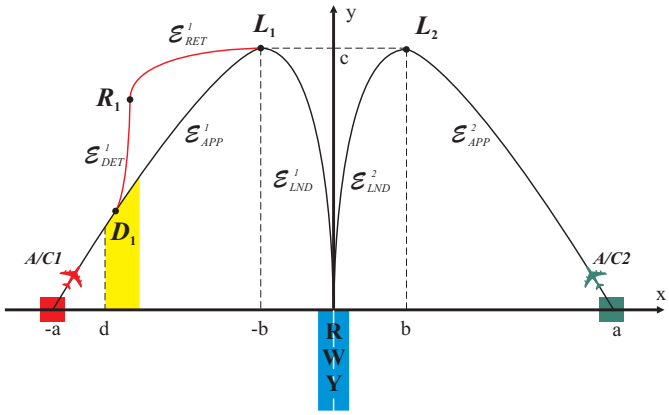

Fig. 1. Landing scheduling scheme.

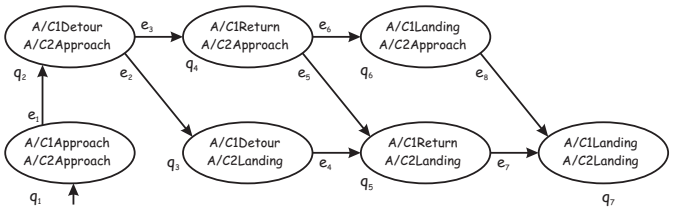

Fig. 2. Discrete layer of $\mathcal{H}$.

instance, the two aircraft could arrive in the landing areas L_1 and L_2 too close one to each other: in this case, the *ATC* usually gives the command to change heading to one of the two aircraft, say *A/C 1*, in order to delay its arrival in the landing area L_1. We assume without loss of generality that the command to change heading is given to *A/C 1*, and that the decision is taken when *A/C 1* is in the *detour zone* D_1. The *ATC* estimates by the current velocity and position measurements of the aircraft, if they could arrive in the landing area too close. The *ATC* has access to the measurement from the radar, and he has to make a quick decision. Typically, safety is the only considered factor. There is no explicit consideration for minimizing the inter–arrival time delay. Furthermore, because of limited time availability, the *ATC* makes its decision based on qualitative and rough estimations of the inter–arrival delay. The reasoning of the *ATC* in this conflict resolution corresponds to the construction, on the basis of the data available from the radar, of a timed model of the aircraft operations, and to the verification that the inter–arrival time to the landing area is not too small.

With the aim of helping the *ATC* to solve the conflict, we can apply the theoretical results of the previous section on a model of the landing scheduling procedure. We propose an optimization of landing traffic scheduling protocol that satisfies not only safety distance, but also a maximal inter–arrival delay. First, we formally define a hybrid model $\mathcal{H}$ of the system. The continuous layer is given by the positions of the aircraft. We propose linear decoupled dynamics for *A/C 1* and *A/C 2*. The discrete layer is given by the current operation of each aircraft, e.g. *A/C 1* can be approaching to the landing zone, changing heading towards R_1, or landing. In Figure 1 both *A/C 1* and *A/C 2* are approaching towards zones L_1 and L_2. Once arrived, they start landing on the runway (*RWY*) at the origin of the axis. The switching between discrete states of the aircraft, and the protocol that is used to decide if *A/C 1* has to change its route, is embedded in the guard and invariant sets. Thus, the conflict resolution protocol is embedded in the hybrid model $\mathcal{H}$. By following the procedure proposed in the previous sections, we construct a durational graph $\mathcal{G}$. By Remark 1, universal temporal properties expressed in TCTL [1] are preserved by the abstraction $\mathcal{G}$. The following step is to (model) check if such a protocol ensures safety, namely the two aircraft do not arrive in the landing area too close one to each other. In addition to this, we also check that the inter–arrival time delay does not exceed a given upper bound. We try to modulate the position of the detour zone D_1 in order to satisfy both safety and efficient traffic scheduling. Following the notation of Figure 1, we define a hybrid model $\mathcal{H}$ for the two aircraft system as follows:

$Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$. In q_1 *A/C 1* and *A/C 2* are both approaching respectively towards L_1 and L_2. In q_2 *A/C 1* starts the detour by changing heading towards R_1. In q_3 *A/C 1* is still proceeding towards R_1 and *A/C 2* is landing. In q_4 *A/C 1* is approaching towards L_1 after the detour and *A/C 2* is still approaching towards L_2. In q_5 *A/C 1* is approaching towards L_1 after the detour and *A/C 2* is landing. In q_6 *A/C 1* is landing and *A/C 2* is still approaching towards L_1. In q_7 both aircraft are landing, hopefully not simultaneously.

Ω is the set of logic propositions on the set of variables $\{\phi_1, \phi_2\}$, where ϕ_1 is true if *A/C 1* is in landing mode, and ϕ_2 is true if *A/C 2* is in landing mode. The function $\omega : Q \to \Omega$ associates to each discrete state $q \in Q$ a formula in Ω:

$$\omega(q_1) = \omega(q_2) = \omega(q_4) = \neg\phi_1 \wedge \neg\phi_2.$$
$$\omega(q_3) = \omega(q_5) = \neg\phi_1 \wedge \phi_2.$$
$$\omega(q_6) = \phi_1 \wedge \neg\phi_2,\ \omega(q_7) = \phi_1 \wedge \phi_2.$$

$\mathbb{R}^7 \times \mathbb{R}^7$ is the continuous state space. The state variable of each aircraft $i = 1, 2$ given by the vector $\chi_i = (\beta_i, r_i, p_i, \varphi_i, \psi_i, x_i, y_i)$, where β_i is the side-slip angle, r_i is the yaw rate, p_i is the roll rate, φ_i is the roll angle, ψ_i is the yaw angle, and (x_i, y_i) is the aircraft position on the plan in Figure 1.

$\{q_1\}$ is the initial discrete state, $\{(0, 0, 0, 0, \frac{1}{4}\pi)\}$ and $\{(0, 0, 0, 0, \frac{3}{4}\pi)\}$ are the initial conditions of the variables $(\beta_i, r_i, p_i, \varphi_i, \psi_i)$ for $i = 1, 2$. The sets $[-a-\alpha_1, -a+\alpha_1] \times [-\alpha_1, +\alpha_1]$ and $[a - \alpha_1, a + \alpha_1] \times [-\alpha_1, +\alpha_1]$, $\alpha_1 > 0$, are the sets of initial conditions of the variables (x_i, y_i): these sets model the arrival positions of the two aircraft introducing non determinism in the hybrid model. When safety distance is respected, the continuous dynamics $\mathcal{E}^1$ and $\mathcal{E}^2$ of each aircraft are decoupled. The dynamics of the variables $(\beta_i, r_i, p_i, \varphi_i, \psi_i)$ are obtained for each aircraft by a feedback control loop designed to asymptotically pursue a reference value ψ_r of the yaw angle, and are characterized by a LTI system $\mathcal{E}(\psi_r)$ defined by the matrices:

$$A = \begin{bmatrix} -0.558 & -0.997 & 0.08 & 0.041 & 0 & 0.007 \\ 0.598 & -0.115 & -0.032 & 0 & 0 & -0.475 \\ -3.05 & 0.388 & -0.465 & 0 & 0 & 0.153 \\ 0 & 0.08 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \quad B = \begin{bmatrix} 0.0073 \\ -0.4750 \\ 0.1530 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

with reference input ψ_r. Thus $\{\mathcal{E}_q^1 \times \mathcal{E}_q^2\}_{q\in Q}$ can be defined by means of the reference yaw angle ψ_r, that defines the heading for the aircraft:

$$\begin{aligned}
&\mathcal{E}^1_{q_1} = \mathcal{E}(\frac{\pi}{4}), && \mathcal{E}^1_{q_2} = \mathcal{E}^1_{q_3} = \mathcal{E}(\frac{\pi}{2}),\\
&\mathcal{E}^1_{q_4} = \mathcal{E}^1_{q_5} = \mathcal{E}(0), && \mathcal{E}^1_{q_6} = \mathcal{E}^1_{q_7} = \mathcal{E}(-\frac{\pi}{2}),\\
&\mathcal{E}^2_{q_1} = \mathcal{E}^2_{q_2} = \mathcal{E}^2_{q_4} = \mathcal{E}^2_{q_6} = \mathcal{E}(-\frac{\pi}{4}),\\
&\mathcal{E}^2_{q_3} = \mathcal{E}^2_{q_5} = \mathcal{E}^2_{q_7} = \mathcal{E}(-\frac{\pi}{2}).
\end{aligned}$$

The dynamics of the variables (x_i, y_i) are defined by the nonlinear dynamics $\dot{x}_i = V_i \cos\psi_i, \dot{y}_i = V_i \sin\psi_i$. We assume for simplicity and w.l.o.g. that the norm V_i of the speed of aircraft i is constant.

E is defined by the directed edges in Figure 2. The invariant sets defined as follows:

$$\begin{aligned}
Inv_{q_1} &= \{(\chi_1,\chi_2) : x_1 < -d + \alpha_2, x_2 > b\}\\
Inv_{q_2} &= \{(\chi_1,\chi_2) : y_1 < c, x_2 > b\}\\
Inv_{q_3} &= \{(\chi_1,\chi_2) : y_1 < c, x_2 < b\}\\
Inv_{q_4} &= \{(\chi_1,\chi_2) : x_1 < -b, x_2 > b\}\\
Inv_{q_5} &= \{(\chi_1,\chi_2) : x_1 < -b, x_2 < b\}\\
Inv_{q_6} &= \{(\chi_1,\chi_2) : x_1 > -b, x_2 > b\}\\
Inv_{q_7} &= \{(\chi_1,\chi_2) : x_1 > -b, x_2 < b\}
\end{aligned}$$

The guard sets defined as follows:

$$\begin{aligned}
&G_{e_1} = \{(\chi_1,\chi_2) : x_1 \geq -d\}, && G_{e_2} = \{(\chi_1,\chi_2) : x_2 \leq b\},\\
&G_{e_3} = \{(\chi_1,\chi_2) : y_1 \geq c\}, && G_{e_4} = \{(\chi_1,\chi_2) : y_1 \geq c\},\\
&G_{e_5} = \{(\chi_1,\chi_2) : x_2 \leq b\}, && G_{e_6} = \{(\chi_1,\chi_2) : x_1 \geq -b\},\\
&G_{e_7} = \{(\chi_1,\chi_2) : x_1 \geq -b\}, && G_{e_8} = \{(\chi_1,\chi_2) : x_2 \leq b\}
\end{aligned}$$

with $\alpha_2 > 0$. The reset is defined by the identity function $\forall e \in E, \forall x \in G_e, R_e(x) = x$; α_1 is a parameter that can be

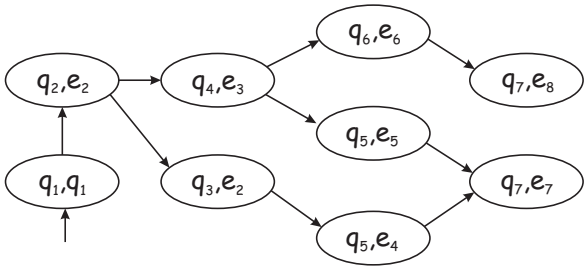

Fig. 3. Durational graph $\mathcal{G}$.

used to introduce non determinism in the initial position of the two aircraft. Similarly, α_2 introduces non determinism in the position where the veer manoeuvre is started by *A/C 1*, w.r.t. the desired value d. The same reasoning can be applied when introducing non determinism in the model to take in consideration radar measure errors, nondeterministic delay by the pilot flying in executing a manoeuvre due to communication delay or human inaccuracy, and nondeterministic anticipation or delay by the *ATC* in taking a decision or giving a command to the pilot.

Using the procedure proposed in Section III, it is possible to construct from $\mathcal{H}$ the durational graph $\mathcal{G}$ illustrated in Figure 3. We have computed the guard sets of $\mathcal{G}$ using a Matlab routine that exploits the methodology proposed in [20], that can be applied to linear dynamics. To model the change of heading e.g. from $\psi_r = \frac{\pi}{4}$ to $\psi_r = \frac{\pi}{2}$ (transition e_1 of $\mathcal{H}$) with a LTI model, and since the dynamics of (x_i, y_i) are nonlinear, we have implemented in our algorithm a linearized hybrid automaton where for each discrete state the value of the variable ψ_i is such that $\sin\psi_i \simeq \psi_i$ and $\cos\psi_i \simeq 1$ in the neighborhood of the reference heading ψ_r. Roughly speaking, we split the veer in a finite sequence of small changes of heading. We have used in our routine the following numeric values: $a = 100$ km, $b = 10$ km, $c = 90$ km, $V_1 = V_2 = 400$ km/h, $\alpha_1 = 5$ km, $\alpha_2 = 10$ km. The variable d is the desired position where the detour command is given to *A/C 1* to change the heading to $\psi_r = \frac{\pi}{2}$. The set of positions where the veer might start is represented by zone D_1 in Figure 1. Theorem 1 implies that $\mathcal{H} \preceq \mathcal{G}$, and by Remark 1 it follows that universal formulae satisfied by $\mathcal{G}$ are also satisfied by $\mathcal{H}$. The properties we require from the landing scheduling protocol are the following:

1) **Safety:** we require that a landing of *A/C 1* can only start after a safety delay t_{min} since landing of *A/C 2* is started.
2) **Efficient traffic scheduling:** we require that landing of *A/C 1* must start within a time t_{max} since landing of *A/C 2* is started.

Thus, in order to check if the system $\mathcal{H}$ satisfies safety and efficient traffic scheduling, we can model check on $\mathcal{G}$ the following TCTL formula:

$$\forall\left(\neg\phi_1 \wedge \neg\phi_2 \,\mathcal{U}_{t\geq 0}\left(\neg\phi_1 \wedge \phi_2 \,\mathcal{U}_{t_{min}\leq t\leq t_{max}} \phi_1 \wedge \phi_2\right)\right)$$

Namely *A/C 1* starts landing only after *A/C 2* is already landing, with a delay time in the set $[t_{min}, t_{max}]$. The verification of this formula requires a TCTL verifier for timed automata, e.g. *KRONOS* [28]. In our case, since the clock of a durational graph is always reset to zero, we have been able to use deductive reasoning to model check the formula on $\mathcal{G}$. Let us require that the inter–arrival delay

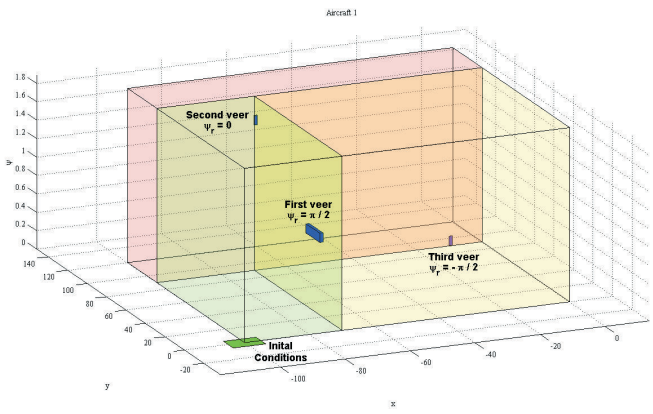

Fig. 4. Plot of the abstraction algorithm output for aircraft *A/C 1*, for $d = -70$. The blue sets are the over approximations of the aircraft positions when the guards are hit.

belongs to the time interval $[t_{min} = 60, t_{max} = 180]$s. Since

Detour zone	Inter-arrival time	Valuation
$d = -70$	$[53, 155]$s	possibly unsafe
$d = -80$	$[71, 173]$s	safe & efficient
$d = -90$	$[88, 190]$s	possibly inefficient

TABLE I

INTER–ARRIVAL TIMES DEPENDING ON THE DETOUR POSITION d

our abstraction algorithm runs in a few seconds, we have performed simulations modulating the value of d to yield the desired properties on the system, and model checked the formula on the system $\mathcal{G}_d$. As shown in Table I, for $d = -70$ km the safety constraint is not satisfied, since the minimum inter-arrival time is $53\text{s} < 60\text{s}$. For $d = -90$ km the system is safe but the efficiency specification is not satisfied, since the maximum inter–arrival time is $190\text{s} > 180\text{s}$. With a detour zone set at $d = -80$ km both safety and efficient traffic scheduling requirements are satisfied, and the inter–arrival gap is $[71, 173]$s.

VI. CONCLUSIONS

In this work we proposed a procedure to verify temporal properties of a hybrid automaton $\mathcal{H}$ using a durational graph $\mathcal{G}$, that is an abstraction of the original system. We showed that $\mathcal{G}$ preserves certain temporal properties of $\mathcal{H}$, namely the universal TCTL formulae. We applied the procedure to a relevant real case study in the context of air traffic management, the landing scheduling. Since our current implementation only generates the abstraction, we did not relate it to the existing model checking tools for hybrid automata. We plan to build a tool that implements the abstraction algorithm and provides as output the standard input file format for *UPPAAL* [22], so that model checking can be performed automatically on the hybrid automaton. Current work also deals with the definition of a metric for quantifying the distance between timed trajectories of the original system and of the abstraction.

REFERENCES

[1] R. Alur, C. Courcoubetis, and D.L. Dill. Model–checking in dense real–time. *Information and Computation*, 104(1):2–34, 1993.

[2] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[3] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[4] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. *Proccedings of the IEEE*, 88(2):971–984, July 2000.

[5] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.

[6] H. Anai and V.Weispfenning. Reach set computations using real quantifier elimination. In M. D. Di Benedetto and A.L. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 103–117. Springer Verlag, 2001.

[7] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise linear dynamical systems. In *Hybrid Systems: Computation and Control, Pittsburgh, USA*, Lecture Notes in Computer Science. Springer Verlag, March 2000.

[8] M. Babaali and G. J. Pappas. Observability of switched linear systems in continuous time. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 103–117. Springer Verlag, 2005.

[9] A. Bemporad, F. D. Torrisi, and M. Morari. Discrete–time hybrid modeling and verification of the batch evaporator process benchmark. *European Journal of Control*, 7(4):382–399, 2001.

[10] A. Chutinan and B. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In *Proceedings of the 37^{th} IEEE Conference on Decision and Control, Tampa, FL*, pages 2089–2094, December 1998.

[11] A. Chutinan and B.H. Krogh. Computing approximating automata for a class of linear hybrid systems. *In Hybrid Systems V, Lecture Notes in Computer Science, Springer Verlag*, pages 16–37, 1998.

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

[13] E. De Santis, M. D. Di Benedetto, S. Di Gennaro, A. D'Innocenzo, and G. Pola. Critical observability of a class of hybrid systems and application to air traffic management. *Book Chapter of Lecture Notes on Control and Information Sciences, Springer Verlag*, 2005.

[14] A. D'Innocenzo, 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 169–183. Springer Verlag, 2006.

[15] A. D'Innocenzo, A.A. Julius, G.J. Pappas, M.D. Di Benedetto, and S. Di Gennaro. Verification of temporal properties on hybrid automata by simulation relations. Technical report, University of L'Aquila, 2007. www.diel.univaq.it/tr/web/web_search_tr.php.

[16] 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 291–305. Springer Verlag, 2005.

[17] L.J. Guibas, An Nguyen, and Li Zhang. Zonotopes as bounding volumes. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms. Baltimore, Maryland, USA*, pages 803–812, 2003.

[18] Zhi Han and B. H. Krogh. Reachability analysis of large–scale affine systems using low–dimensional polytopes. In J. Hespanha and A. Tiwari, editors, *Hybrid Systems: Computation and Control*, volume 3927 of *Lecture Notes in Computer Science*, pages 287–301. Springer Verlag, 2006.

[19] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:110–122, 1997.

[20] A.A. Julius, G. Fainekos, M. Anand, I. Lee, and G.J. Pappas. Robust test generation and coverage for hybrid systems. In *Hybrid Systems: Computation and Control, To appear*, Lecture Notes in Computer Science. Springer Verlag, 2007.

[21] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic reachability computations for families of linear vector fields. *Journal of Symbolic Computation*, 32(3):231–253, September 2001.

[22] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1):134–152, December 1997.

[23] J. Lunze. Discrete-event modelling and fault diagnosis of discretely controlled continuous systems. In *Proceedings of the 2^{nd} IFAC Conference on Analysis and Design of Hybrid Systems (ADHS), Alghero, Sardinia, Italy*, June 7-9 2006.

[24] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specications for hybrid systems. *Automatica, Special Issue on Hybrid Systems*, 35, 1999.

[25] G.J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035–2047, December 2003.

[26] O. Stursberg and S. Kowalewsky. Approximating switched continuous systems by rectangular automata. In *Proceedings of the 1999 European Control Conference, Karlsruhe, Germany*, 1999.

[27] H. Yazarel and G. J. Pappas. Geometric programming relaxations for linear system reachability. In *Proceedings of the 2004 American Control Conference, Boston, MA*, June 2004.

[28] S. Yovine. Kronos: A verification tool for real-time systems. *International Journal of Software Tools for Technology Transfer, Springer-Verlag*, 1(1):123–133, October 1997.