Temporal Precedence Checking for Switched Models and its Application to a Parallel Landing Protocol

Parasara Sridhar Duggirala[1], Le Wang[2], Sayan Mitra[2], and Mahesh Viswanathan[1]

[1] Dept. of Computer Science, Univ. of Illinois at Urbana Champaign,
duggira3@illinois.edu, vmahesh@illinois.edu
[2] Dept. of Electrical & Computer Engineering, Univ. of Illinois at Urbana Champaign,
lewang2@illinois.edu, mitras@illinois.edu

Abstract. We present an algorithm for checking temporal precedence properties of nonlinear switched systems. This class of properties subsume bounded safety and capture requirements about visiting a sequence of predicates within given time intervals. Our algorithm handles nonlinear predicates that arise from dynamics-based predictions used in alerting protocols for state-of-the-art transportation systems. It is sound and relatively complete for nonlinear switch systems that robustly satisfy the given property. The algorithm is implemented in our Compare Execute Check Engine (C2E2) using validated simulations. As a case study, we consider NASA's Adjacent Landing Alerting System (ALAS), which is an alerting protocol for closely spaced parallel runways. Using our approach, we study the performance of the ALAS protocol with respect to false and missed alerts for different operating conditions such as initial velocities, bank angles, initial longitudinal separation, and runway configurations.

1 Introduction

Dynamic analysis presents a scalable alternative to static analysis for models with nonlinear dynamics. The basic procedure for dynamic safety verification has three building blocks: (a) a simulation engine, (b) a generalization or bloating procedure, and (c) a satisfiability checker. The simulation engine generates validated simulations of the model with some rigorous error bounds. The generalization procedure uses additional model information to over-approximate bounded-time reach sets from the simulations. This additional model information could be, for example, statically computed Lipschitz constants [13], contraction metrics [8] or more general designer-provided annotations [5]. Finally, the approximation is checked by a satisfiability procedure for inferring safety or for iteratively refining its precision. With these three pieces it is possible to design sound and relatively complete algorithms for bounded time safety verification that also scale to moderately high-dimensional models [5].

In this paper, we extend the reach of the above procedure in two significant ways. First, our new algorithm verifies *temporal precedence properties* which generalize bounded safety. A model $\mathcal{A}$ satisfies temporal precedence $P_1 \prec_b P_2$ if along every trajectory of $\mathcal{A}$, for any time at which the predicate P_2 holds, there

exists an instant of time, at least b time units sooner, where the predicate P_1 must hold. The key subroutine in the new verification algorithm uses the above simulation-based reach set approximation procedure for estimating the time intervals over which the predicates P_1 and P_2 may or must hold. These estimates are constructed so that the algorithm is sound. Moreover, we show that the algorithm is guaranteed to terminate whenever $\mathcal{A}$ satisfies the property robustly (relatively complete). That is, not only does every trajectory ξ satisfy $P_1 \prec_b P_2$, but any small time-shifts and value perturbations of ξ also satisfy $P_1 \prec_b P_2$. Such relative completeness guarantees usually have the most precision that one can hope for in any formal analysis of models involving physical quantities.

Secondly, we develop a new approach to checking satisfiability of nonlinear guarantee predicates [9]. If P_1 and P_2 in the above type of temporal precedence property are in propositional logic or uses linear arithmetic, then existing solvers can efficiently check whether a set of states satisfy them. On the other hand, if they are written as $\exists t > 0, f_p(x,t) > 0$, where f_p is a nonlinear real-valued function then the options are limited. Quantifier elimination is an expensive option, but even that is feasible only if f_p has a closed form definition of a special form (such as polynomial functions). If f_p is implicitly defined as the solution of a set of ODEs (with no analytical solution) then quantifier elimination is impossible. We provide a sound and relatively complete procedure for checking bounded time guarantee predicates using simulation-based over-approximations of $f_p(x,t)$.

These extensions allow us to handle an interesting and difficult verification problem arising from a parallel landing protocol. The Simplified Aircraft-based Paired Approach (SAPA) [6] is an advanced operational concept that enables dependent approaches in closely spaced parallel runways. In the presence of blundering aircraft, the SAPA procedure relies on an alerting algorithm called Adjacent Landing Alerting System (ALAS) [12]. ALAS uses linear and nonlinear projections of the current aircraft positions, velocity vectors, and bank angles to detect possible conflicts between the landing aircrafts. Preliminary studies have shown that the SAPA/ALAS protocol may be unsafe for some parallel runway geometries. Given the nonlinear characteristics of the ALAS logic, finding operating conditions under which the SAPA/ALAS protocol satisfy safety properties is a challenging problem.

This papers models the SAPA/ALAS protocol as a switched system and applies the verification algorithms to formally check properties of the alerting algorithm for various configurations. We verify the property that ALAS issues an alert at least b seconds before an unsafe scenario is encountered. This is modeled as temporal precedence property $Alert \prec_b Unsafe$. Since ALAS uses projections of nonlinear dynamics to issue an alert, we apply the algorithm for inferring guarantee predicates. We implemented these algorithms in C2E2 and discovered operating conditions with the possibilities of false and missed alerts.

Related Work. There are several MATLAB based tools for analyzing properties of switched systems using simulations. Breach [4] uses sensitivity analysis for analyzing STL properties of systems using simulations. This analysis

is sound and relatively complete for linear systems, but does not provide formal guarantees for nonlinear systems. S-Taliro [11] is a falsification engine that search for counterexamples using Monte-Carlo techniques and hence provides only probabilistic guarantees. STRONG [3] uses robustness analysis for coverage of all executions from a given initial set by constructing bisimulation functions. Currently this tool computes bisimulation functions for only linear or affine hybrid systems and does not handle nonlinear systems.

2 System Models and Properties

For a vector v in $\mathbb{R}^n$, $|v|$ stands for ℓ^2-norm. Given intervals I,I' over reals, we say that $I < I'$ iff $\forall v \in I, \forall v' \in I', v < v'$. For a real number b, $I - b = \{v - b \mid v \in I\}$. Subtraction operation over intervals is defined as, $I - I' = \{v - v' \mid v \in I, v' \in I'\}$. $I \times I' = \{v \times v' \mid v \in I, v' \in I'\}$. For $\delta \in \mathbb{R}_{\geq 0}$ and $x \in \mathbb{R}^n$, $B_\delta(x) \subseteq \mathbb{R}^n$ is the closed ball with radius δ centered at x. For a set $S \subseteq \mathbb{R}^n$, $B_\delta(S) = \cup_{x \in S} B_\delta(v)$. For any function $V : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$, given a $\delta > 0$, $B^V_\delta(x) = \{y \mid V(x, y) \leq \delta\}$. For a set $S \subseteq \mathbb{R}^n$, $B^V_\delta(S) = \cup_{x \in S} B^V_\delta(x)$. For a bounded set A, $dia(A) = \sup_{x,y \in A} |x - y|$ denotes the diameter of A.

A real-valued function $\alpha : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$ is called a *class $\mathcal{K}$ function* if $\alpha(0) = 0$ and α is strictly increasing. It is a *class $\mathcal{K}_\infty$ function* if additionally $\alpha(x) \to \infty$ as $x \to \infty$. For a function $h : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and a positive real $\delta > 0$, the δ-left shift of h is the function $h_\delta : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ defined as $h_\delta(t) = h(t + \delta)$ for any $t \in \mathbb{R}_{\geq 0}$. A *$\delta$-perturbation* of h is any function $g : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ such that for all t, $|g(t) - h(t)| < \delta$. A càdlàg function is a function which is *continuous from the right* and *has a limit from the left* for every element in its domain.

2.1 The Switched System Model

We will use the *switch system* formalism [7] for modeling continuous systems. The evolution of an n dimensional switched system is specified by a collection of ordinary differential equations (ODEs) also called as *modes* or *locations* indexed by a set $\mathcal{I}$ and a *switching signal* that specifies which ODE is active at a given point in time. Fixing a switching signal and an initial state, the system is deterministic. Its behavior is the continuous, piece-wise differentiable function of time obtained by pasting together the solutions of the relevant ODEs. We fix $\mathcal{I}$ as the set of modes and n as the dimension of the system with $\mathbb{R}^n$ as state space for the rest of the document.

Definition 1. *Given the set of modes $\mathcal{I}$ and the dimension n, a* switched system $\mathcal{A}$ *is specified by the tuple $\langle \Theta, \mathcal{F}, \Sigma \rangle$, with*

(i) $\Theta \subseteq \mathbb{R}^n$, *a compact set of* initial states,
(ii) $\mathcal{F} = \{f_i : \mathbb{R}^n \to \mathbb{R}^n\}_{i \in \mathcal{I}}$, *an indexed collection of continuous, locally Lipschitz functions, and*
(iii) Σ, *a set of switching signals, where each $\sigma \in \Sigma$ is a càdlàg function $\sigma : \mathbb{R}_{\geq 0} \to \mathcal{I}$.*

The semantics of $\mathcal{A}$ is defined in terms of its solutions or *trajectories*. For a given initial state $x_0 \in \Theta$ and a switching signal $\sigma \in \Sigma$, the solution or the *trajectory* of the switched system is a function $\xi_{x_0,\sigma} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, such that: $\xi_{x_0,\sigma}(0) = x_0$, and for any $t > 0$ it satisfies the differential equation:

$$\dot{\xi}_{x_0,\sigma}(t) = f_{\sigma(t)}(\xi_{x_0,\sigma}(t)). \tag{1}$$

When clear from context, we drop the x_0 and σ subscripts from ξ. Under the stated locally Lipschitz assumption of the f_i's and the càdlàg assumption on σ, it is well-known that Equation (1) has a unique solution and that indeed the trajectory ξ is a well-defined function.

Example. A simple switched system model of a thermostat has two modes $\mathcal{I} = \{on, off\}$ and a single continuous dimension with initial value, say $x = 62$. The continuous dynamics is defined by the linear ODEs $\dot{x} = -kx$ for *off* and $\dot{x} = h - kx$ for on, where k and h are parameters of the thermostat. Thus, $f_{on}(x) = -kx$ and $f_{off}(x) = h - kx$. For a particular switching signal σ, the solution $\xi_{x_0,\sigma}$ is shown in Figure 1.

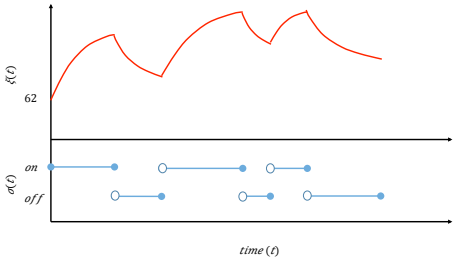


Fig. 1. A switching signal and trajectory of thermostat model.

We are interested in verifying the system for a set of switching signals Σ with bounded time and finite number of mode switches. A bounded time switching signal can be represented as a sequence $\sigma = m_0, m_1, \ldots, m_k$ where each m_i is a pair in $\mathcal{I} \times \mathbb{R}_+$, with the two components denoted by $m_i.mode$ and $m_i.time$. The sequence define $\sigma(t) = m_i.mode$ for all $t \in [\sum_{j=0}^{i-1} m_j.time, \sum_{j=0}^{i} m_j.time)$. We consider a set of switching signals Σ that are represented as a switching interval sequence $S = q_0, q_1, \ldots q_k$, where each q_j is a pair with $q_j.mode \in \mathcal{I}$ and $q_j.range$ is an open interval in $\mathbb{R}_{\geq 0}$. Given a switching interval sequence S, the set $sig(S)$ denotes the set of switching signals $\sigma = m_0, m_1, \ldots, m_k$, such that $m_j.mode = q_j.mode$ and $m_j.time \in q_j.range$. By abuse of notation, we will use interchangeably a set of switching signals Σ and its finite representation S with $sig(S) = \Sigma$. We denote by $width(S)$ the size of the largest interval $q_i.range$. The refinement operation of Σ, denoted as $refine(S)$, gives a finite set of switching interval sequences $\mathcal{S}$ such that $\bigcup_{S' \in \mathcal{S}} sig(S') = sig(S)$ and for each $S' \in \mathcal{S}$, $width(S') \leq width(S)/2$.

2.2 Temporal Precedence with Guarantee Predicates

A *predicate* for the switched system $\mathcal{A}$ is a computable function $P : \mathbb{R}^n \to \{\top, \bot\}$ that maps each state in $\mathbb{R}^n$ to either $\top$ (true) or $\bot$ (false). The predicate is said to be satisfied by a state $x \in \mathbb{R}^n$ if $P(x) = \top$. A *guarantee predicate* [9] $P(x)$ is a predicate of the form $\exists t > 0, f_p(x, t) > 0$, where $f_p : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is called a *lookahead function*. A guarantee predicate holds at a state x if there exists some

future time t at which $f_p(x,t) > 0$ holds. Using a quantifier elimination procedure, one could reduce a guarantee predicate to an ordinary predicate without the existential quantifier. However, this is an expensive operation, and more importantly, it is only feasible for restricted classes of real-valued lookahead functions with explicit closed form definitions. We present in Section 3.2 a technique to handle guarantee predicates with lookahead functions as solutions to nonlinear ODE. As we will see in Section 4, such lookaheads are particularly useful in designing alert mechanisms such as ALAS.

Temporal precedence properties are a class of properties specified by a pair of predicates that must hold for any behavior of the system with some minimum time gap between them. More precisely, a temporal precedence property ϕ is written as $\phi = P_1 \prec_b P_2$, where P_1 and P_2 are (possibly guarantee) predicates and b is a positive real number. The property $\phi = P_1 \prec_b P_2$ is satisfied by a particular trajectory ξ of $\mathcal{A}$ iff

$$\forall t_2 > 0, \text{if } P_2(\xi(t_2)) \text{ then } \exists t_1, 0 < t_1 < t_2 - b, P_1(\xi(t_1)). \tag{2}$$

In other words, along ξ, predicate P_1 should be should be satisfied at least b time units *before* any instance of P_2 is satisfied. We say that $\mathcal{A}$ satisfies ϕ, if every trajectory of $\mathcal{A}$ satisfies ϕ. With a collection of precedence properties, we can state requirements about ordering of some predicates before others.

The property ϕ is said to be *robustly satisfied* by a system if $\exists \tau > 0, \delta > 0$ such that all $\tau' < \tau$ left shifts and all δ-perturbations of all trajectories ξ satisfy the property. An execution ξ is said to *robustly violate* a precedence property $P_1 \prec_b P_2$ if there is a time instant t_2 such that $P_2(\xi(t_2))$ holds, and for some $\delta > 0$, all δ-perturbations ξ' of ξ and $t_1 \in (0, t_2 - b)$, P_1 does not hold in ξ' at time t_1. We will say that a system robustly violates $\phi = P_1 \prec_b P_2$ if some execution ξ (from an initial state) robustly violates ϕ.

3 Simulation-based Verification of Temporal Precedence

In this section, we present an algorithm for verifying temporal precedence properties of switched systems and establish its correctness. Similar to the simulation-based safety verification algorithm presented in our earlier work [5], this algorithm has three components: (a) it uses validated simulations for the dynamics in $\mathcal{F}$, (b) it requires model annotations called discrepancy functions for the dynamics in in $\mathcal{F}$. Finally, (c) it requires a procedure for checking satisfiability of nonlinear guarantee predicates arising from solutions of differential equations. We first introduce the two main building blocks, simulations, and discrepancy functions and then provide the algorithm for verifying temporal properties.

3.1 Simulations and Annotations

For a given initial state x_0 and an ODE $\dot{x} = f(x,t)$ which admits a solution ξ, a fixed time-step numerical integrator produces a sequence of sample points

e.g., $x_1, x_2, \ldots, x_l \in \mathbb{R}^n$ that approximate the trajectory ξ_{x_0} at a sequence of time points, say $\xi_{x_0}(h), \xi_{x_0}(2h), \ldots, \xi_{x_0}(l \times h)$. However, these simulations do not provide any rigorous guarantees about the errors incurred during numerical approximations. Rigorous error bounds on these simulations, which can be made arbitrarily small, are required for performing formal analysis. We present one such notion of a simulation for an ODE as follows:

Definition 2. *Consider an ODE $\dot{x} = f(x,t)$. Given an initial state, x_0, a time bound $T > 0$, error bound $\epsilon > 0$, and a time step $\tau > 0$, an (x_0, T, ϵ, τ)-simulation trace is a finite sequence $(R_1, [t_0, t_1]), (R_2, [t_1, t_2]), \ldots, (R_l, [t_{l-1}, t_l])$ where each $R_j \subseteq \mathbb{R}^n$, and $t_j \in \mathbb{R}_{\geq 0}$ such that $\forall j, 1 \leq j \leq l$*

(1) $t_{j-1} < t_j, t_j - t_{j-1} \leq \tau, t_0 = 0$, and $t_l = T$,
(2) $\forall t \in [t_{j-1}, t_j], \xi_{x_0}(t) \in R_j$, and
(3) $dia(R_j) \leq \epsilon$.

Numerical ODE solvers such as CAPD [1] and VNODE-LP [10] can be used to generate such simulations for arbitrary values of τ and ϵ using Taylor Models and interval arithmetic. We now define the next building block, a type of model annotation for ODEs called *discrepancy function*.

Definition 3. *A smooth function $V : \mathbb{R}^{2n} \to \mathbb{R}_{\geq 0}$ is called a* discrepancy function *for an ODE $\dot{x} = f(x,t)$, if and only if there are functions $\underline{\alpha}, \overline{\alpha} \in \mathcal{K}_\infty$ and a uniformly continuous function $\beta : \mathbb{R}^{2n} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ with $\beta(x_1, x_2, t) \to 0$ as $|x_1 - x_2| \to 0$ such that for any pair of states $x_1, x_2 \in \mathbb{R}^n$:*

$$\underline{\alpha}(|x_1 - x_2|) \leq V(x_1, x_2) \leq \overline{\alpha}(|x_1 - x_2|) \text{ and} \tag{3}$$
$$\forall\, t > 0.\ V(\xi_{x_1}(t), \xi_{x_2}(t)) \leq \beta(x_1, x_2, t), \tag{4}$$

where ξ denotes the solution of the differential equation. A tuple $(\underline{\alpha}, \overline{\alpha}, \beta)$ satisfying the above conditions is called a witness *to the discrepancy function.*

The discrepancy function provides an upper bound on the distance between two trajectories starting from different initial states x_1 and x_2. This upper bound, together with a simulation, is used to compute an overapproximation of the set of all reachable states of the system from a neighborhood of the simulation. For linear and affine dynamics such discrepancy functions can be computed by solving semidefinite programs [5]. In [5], we identified classes of nonlinear ODEs for which Lipschitz constants, contraction metrics, and incremental Lyapunov functions can be computed which are all special instances of Definition 3. For the switched systems $\mathcal{A}$ with a set of differential equations $\mathcal{F} = \{f_i\}_{i \in \mathcal{I}}$, a discrepancy function for each f_i (namely, V_i and its witness $(\underline{\alpha}_i, \overline{\alpha}_i, \beta_i)$) is required. Using discrepancy function and validated simulations as building blocks, we compute a bounded overapproximation of the reachable set for initial set Θ, set of switching signals S, and time step τ. We present one such notion of $ReachTube$ as follows:

Definition 4. *Given an initial set of states Θ, switching interval sequence S, dynamics $\mathcal{F}$, time step $\tau > 0$, and error bound $\epsilon > 0$, a $(\Theta, S, \epsilon, \tau)$-$ReachTube$ is a sequence $\psi = (O_1, [t_0, t_1]), (O_2, [t_1, t_2]), \ldots, (O_l, [t_{l-1}, t_l])$ where O_j is a set of pairs (R, h) such that $R \subseteq \mathbb{R}^n$, and $h \in \mathcal{I}$, such that, $\forall j, 1 \leq j \leq l$*

(1) $t_{j-1} < t_j, t_j - t_{j-1} \leq \tau, t_0 = 0$,
(2) $\forall x_0 \in \Theta, \forall \sigma \in sig(S), \forall t \in [t_{j-1}, t_j], \exists (R, h) \in O_j$, *such that,* $\xi_{x_0,\sigma}(t) \in R, \sigma(t) = h$,
(3) $\forall (R, h) \in O_j, dia(R) \leq \epsilon$, *and*
(4) each mode in $\mathcal{I}$ occurs at most once in O_j.

Intuitively, for every given time interval $[t_{j-1}, t_j]$, the set O_j contains an (R, h) pair such that R overapproximates the reachable set for the mode h in the given interval duration. In our previous work on verification using simulations [5], we presented an algorithm that computes overapproximation of the reachable set using sampled executions and annotations. We outline the procedure ComputeReachTube$(\Theta, S, \delta, \epsilon', \tau)$ that takes as input the initial set Θ, switching signals S, partitioning parameter δ, simulation error ϵ' and time step τ, to compute the $ReachTube$ ψ and error ϵ such that ψ is a $(\Theta, S, \epsilon, \tau)$-$ReachTube$.

1. Assign to Q, the set of initial states Θ.
2. For each q_i in the switching interval sequence $S = q_0, q_1, \ldots, q_k$.
3. Compute $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$, a δ-partitioning of Q, such that $Q \subseteq \cup B_\delta(x_i)$.
4. Generate a validated simulation (Definition 2) η for every state $x \in \mathcal{X}$ with error ϵ', time step τ. Then, compute the $ReachTube$ for $B_\delta(x_0)$ by bloating η as $B_\epsilon^{V_{q_i.mode}}(\eta)$, where $\epsilon = sup\{\beta_{q_i.mode}(y, x, t) | y \in B_\delta(x)\}$.
5. Compute the union of each of the $ReachTubes$ for $B_\delta(x_0)$ as the $ReachTube$ for mode $q_i.mode$.
6. Compute the initial set for the next mode by taking the projection of $ReachTube$ for $q_i.mode$ over the interval $q_i.range$ as Q. Repeat steps 3 - 6 for q_{i+1}.

The order of overapproximation of the $ReachTube$ computed using the procedure described above is the *maximum bloating* performed using the annotation $V_{q_i.mode}$ and $\beta_{q_i.mode}$ for all the modes in S. This overapproximation and the error in simulation gives us the value of ϵ such that ψ is a $(\Theta, S, \epsilon, \tau)$-$ReachTube$. The nondeterminism during the switching times from one mode to another enables the reachable set to be in two different modes at a given instance of time, which is reflected in O_j. From [5], we get the following proposition.

Proposition 1. *Given an initial set Θ, switching signals S, partitioning parameter δ, simulation error ϵ' and time step τ, let the tuple $\langle \psi, \epsilon \rangle =$ ComputeReachTube$(\Theta, S, \delta, \epsilon', \tau)$. Then, as $dia(\Theta) \to 0$, $width(S) \to 0$, $\delta \to 0$, $\epsilon' \to 0$, and $\tau \to 0$, we have $\epsilon \to 0$.*

3.2 Temporal Precedence Verification Algorithm

CheckRefine (see Figure 2) performs the following steps iteratively: (1) Create an initial partition of the set of start states Θ. (2) Compute the $ReachTubes$ for each these partitions as given in Definition 4. (3) Check the temporal precedence property for the $ReachTube$. (4) Refine the partitioning if the above check is inconclusive, and repeat steps (2)-(4).

A key step in the verification procedure is to check, whether a given $ReachTube$ satisfies a temporal precedence property. We define collection of intervals $mustInt$, $notInt$, and $mayInt$ for a given predicate P and $ReachTube$ ψ which will help us in the verification of temporal precedence property.

Definition 5. *Given a* $ReachTube$ $\psi = (O_1, [t_0, t_1]), \ldots, (O_l, [t_{l-1}, t_l])$ *and a predicate* P*, we define sets of intervals* $mustInt(P, \psi)$*,* $notInt(P, \psi)$*, and* $mayInt(P, \psi)$ *such that* $\forall j > 0$

$$[t_{j-1}, t_j] \in mustInt(P, \psi) \text{ iff } \forall (R, h) \in O_j, R \subseteq P.$$
$$[t_{j-1}, t_j] \in notInt(P, \psi) \text{ iff } \forall (R, h) \in O_j, R \subseteq P^c.$$
$$[t_{j-1}, t_j] \in mayInt(P, \psi) \text{ otherwise.}$$

Definition 5 classifies an interval $[t_{j-1}, t_j]$ as an element of $mustInt(P, \psi)$ only if the overapproximation of the reachable set for that interval is contained in P. Similar is the case with $notInt(P, \psi)$. However if the overapproximation of the reachable set cannot conclude either of the cases, then the interval is classified as $mayInt(P, \psi)$. There are two possible reasons for this: first, the order of overapproximation is too coarse to prove containment in either P or P^c; second, the execution moves from the states satisfying P to states not satisfying P during that interval. Thus, better estimates of $mustInt$, $notInt$ and $mayInt$ can be obtained by improving the accuracy of $ReachTube$ ψ. We now proceed to define the conditions when a $ReachTube$ *satisfies* or *violates* a temporal precedence property $P_1 \prec_b P_2$.

Remark 1: To compute $mustInt$, $mayInt$, and $notInt$ as defined in Definition 5, we need to be able to check if $R \subseteq P$ or $R \subseteq P^c$. However, for guarantee predicates with lookahead functions that use the solutions of ODEs, it is unclear how to perform these checks. In Section 3.3, we will describe a simulation-based method to address this challenge. The algorithm in Section 3.3 will, in fact, provide weaker guarantees. Assuming P is an open set, our algorithm will answer correctly when $R \subseteq P$ and when for some $\delta > 0$, $B_\delta(R) \subseteq P^c$; in other cases, our algorithm may not terminate. Such weaker guarantees will turn out to be sufficient for our purposes.

Definition 6. *Given* $ReachTube$ ψ*, and a temporal precedence property* $P_1 \prec_b P_2$*,* ψ *is said to* satisfy *the property iff for any interval* I'*,* $I' \in mustInt(P_2, \psi) \cup mayInt(P_2, \psi)$*, exists interval* I*,* $I \in mustInt(P_1, \psi)$ *such that* $I < I' - b$*. Also,* ψ *is said to* violate *the property if* $\exists I' \in mustInt(P_2, \psi)$ *such that,* $\forall I \in mustInt(P_1, \psi) \cup mayInt(P_1, \psi)$*,* $I' - b < I$*.*

From Definition 6 it is clear that if a $ReachTube$ ψ satisfies a temporal precedence property, then for all the trajectories corresponding to the $ReachTube$, the predicate P_1 is satisfied at least b time units before P_2. Also, if the $ReachTube$ violates the property, then it is clear that there exists at least one trajectory such that for an instance of time, i.e., in $I' \in mustInt(P_2, \psi)$ at all the time instances at least b units before, the predicate P_1 is not satisfied. In all other cases, the $ReachTube$ cannot infer whether the property is satisfied or violated. As this

inference depends on the accuracy of $mustInt$, $notInt$ and $mayInt$. More accurate $ReachTubes$ produce better estimates of these intervals and hence help in better inference of temporal precedence property.

Given a system $\mathcal{A}$ and property $P_1 \prec_b P_2$, one can compute the $ReachTube$ for the system and apply Definition 6 to check whether the system satisfies the temporal precedence property. This is however not guaranteed to be useful as the approximation of $ReachTube$ computed might be too coarse. In CheckRefine, we give an algorithm that at each iteration refines the inputs to compute more precise $ReachTubes$. Proposition 1 guarantees that these $ReachTubes$ can be made arbitrarily precise.

1: **Input:** $\mathcal{A} = \langle \Theta, \mathcal{F}, \Sigma \rangle$, $\{V_i, (\underline{\alpha}_i, \overline{\alpha}_i, \beta_i)\}_{i \in \mathcal{I}}$, $P_1 \prec_b P_2$, δ_0, δ'_0, ϵ'_0, τ_0.
2: $Q \leftarrow \Theta$; $\Omega \leftarrow \{\Sigma\}$; $\delta \leftarrow \delta_0$; $\delta' \leftarrow \delta'_0$; $\epsilon' \leftarrow \epsilon'_0$; $\tau \leftarrow \tau_0$
3: **while** $Q \neq \emptyset$ **do**
4: $\quad \mathcal{X} \leftarrow \delta\text{-}partition(Q)$;
5: $\quad$ **for all** $x_0 \in \mathcal{X}$ **do**
6: $\quad\quad$ **for all** $S \in \Omega$ **do**
7: $\quad\quad\quad \langle \psi, \epsilon \rangle = \mathsf{ComputeReachTube}(B_\delta(x_0), S, \delta', \epsilon', \tau)$
8: $\quad\quad\quad$ **if** ψ satisfies $P_1 \prec_b P_2$ **then continue**;
9: $\quad\quad\quad$ **else if** ψ falsifies $P_1 \prec_b P_2$ **return** "Property $P_1 \prec_b P_2$ is violated"
10: $\quad\quad\quad$ **else**
11: $\quad\quad\quad\quad \Omega \leftarrow \Omega \setminus \{S\} \cup refine(S)$; $\delta \leftarrow \delta/2$; $\delta' \leftarrow \delta'/2$, $\epsilon' \leftarrow \epsilon'/2$; $\tau \leftarrow \tau/2$;
12: $\quad\quad\quad\quad$ **goto** Line 3
13: $\quad\quad\quad$ **end if**
14: $\quad\quad$ **end for**
15: $\quad\quad Q \leftarrow Q \setminus B_\delta(x_0)$
16: $\quad$ **end for**
17: **end while**
18: **return** "Property $P_1 \prec_b P_2$ is satisfied".

Fig. 2. Algorithm CheckRefine: Partitioning and refinement algorithm for verification of temporal precedence properties.

The algorithm (in Figure 2) first partitions the initial set into δ-neighborhoods (line 3) and compute $ReachTubes$ for every switching interval sequence in Ω (line 7). If all these $ReachTubes$ (that is all the executions from neighborhood) satisfy the property, then the neighborhood is removed from Q. Similarly, CheckRefine returns that the property is violated only when $ReachTube$ violates the property. If neither can be inferred, then the parameters to function ComputeReachTube are refined in line 11 to increase their precision. Since this operation is iteratively performed to obtain arbitrarily precise $ReachTubes$, Soundness and Relative completeness follow from Definition 6 and Proposition 1.

Theorem 1 (Soundness and Relative Completeness). *Algorithm* CheckRefine *is sound, i.e., if it returns that the system satisfies the property, then the property is indeed satisfied. If it returns that the property is violated, then the property is indeed violated by the system. Further, if we assume that predicates P_1 and P_2 are open sets, and there is a procedure that correctly determines if for a set R, $R \subseteq P_i$ (for $i = 1, 2$) or if there is $\delta > 0$ such that $B_\delta(R) \subseteq P_i^c$ (for $i = 1, 2$). Then, if the system $\mathcal{A}$ satisfies $P_1 \prec_b P_2$ or robustly violates $P_1 \prec_b P_2$ then* CheckRefine *terminates with the right answer.*

3.3 Verification of Guarantee Predicates

As discussed in the Section 2.2, guarantee predicates are of the form $P(x) = \exists t > 0, f_p(x,t) > 0$, where f_p is called a lookahead function. In this section, we present an algorithm for time bounded verification of such predicates of the special form $P(x) = \exists 0 < t < T_l, w_p(\xi'_x(t)) > 0$, where w_p is a continuous function and ξ' is solution of ODE $\dot{y} = g(y,t)$. As specified in Remark 1, we give a decision procedure to check $R \subseteq P$ or an open cover of R is contained in P^c. This algorithm similar to CheckRefine computes successively better approximations for the $ReachTube$ and checks whether the predicate $P' \equiv w_p(x) > 0$ is satisfied by the reach tube. This is done by calculating $mustInt(P', \psi)$ and $mayInt(P', \psi)$ as defined in Definition 5. If the $mustInt$ is non-empty, then it implies that the predicate P is satisfied by the $ReachTube$ and hence $R \subseteq P$. If both the $mayInt$ and $mustInt$ are empty sets, then, clearly the predicate P is not satisfied in the bounded time T_l by any state in R, and hence an open cover of R is contained in P^c. Soundness and Relative Completeness of CheckGuarantee follow from CheckRefine(proofs in full version [3].).

Input: R, $\dot{y} = g(y,t)$, S', $V_g(x_1,x_2)$, $(\underline{\alpha}_g, \overline{\alpha}_g, \beta_g)$ w_p, δ, τ, T_l
while $R \neq \emptyset$ **do**
  $\mathcal{X} \leftarrow \delta\text{-}partition(R)$;
  for all $x_0 \in \mathcal{X}$ **do**
    $\langle \psi, \epsilon \rangle = \mathsf{ComputeReachTube}(B_\delta(x_0), S', \delta, \delta, \tau)$;
    if $mustInt(w_p, \psi) \neq \emptyset$ **then** $R \leftarrow R \setminus B_\delta(x_0)$
    else if $mustInt(w_p, \psi) \cup mayInt(w_p, \psi) = \emptyset$ **then return** "UNSAT"
    end if
  end for
  $\delta \leftarrow \delta/2$; $\tau \leftarrow \tau/2$;
end while
return "SAT".

Fig. 3. Algorithm CheckGuarantee: Decides whether a lookahead predicate is satisfied in a given set R

4 Case Study: A Parallel Landing Protocol

The Simplified Aircraft-based Paired Approach (SAPA) is an advanced operational concept proposed by the US Federal Aviation Administration (FAA) [6]. The SAPA concept supports dependent, low-visibility parallel approach operations to runways with lateral spacing closer than 2500 ft. A Monte-Carlo study conducted by NASA has concluded that the basic SAPA concept is technically and operationally feasible [6]. In the presence of a blundering aircraft, i.e., unacceptable crossing of paired aircraft paths, SAPA relies on an alerting mechanism to avoid aircraft blunders.

NASA's Adjacent Landing Alerting System (ALAS) is an alerting algorithm for the SAPA concept [12]. ALAS is a pair-wise state-based algorithm, i.e., ALAS detects possible conflicts between two aircraft by checking if the linear and curved projected trajectories of the current state violate predefined separation

[3] https://wiki.cites.illinois.edu/wiki/display/MitraResearch/Verification+of+a+Parallel+Landing+Protocol

minima. The two aircraft are referred to as *ownship* and *intruder*. When the ALAS algorithm is deployed in an aircraft following the SAPA procedure, the aircraft considers itself to be the ownship, while any other aircraft is considered to be an intruder.

A formal static analysis of the ALAS algorithm is challenging due to the complexity of the SAPA/ALAS protocol and the large set of configurable parameters that enable different alerting thresholds, aircraft performances, and runway geometries. The dynamic analysis presented in this paper is well suited for finding operating conditions under which the SAPA procedure with the ALAS algorithm performs well with respect to false and missed alerts.

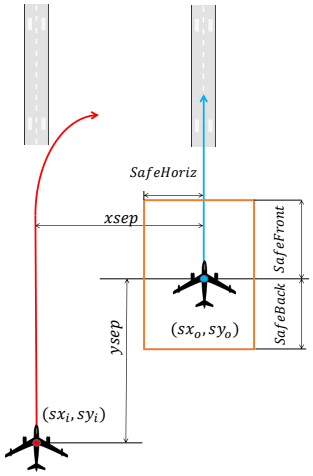


Fig. 4. Possible blundering scenario during parallel approach of aircraft. Intruder (red) & ownship (blue).

4.1 Aircraft Dynamics

For the analysis of the SAPA/ALAS protocol, this paper considers a blundering scenario where the intruder aircraft turns towards the ownship during the landing approach as shown in Figure 4. We model the dynamics of the aircraft as a switched system with continuous variables sx_i, sy_i, vx_i, vy_i and sx_o, sy_o, vx_o, and vy_o representing the position and velocity of intruder and ownship respectively. The switching system has two modes: *approach* and *turn*. The mode *approach* represents the phase when both aircraft are heading towards the runway with constant speed. The differential equation for this mode is

$$\begin{bmatrix} \dot{sx_o} \\ \dot{sy_o} \\ \dot{vx_o} \\ \dot{vy_o} \end{bmatrix} = \begin{bmatrix} 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{bmatrix} \begin{bmatrix} sx_o \\ sy_o \\ vx_o \\ vy_o \end{bmatrix} \text{ and } \begin{bmatrix} \dot{sx_i} \\ \dot{sy_i} \\ \dot{vx_i} \\ \dot{vy_i} \end{bmatrix} = \begin{bmatrix} 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{bmatrix} \begin{bmatrix} sx_i \\ sy_i \\ vx_i \\ vy_i \end{bmatrix}. \tag{5}$$

The mode *turn* represents the blundering trajectory of intruder. In this mode, the intruder banks at an angle ϕ_i to turn away from the runway towards the ownship. The switching signal determines the time of transition from *approach* to *turn*. In this mode, the differential equation of the ownship remains the same as that of *approach*, but the intruder's turning motion with banking angle ϕ_i is

$$\begin{bmatrix} \dot{sx_i} \\ \dot{sy_i} \\ \dot{vx_i} \\ \dot{vy_i} \end{bmatrix} = \begin{bmatrix} 0\,0 & 1 & 0 \\ 0\,0 & 0 & 1 \\ 0\,0 & 0 & \omega_i \\ 0\,0 & -\omega_i & 0 \end{bmatrix} \begin{bmatrix} sx_i \\ sy_i \\ vx_i \\ vy_i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega_i - c_y \\ \omega_i + c_x \end{bmatrix}, \tag{6}$$

where c_x and c_y are constant functions of the initial states of the ownship and intruder, and ω_i is the angular speed of intruder. Given the bank angle ϕ_i, the angular speed is given by $w_i = \frac{G|\tan(\phi_i)|}{\sqrt{{vx_i}^2+{vy_i}^2}}$, where G is the gravitational constant. The upper bound on the bank angle ϕ_i is denoted as ϕ_{max}.

The system starts in the $approach$ mode with the initial position of the intruder at $sx_i = sy_i = 0$ and the ownship at $sx_o = xsep$ and $sy_o = ysep$, where $xsep$ denotes the lateral separation between the runways and $ysep$ denotes the initial longitudinal separation between the aircraft. The initial velocities of both aircraft along the x-axis are 0 and the initial velocities along the y-axis are parameters. The time of switching from $approach$ mode to $turn$ mode is nondeterministically chosen from the interval $T_{switch} = [2.3, 2.8]$. These parameters and the initial values of the variables are constrained by the SAPA procedure [6].

4.2 ALAS Algorithm and Verification of Temporal Precedence Property

ALAS Algorithm: ALAS issues an alert when the aircraft are predicted to violate some distance thresholds called $Front$ and $Back$ specified by ALAS [12]. To predict this violation, the aircraft projects the current state of the system with three different dynamics: first, the intruder does not turn (i.e., banking angle 0°), second, the intruder turns with the specified bank angle ϕ_i and third, the intruder turns with the maximum bank angle ϕ_{max}. If any of these projections violates the distance thresholds, then an alert is issued. We represent the alert predicate for the projections as $Alert_0$, $Alert_{\phi_i}$ and $Alert_{\phi_{max}}$ respectively. Thus the alert predicate is defined as $Alert \equiv Alert_0 \vee Alert_{\phi_i} \vee Alert_{\phi_{max}}$. These are guarantee predicates that check for future violations of distance thresholds.

We now describe $Alarm_\pi$ in detail (π being the bank angle considered). The lookahead function for $Alert_\pi$ is defined as follows: from a given state x, it computes the projected trajectory of the aircraft when intruder turns at bank angle π. If these trajectories intersect, then it computes the times of intersection. That is, it computes t_i, t_o such that $sx'_i(t_i) = sx'_o(t_o)$ and $sy'_i(t_i) = sy'_o(t_o)$, where $sx'_i, sy'_i, sx'_o, sy'_o$ represent the positions of the intruder and ownship aircraft in the projected trajectory. If such t_i and t_o exist, the $Alert_\pi$ is defined as:

$$\begin{aligned} Alert_\pi(x) \equiv \text{ iff } t_i > t_0 \ ? \ & (\Delta t^2 \times (vx_o^2 + vy_o^2) < Back^2) \\ : \ & (\Delta t^2 \times (vx_o^2 + vy_o^2) < Front^2), \end{aligned}$$

where $\Delta t = t_i - t_o$. If such t_i and t_o do not exist, then $Alert_\pi(x) = \bot$. The expression $a \ ? \ b \ : c$ is a short hand for **if**(a) **then** b **else** c.

As the guarantee predicates cannot be handled by SMT solvers, we proposed in Section 3.3, a simulation based algorithm for handling them. In this case study, we apply the proposed technique to resolve the nonlinearities of t_o and t_i in the $Alert_\pi$ predicate. For a given state x, we compute the bounded time (T_l) projected $ReachTubes$ ψ' to compute intervals T_o and T_i such that $t_i \in T_i$ and $t_o \in T_o$. Then, an overapproximation $Alert'_\pi$ of $Alert_\pi$ is computed as: $Alert'_\pi(x) = \top$ if and only if

$$\begin{aligned} T_i > T_0 \ ? \ & (\Delta T^2 \times (vx_o^2 + vy_o^2) < Back^2) \\ : \ & (\Delta T^2 \times (vx_o^2 + vy_o^2) < Front^2) \end{aligned}$$

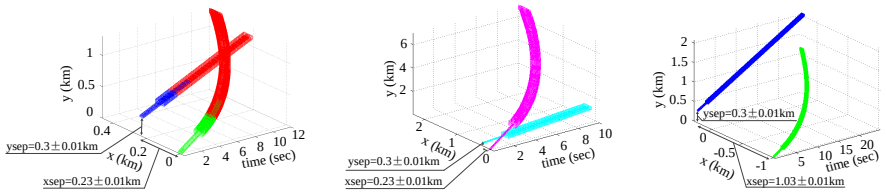
where $\Delta T = T_i - T_o$. The numerical values of T_i and T_o computed simplify the $Alert'_\pi$ predicate and can be handled by SMT solvers.

Safety of Aircraft: A state of the system where the intruder aircraft is inside a safety area surrounding the ownship is said to be *unsafe*. In this paper, we consider a safety area of rectangular shape that is $SafeHoriz$ wide, starts a distance $SafeBack$ behinds the ownship and finishes a distance $SafeFront$ in front of the ownship. The values $SafeHoriz$, $SafeBack$ and $SafeFront$ are to be determined by a safety analysis of the SAPA concept. In this paper, we consider them to be given constant values. Formally, the predicate $Unsafe$ is defined as $Unsafe(x) \equiv (sy_i > sy_o ? sy_i - sy_o < SafeFront : sy_o - sy_i < SafeBack)$ and $|sx_i - sx_o| < SafeHoriz$.

The main correctness property for the ALAS algorithm is that an alert is raised at least b seconds before the intruder violates the safety buffer where b is in the range $[4, 15]$. This can written as a temporal precedence property $Alert \prec_b Unsafe$.

4.3 Verification Scenarios and C2E2 Performance

Tool overview. The verification algorithms of Section 3 are implemented in the tool Compute Execute Check Engine (C2E2). C2E2 accepts Stateflow (SF) charts as inputs, translates them to C++ using CAPD [1] for generating rigorous simulations. For checking sat queries it used Z3 [2] and GLPK [4]. The discrepancy functions for the aircraft dynamics were obtained by computing incremental Lyapunov-like function using MATLAB [5]. The following experiments were performed on Intel Quad Core machine 2.33 GHz with 4GM memory.



(a) Scenario 1. (b) Scenario 2. (c) Scenario 3.

Fig. 5. Figure depicting the set of reachable states of the system. Color coding is used to depict whether the alert is issued by the alerting algorithm

Results. We verify the temporal precedence property $Alert \prec_b Unsafe$. In this section, we check the temporal precedence property for several configurations of the system (i.e., values of parameters and initial values of state variables). For all these experiments, we fix the time bound for verification as 15 sec and the time bound for projection as 25 sec.

Scenario 1. The system configuration is specified by the following parameters and variables: $xsep \in [0.22, 0.24]$ km, $ysep \in [0.2, 04]$ km, $\phi_i = 30°$,

[4] http://www.gnu.org/software/glpk

$\phi_{max} = 45°$, $vy_o = 0.07$ km/sec and $vy_i = 0.08$ km/sec. With this configuration, C2E2 proves that the system satisfies the temporal precedence property $Alert \prec_4 Unsafe$ and an alert is generated 4.38 seconds before the safety is violated. The set of reachable states of the ownship and the intruder when the safety property is violated is shown in *red* and the safe states reached are shown in blue and green respectively in Figure 5(a).

Scenario 2. Increasing the intruder velocity to $vy_i = 0.11$ km/sec, and bank angle $\phi_i = 45°$ from the configuration of Scenario 1 results in Scenario 2. In this case, the safe separation between the intruder and the ownship is always maintained as the intruder completes the turn behind the ownship. Also, the alarm is not raised and hence the property $Alert \prec_b Unsafe$ is satisfied.

Scenario 3. Changing the configuration by $vy_i = 0.11$ km/sec, $xsep \in [1.02, 1.04]$ km, and $\phi_i = 45°$ from Scenario 1 results in Scenario 3. C2E2 proves that ALAS issues a false-alert, i.e. an alert is issued even when the safety of the system is maintained. Though the property $Alert \prec_b Unsafe$ is not violated, avoiding such circumstances improves the efficiency of the protocol and C2E2 can help identify such configurations.

Scenario 4. Placing the intruder in front of ownship i.e., $ysep = -0.3$ km and $vy_i = 0.115$ km/sec from configuration in Scenario 1 results in Scenario 4. C2E2, in this scenario proves that the ALAS algorithm misses an alert, i.e., does not issue an alert before the safety separation is violated. Such scenarios should always be avoided as they might lead to catastrophic situations. This demonstrates that C2E2 can aid in identifying scenarios which should be avoided and help design the safe operational conditions for the protocol.

Scen.	$A \prec_4 U$	time (m:s)	Refs.	$A \prec_t U$
6	False	3:27	5	2.16
7	True	1:13	0	–
8	True	2:21	0	–
6.1	False	7:18	8	1.54
7.1	True	2:34	0	–
8.1	True	4:55	0	–
9	False	2:18	2	1.8
10	False	3:04	3	2.4
9.1	False	4:30	2	1.8
10.1	False	6:11	3	2.4

Table 1. Running times. Columns 2-5: Verification Result, Running time, # of refinements, value of b for which $A \prec_b U$ is satisfied.

Scenario 5. Reducing the $xsep \in [0.15, 0.17]$ km and $ysep \in [0.19, 0.21]$ km from configuration in Scenario 1 gives Scenario 5. For this scenario, C2E2 did not terminate in 30 mins. Recall that the verification algorithm presented in Section 3 is sound and relatively complete only if the system robustly satisfies the property. Thus, we conjecture that the Scenario 5 does not satisfy the property robustly. Upon closer inspection we observe that the partitioning parameter $\delta = 0.0005$ and time step $\tau = 0.001$ (typical values at termination are $\delta = 0.005$ and $\tau = 0.01$), which support our conjecture.

Performance of C2E2. The running time of verification procedure and their outcomes for several other scenarios are presented in Table 1. In Scenarios 6-8, we introduce uncertainty in the initial velocities of the aircraft with all other parameters remaining the same as in Scenario 1. The velocity of the aircraft are changed to be $vy_o \in [0.07, 0.075]$ in scenario 5, $vy_i \in [0.107, 0.117]$ in scenario 6, and $vx_i \in [0.0, 0.005]$ in scenario 7 respectively. Scenarios $x.1$ is sim-

ilar to Scenario x (for x being 6,7,8) however with twice the uncertainty in the velocity. Scenario 9 is obtained by changing the runway separation to be $xsep = 0.5 \pm 0.01$. Scenario 10 is obtained by reducing the $xsep = 0.2 \pm 0.01$. Scenario $x.1$ is similar to Scenario x (for x being 9,10) however with twice the time horizon for verification and projection. We can draw conclusion that the verification time depends on time horizon approximately linearly.

5 Conclusion

In this paper, we presented a dynamic analysis technique that verifies *temporal precedence properties* and an approach to verify guarantee predicates that use solutions of ODEs as lookahead functions. We proved soundness and relative completeness of both these techniques and applied the verification technique on SAPA concept with ALAS alerting system as a case study. The case study demonstrated that our technique can not only verify the properties of the ALAS protocol, but also could identify conditions for false and missed alert which are crucial in designing operational conditions.

Acknowledgement: The authors would like to sincerely thank Dr. César Muñoz from NASA Langley research center for helpful discussions and inputs.

References

1. D. Wilczak, P. Zgliczyski, Computer Assisted Proofs in Dynamic Groups (CAPD). http://capd.ii.uj.edu.pl/
2. L. De Moura and N. Bjørner. Z3: an efficient SMT solver. In TACAS '08.
3. Y. Deng, A. Rajhans, and A. Agung Julius. Strong: A trajectory-based verification toolbox for hybrid systems. In QEST '13.
4. A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In CAV '10.
5. P. S. Duggirala, S. Mitra, and M. Viswanathan. Verification of annotated models from executions. In EMSOFT '13.
6. S. C. Johnson, G. W. Lohr, B. T. McKissick, N. M. Guerreiro, and P. Volk. Simplified aircraft-based paired approach: Concept definition and initial analysis. Technical Report NASA/TP-2013-217994, NASA, Langley Research Center, 2013.
7. D. Liberzon. *Switching in Systems and Control*. Systems and Control: Foundations and Applications. Birkhauser, Boston, June 2003.
8. W. Lohmiller and J. J. E. Slotine. On contraction analysis for non-linear systems. Automatica '98.
9. Z. Manna and A. Pnueli. A hierarchy of temporal properties. In PODC '87.
10. N. Nedialkov. VNODE-LP: Validated solutions for initial value problem for ODEs. Technical report, McMaster University, 2006.
11. T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivancic, A. Gupta, and G.J. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In HSCC '10.
12. R. B. Perry, M. M. Madden, W. T.-Pomales, and Ricky W. Butler. The simplified aircraft-based paired approach with the ALAS alerting algorithm. Technical Report NASA/TM-2013-217804, NASA, Langley Research Center, 2013.
13. G.R. Wood and B.P. Zhang. Estimation of the Lipschitz constant of a function. *Journal of Global Optimization* '96.