Symbolic Reachability Analysis of Lazy Linear Hybrid Automata

Susmit Kumar Jha
Bryan Brady
Sanjit A. Seshia

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-32
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-32.html

February 28, 2007

Symbolic Reachability Analysis of Lazy Linear Hybrid Automata

Susmit Jha, Bryan A. Brady, and Sanjit A. Seshia

EECS Department, UC Berkeley
{jha,bbrady,sseshia}@eecs.berkeley.edu

Abstract. Lazy linear hybrid automata (LLHA) model the discrete time behavior of control systems containing finite-precision sensors and actuators interacting with their environment under bounded inertial delays. In this paper, we present a symbolic technique for reachability analysis of lazy linear hybrid automata. The model permits only linear flow constraints but the invariants and guards can be any computable function. We present an abstraction hierarchy for LLHA. Our verification technique is based on bounded model checking and k-induction for reachability analysis at different levels of the abstraction hierarchy within an abstraction-refinement framework. The counterexamples obtained during BMC are used to construct refinements in each iteration. Our technique is practical and compares favorably with state-of-the-art tools, as demonstrated on examples that include the Air Traffic Alert and Collision Avoidance System (TCAS).

1 Introduction

A hybrid system is a dynamical system which exhibits both discrete and continuous behavior. Hybrid automata [4] have proved to be useful mathematical structures for modeling systems comprising discrete transition systems interacting with continuous dynamical systems. However, it is clear that in any implementation of a hybrid automaton, the state of the dynamical system reported to the discrete controller are digitized with finite precision by sensors, and the output signals of the controller transmitted to its actuators are also of finite precision. Further, the controller can only observe continuous state variables at discrete time points. Hence, it is somewhat unrealistic to assume that the controller can interact with its environment continuously and with infinite precision.

The inherent discrete nature of a controller of a hybrid system has led to recent efforts [15, 2, 3, 1] toward studying the discrete time behavior of hybrid systems. A similar argument in favor of focusing on discrete time behavior is presented by Henzinger and Kopke [11]. *Lazy linear hybrid automata*(LLHA) [2, 3] models the discrete time behavior of hybrid systems having finite precision and bounded delays in actuation and sensing. Further, their definition of LLHA allows nonlinear invariants, guards and resets. However, the discrete behavior in this model depends on the sampling frequency of the controller as well as the precision of variables, and hence, these representations are very large and any

enumerative analysis would not be feasible for systems of appreciable size and complexity.

In this paper, we present a symbolic technique for reachability analysis of *lazy linear hybrid automata*. We make the following novel contributions:

1. On the theoretical side, we present an abstraction hierarchy for LLHA that can be used for reachability analysis within a counterexample-guided abstraction-refinement framework.
2. We give an implementation of a symbolic model checker for LLHA based on bounded model checking and k-induction that operates at any level of abstraction.
3. We demonstrate the scalability of our methods in comparison to other state-of-the-art tools on examples such as Automated Highway Control System (AHS) and the Air Traffic Alert and Collision Avoidance System (TCAS).

2 Related Work

PHAver (Polyhedral Hybrid Automaton Verifyer) [10] is a tool for verifying safety properties of hybrid systems. It uses on-the-fly over-approximation to handle affine flows by iterative partitioning of the state space. It cannot handle non-linear invariants or guards.

The discrete hybrid automata underlying the HYSDEL tool [15] is formed by the connection of a finite state machine with a switched affine system through an interface. Our work is similar to HYSDEL in its considering an inertial interface between the digital and the continuous components of the hybrid system. Unlike our symbolic approach, HYSDEL uses numerical simulation for analysis. Also, our technique allows guards and invariants that use any computable function.

HSolver [16, 8] allows general constraints over variables as invariants and guards. It uses interval arithmetic to check whether trajectories can move over the boundaries in a rectangular grid. Our technique uses SAT-based decision procedures for finite-precision arithmetic to do a symbolic analysis instead of an enumerative analysis. Further, invariants and guards in LLHA can be constraints over variables as well as over rates of the change of variables unlike the models allowed in HSolver. We illustrate the utility of such invariants and guards with a practical example of Air Traffic Alert and Collision Avoidance System (TCAS).

Another closely related tool is HybridSAL [19, 18], which constructs discrete finite state abstractions for hybrid systems using predicate abstraction. The tool uses decision procedures and the SAL explicit state model checker. Our approach performs abstraction over the domain of variables, and uses symbolic model checking based on bit-vector decision procedures.

The examples used in this paper have been well-studied; for details on previous case studies, we refer the reader to the relevant references on TCAS [14, 13] and AHS [9, 12].

3 Lazy Linear Hybrid Automata

Definition 1. *A finite precision lazy linear hybrid automaton(LLHA) [3] is a tuple* $(X, V, init, flow, inv, E, jump, \Sigma, D, \epsilon, B, P)$*. The components of LLHA are as follows:*

- *Variables : A finite ordered set* $X = \{x_1, x_2, \ldots, x_n\}$ *of continuous variables.*
- *Control modes : A finite set* V *of control modes.*
- *Initial conditions : A labeling function init that assigns an initial condition to each control mode* $v \in V$*. The initial condition is a predicate over the variables in* X*.*
- *Flow conditions : A labeling function* $flow$ *which assigns a flow condition to each control mode* $v \in V$*.* $flow(v)$ *is required to be a convex linear predicate over the rate of change of variables, that is,* $\dot{X}$*. Such a flow condition can be made rectangular by simplification [4]. If rates of change* $\dot{X}$ *satisfy the flow condition* $flow(s)$*, then we write* $\dot{X} \models flow(s)$*.*
- *Invariant condition : A labeling function inv that assigns an invariant condition to each control mode* $v \in V$*. The invariant condition* $inv(v)$ *is a convex predicate over the variables in* $X \cup \dot{X}$*.*
- *Control switches : A set* E *of edges* (v, v') *from a source mode* $v \in V$ *to a target mode* $v' \in V$*. A function update associates a variable assignment to each control switch.*
- *Jump conditions : A labeling function jump that assigns a jump condition to each control switch* $e \in E$*. A jump condition from the control mode* i *to* j*,* $\psi_{(s,s')}$ *is a predicate over the variables in* $X \cup \dot{X}$*.*
- *Delay parameters :* $D = \{g, \delta_g, h, \delta_h\}$ *is the set of delay parameters such that* $0 \leq g \leq g + \delta_g < h \leq h + \delta_h \leq P$*, where* h *denotes the sensing delay,* g *denotes the actuation delay and* P *is the sampling interval of the controller.*
- *Precision :* ϵ_i *is the precision of measurement of variable* x_i*.*
- *Range :* $B_i = [B_{i_{min}}, B_{i_{max}}]$ *is the range of the variable* x_i*.*

The lazy semantics of hybrid automata [2, 3] means that if a control mode switch took place at time T_k, then the delay in actuating a change in $flow$ lies between $[T_k + g, T_k + g + \delta_g]$. Similarly, a control decision made at time T_{k+1} is based on the values of variables read by the controller at some time in the interval $[T_k + h, T_k + h + \delta_h]$. The parameters δ_g and δ_h represent the bounded uncertainty in actuation and sensing delay respectively. Since the sampling frequency of any implementation of a hybrid automata is always finite, we focus on the discrete time behavior of the hybrid automata.

The precision ϵ_i depends on the accuracy of the sensors measuring x_i from the continuous dynamical system. Guards and state invariants are evaluated on the values of the x_i variables rounded based on ϵ_i. The parameter B reflects the range of values which can be taken by a state variable associated with a fixed width register. Unlike the conventional definition of linear hybrid automata [11], invariants and guards in LLHA can be any computable function.

Definition 2. *A configuration of the hybrid automata, with n continuous variables, is a n+1-tuple,* $c = (s, x_1, x_2, \ldots, x_n)$ *where* $s \in V$ *is the control mode,* $x_1, x_2, \ldots, x_n$ *is the valuation of the continuous variables of the hybrid automata.*

The semantics of a hybrid automaton describes its evolution in terms of change in configuration. We use the notation $c + \alpha$ to denote the configuration in which continuous state variables are incremented by α. Also, we extend the order relation on the continuous variables to configurations. If for each x_i in c and corresponding x_i' in c', if $x_i \leq x_i'$, we write $c \leq c'$.

We define a symbolic collection of configurations as a state of the hybrid automaton and describe the hybrid automaton evolution in terms of change in state. This definition is used in Section 5 to present the bounded model checking algorithm.

Definition 3. *A state of the hybrid automaton is a pair* (v, ϕ) *consisting of a control mode* $v \in V$ *and a predicate* ϕ *over the variables* X*. We identify that the state of a hybrid automaton can change in two ways - flow or jump.*

- *flow: The changed state due to flow after time* T *is* ϕ^T*, defined as* $\phi^T = \exists \dot{X} \{(\phi \wedge inv(s))[X \leftarrow X + \dot{X}T] \wedge \dot{X} \models flow(s) \wedge inv(s)\}$. [1]
- *jump: If* (s, ϕ) *is state of a system, and* (s, s') *is a control switch such that* $\phi \models jump(s, s')$*, then the state of the system can change to* (s', ϕ') *such that if* $update_{(s,s')}$ *was the update function over* $Y \subseteq X$*,*
$$\phi' = (\phi \wedge inv(s) \wedge \psi_{(s,s')})[Y \leftarrow update_{(s,s')}(Y)].$$

A state $s_2 = (v, \phi_2)$ is reachable from $s_1 = (u, \phi_1)$ if and only if there is a sequence of flow or jump transitions from s_1 to s_2.

We now use these definitions to present a hierarchy of abstractions of lazy linear hybrid automata.

4 Hierarchical Abstraction

We detail the theory underlying our hierarchical abstraction technique below.

Agrawal and Thiagarajan [2, 3] use two fundamental quantities in their analysis. The *fundamental time interval* is $\Delta =$ G.C.D of $\{P, g, \delta_g, h, \delta_h\}$. The corresponding abstraction quantum is $\Gamma =$ G.C.D of $\bigcup_i \{\epsilon_i/2, B_i^{min}, B_i^{max}, V_i^{in}, \dot{x}_i\Delta\}$. The quantities Δ and Γ also play a central role in what follows.

Abstraction. We begin with basic definitions on how abstraction is performed. For ease of presentation, all variables are abstracted in the same way.

Definition 4. Q_Π *is a surjection from the continuous variables to integers using* abstraction quantum $\Pi = 2^k\Gamma$ *for some integer* k*. That is,*
$Q_\Pi : \mathbb{R} \to \mathbb{R}$*, and* $Q_\Pi(x_i) = k_i\Pi$ *iff* $x_i = k_i\Pi + \pi_i$*, where* $k_i \in \mathbb{Z}$ *and* $0 \leq \pi_i < \Pi$*.*

[1] $\dot{X}$ denotes $\int_0^T \dot{X}(t)dt/T$, which satisfies the invariants because they are convex.

A configuration $c^d = (s^d, x_1^d, x_2^d, \ldots, x_n^d)$ is a Π-abstraction of a concrete configuration $c = (s, x_1, x_2, \ldots, x_n)$ iff $s^d = s$ and $x_i^d = Q_\Pi(x_i)$.

As before, an abstract transition has two components: $flow$ and $jump$. The $flow$ is simply the evolution of the system in discrete time steps of duration Δ. The definition of $jump$ remains the same as for LLHA.

While performing the above abstraction on configurations/states, we must ensure that state invariants and guards are also correspondingly abstracted. This must be done in order to ensure that $flow$ and $jump$ transitions that are feasible in the concrete LLHA continue to be feasible in the abstract transition system, at the possible cost of introducing additional (spurious) behaviors.

The invariants or guards can be expressed as a boolean combination of atomic predicates in negation normal form, where each predicate is of the form $f(x_1, x_2 \ldots, x_n) \leq b$ where $b \in \mathbb{Q}$.
If Φ is an invariant or guard, then $\Phi = f_{bool}(c_1, c_2, \ldots, c_n)$ where the constraint c_i is $f_i \leq b_i$,
where f_{bool} represents a boolean combination of its parameters.

Each predicate in the abstract invariant or guard can be abstracted using the monotonicity of f with respect to each variable x_i over the range $[Q_\Pi(x_i), Q_\Pi(x_i + \Pi)]$, that is, $f_{x_i} = \frac{\delta f}{\delta x_i}$ is of the same sign over the range of interest. In particular, all polynomials which are linear in each variable, are always monotonic with respect to each variable.

In order to define abstract state invariants and guards, we first describe how to construct abstract inequalities using the above observation about well-behaved invariants and guards. Without loss of generality, let us assume that $f(x_1, x_2 \ldots, x_n) \leq b$ be an inequality such that its partial derivatives with respect to each variable is of the same sign over the range of interest $[Q_\Pi(x_i), Q_\Pi(x_i + \Pi)]$. Its conservative abstraction is the relaxed inequality

$$c_i' \equiv f(k_1, k_2 \ldots, k_n) \leq b'$$
where $b' = Q_\Pi(b + \Pi)$ and $k_i = Q_\Pi(x_i)$ if $f_{x_i} \geq 0$
$= Q_\Pi(x_i + \Pi)$ if $f_{x_i} < 0$

This abstraction rounds up or down each variable to the nearest multiple of Π depending on whether the function f decreases or increases with increase in the variable. The constant b is always rounded up. All assignments to the variables which satisfied the earlier constraint also satisfy the relaxed constraint. Hence, this is an overapproximation of the original constraint. The intuition behind the following definition of abstract guards and invariants is to relax the atomic constraints so that if $\Phi(x_1, x_2, \ldots, x_n)$ denotes a state invariant or guard, then the corresponding abstracted invariant or guard is $\Phi^a(k_1, k_2, \ldots, k_n)$ such that

$$\forall x_1, x_2, \ldots, x_n \; [\Phi(x_1, x_2, \ldots, x_n) \implies \exists k_1, k_2, \ldots, k_n \; \Phi^a(k_1, k_2, \ldots, k_n)]$$

If $\Phi(x_1, x_2, \ldots, x_n) = f_{bool}(f_1 \leq b_1, \ldots, f_n \leq b_n)$ is the invariant or guard, the abstract state invariant or guard is defined as
$\Phi^a(k_1, k_2, \ldots, k_n) = f_{bool}(c_1', c_2', \ldots, c_n')$
where the relaxed inequalities c_i' are obtained from $f_i \leq b_i$ as described above.

We illustrate the above technique for obtaining abstract invariants and guards described above using an example. Let $267(x-35)/x \leq 150$ be an invariant. It abstraction when Γ is 1 and Π is $2^5 = 32$, would be $8((k-2)/k) \leq 5$. The solutions for the invariant are $x \leq 32 \times 267/117$, that is, with $\Gamma = 1$, the feasible values of x are $x \leq 80$. Now, the solutions of the abstract invariant are $k \leq 6$, that is, the abstraction allows $x < 32k = 192$. Thus, this relaxation results into an upper approximation of the behavior of the hybrid automaton.

As before, the abstract state (s_2, ϕ_2^d) is reachable from (s_1, ϕ_1^d) if there exists a sequence of $flow$ or $jump$ transitions from (s_1, ϕ_1^d) to (s_2, ϕ_2^d) in the abstract hybrid automata.

If Π is the quantum used for abstraction by the surjection Q, then the corresponding obtained transition system is called Π-transition system. It can be noted that the Π-transition system obtained from an LLHA does not simulate or bisimulate the LLHA in general. We examine this transition system for the special case that $\Pi = \Gamma$ and we will later augment this transition system to construct an over-approximation of the LLHA for $\Pi = 2^k\Gamma$, $k \geq 1$.

Prior Results. It has been shown [3] that there is a bisimulation relation between the Γ-transition system and the original LLHA. The results are summarized in Theorem 1.

Theorem 1. *[3] Let a configuration of hybrid automata be $c = (s, x_1, x_2, \ldots, x_n)$ and its Γ-abstract configuration be $c^d = (s, Q_\Gamma(x_1), Q_\Gamma(x_2), \ldots, Q_\Gamma(x_n))$. A configuration c' is reachable from c iff $Q_\Gamma(c') = c'^d$ where c'^d is reachable from c^d in Γ-transition system.*

Let x_{max} and x_{min} be the maximum and minimum values that can be attained by any continuous variable. The state space size of the Γ-transition system is $O(2^{2n}(\frac{x_{max}-x_{min}}{\Gamma})^n)$, that is, exponential in the number of continuous variables. This huge state space makes it impractical to do any enumerative reachability analysis.

Our Results. We now define an over-approximation of an LLHA called the *k-abstraction* (of the LLHA).

Definition 5. *A k-abstraction ($k \geq 1$) of a lazy linear hybrid automaton is an augmentation of a Π-transition system of that automaton such that the following additional conditions are satisfied:*

- $\Pi = 2^k\Gamma$.
- *If $\dot{x} = \alpha\Gamma/\Delta$ is a rate of change allowed by $flow(s)$ for some location s in the Π-abstract transition system, then the following two rates of change are allowed in location s in the k-abstraction: $\lfloor(\frac{\alpha}{2^k})\rfloor\Pi/\Delta$ and $\lceil(\frac{\alpha}{2^k})\rceil\Pi/\Delta$.*

The 0-abstraction is defined as the Γ-transition system.

Note that the rates are integral multiples of Γ/Δ. The intuition behind the second condition is to add behaviors to the abstraction so that even if the configuration has been "rounded down" in performing abstraction, it can still evolve

"far enough." For example, if a variable could have a rate of change $35\Gamma/\Delta$, then the 3-abstraction allows rates of change $\lceil \frac{35}{2^3} \rceil (2^3\Gamma/\Delta)$ and $\lfloor \frac{35}{2^3} \rfloor (2^3\Gamma/\Delta)$, that is, $(40\Gamma/\Delta)$ and $(32\Gamma/\Delta)$. We show in the following theorem that this leads to overapproximation of the dynamics of LLHA as expected.

Lemma 1. *Let a configuration of hybrid automata be $c = (s, x_1, x_2, \ldots, x_n)$ and its Π-abstraction be $c^d = (s, k_1\Pi, k_2\Pi, \ldots, k_n\Pi)$, where $\Pi k_i = Q_\Pi(x_i)$. For all configurations c' reachable from c in time $T = l\Delta$ in the LLHA, there exists an abstract configuration c'^d reachable from c^d in the Π-transition system such that c'^d and c' are related as follows:*

$$c'^d \leq Q_\Pi(c') \leq c'^d + l\Pi$$

where $Q_\Pi(c') = (s, k'_1\Pi, k'_2\Pi, \ldots, k'_n\Pi)$ with $\Pi k'_i = Q_\Pi(x'_i)$ for all i.

Proof. For configuration $c = (s, x_1, x_2, \ldots, x_n)$, let $\dot{x}_1, \dot{x}_2 \ldots, \dot{x}_n$ be the rates of change of continuous variables satisfying $flow(s)$ and $\widehat{x_1}, \widehat{x_2}, \ldots, \widehat{x_n}$ be the rates of change of continuous variables satisfying $flow(\widehat{s})$ where $\widehat{s}$ is a predecessor state of s, that is, $(\widehat{s}, s) \in E$. Let c' be a configuration reachable from c. In case of change due to reset of variables at jump, the above lemma follows due to the adjustment to guards and invariants. We prove the above lemma for the case where the change is effected due to flow evolution.

Since, the relation $\leq$ for configurations is defined in terms of the ordering of individual variable, we consider an arbitrary variable in the rest of the proof below. If x_i is the value of the variable in c and x'_i is the value in c' after time T such that the flow rate switched after an actuation delay of t, then
$x'_i = x_i + \widehat{\dot{x}_i}t + \dot{x}_i(T - t)$
Using the definition of Γ and Δ,
$x_i = (m2^k + n)\Gamma + \gamma_i$,
$\widehat{\dot{x}_i}\Delta = (2^k p' + q')\Gamma$,
$\dot{x}_i\Delta = (2^k p + q)\Gamma$,
where $0 \leq n < 2^k$, $0 \leq \gamma_i < \Gamma$, $0 \leq q' < 2^k$, $0 \leq q < 2^k$.

So, $x'_i = (m2^k + n)\Gamma + \gamma_i + (2^k p' + q')\frac{\Gamma}{\Delta}t + (2^k p + q)\frac{\Gamma}{\Delta}(l\Delta - t)$
$= (m2^k + n)\Gamma + \gamma_i + (2^k(p' - p) + (q' - q))\frac{\Gamma}{\Delta}t + (2^k p + q)l\Gamma$
Thus, $x'_i = (m + pl)2^k\Gamma + (n + ql)\Gamma + \gamma_i + (2^k(p' - p) + (q' - q))\frac{\Gamma}{\Delta}t$.

Since $0 \leq t < T$ in the above equation and $2^k\Gamma = \Pi$, x'_i lies in the interval

- $[(m + pl)\Pi, (m + (p' + 1)l + 1)\Pi)$ if $\dot{x}_i \geq \widehat{\dot{x}_i}$
- $[(m + p'l)\Pi, (m + (p + 1)l + 1)\Pi)$ if $\widehat{\dot{x}_i} \geq \dot{x}_i$

So, $Q_\Pi(x'_i)$ lies in the interval

- $[(m + pl)\Pi, (m + (p' + 1)l)\Pi]$ if $\dot{x}_i \geq \widehat{\dot{x}_i}$
- $[(m + p'l)\Pi, (m + (p + 1)l)\Pi]$ if $\widehat{\dot{x}_i} \geq \dot{x}_i$

The value of i^{th} variable in any configuration c'^d reachable from c^d in the Π-transition system, x'^d_i lies in

– $[(m+p'l)\Pi, (m+pl)\Pi]$ if $\dot{x}_i \geq \widehat{\dot{x}_i}$
– $[(m+pl), (m+p'l)]$ if $\widehat{\dot{x}_i} \geq \dot{x}_i$

Thus, for any x'_i, there exists x'^d_i such that $x'^d_i \leq Q_\Pi(x'_i) \leq x'^d_i + l\Pi$. Using the same argument for each variable independently, the lemma immediately follows. □

Lemma 1 provides the intuition behind Definition 5. Note how the construction of k-abstraction from Π-transition system uses the augmented flow to eliminate the $l\Pi$ factor in the above lemma. The allowed increment of Π/Δ in flow values in k-abstraction can capture states reachable in $T = l\Delta$ upto $l\Delta \times \Pi/\Delta = l\Pi$ more than the states captured in Π-transition system. We formally state this below.

Theorem 2. *Let a configuration of hybrid automata be $c = (s, x_1, x_2, \ldots, x_n)$ and its abstraction be $c^d = (s, Q_\Pi(x_1), Q_\Pi(x_2), \ldots, Q_\Pi(x_n))$, where $\Pi = 2^k\Gamma$. If a configuration c' is reachable from c and $Q_\Pi(c') = c'^d$, then c'^d is reachable from c^d in the k-abstraction.*

Using reasoning exactly similar to the one used in Lemma 1, we can prove the hierarchy of k-abstractions presented below.

Lemma 2. *Let a configuration of k-abstraction be*
$c = (s, Q_\Pi(x_1), Q_\Pi(x_2), \ldots, Q_\Pi(x_n))$, where $\Pi = 2^k\Gamma$.
Its abstraction in $\widetilde{k}$-abstraction, where $\widetilde{k} \geq k$
$\widetilde{c} = (s, Q_{\widetilde{\Pi}}(x_1), Q_{\widetilde{\Pi}}(x_2), \ldots, Q_{\widetilde{\Pi}}(x_n))$ where $\widetilde{\Pi} = 2^{\widetilde{k}}\Gamma$.
If a configuration c' is reachable from c in k-abstraction, then

– *$Q_{\Pi'}(c') = \widetilde{c}'$ where $\Pi' = 2^{\widetilde{k}-k}\Gamma$*
– *$\widetilde{c}'$ is reachable from $\widetilde{c}$ in $\widetilde{k}$-abstraction.*

Proof. We use similar notations as in Lemma 1 and put $k_i = Q_\Pi(x_i)$ and $\widetilde{k}_i = Q_{\widetilde{\Pi}}(x_i)$.
Let $\widetilde{k} - k = d$ for some $d > 0$, then $k_i = (m2^d + n)\Pi$, $\widehat{\dot{k}_i}\Delta = (2^d p' + q')\Pi$, $\dot{k}_i\Delta = (2^d p + q)\Pi$ and $T = l\Delta$, where $0 \leq n < 2^d$, $0 \leq q' < 2^d$, $0 \leq q < 2^d$.

Also, $x_i = k_i 2^k + \alpha_k\Gamma = \widetilde{k}_i 2^{\widetilde{k}} + \alpha_{\widetilde{k}}\Gamma$, where $0 \leq \alpha_k < 2^k$ and $0 \leq \alpha_{\widetilde{k}} < 2^{\widetilde{k}}$.
So, $x_i = (m2^d + n)2^k\Gamma + \alpha_k\Gamma$ (using $k_i = (m2^d + n)$)
$= m2^{\widetilde{k}}\Gamma + n2^k\Gamma + \alpha_k\Gamma$ ($\widetilde{k} = k + d$)
Since $n \leq 2^d - 1$ and $\alpha_k \leq 2^k - 1$, so $n2^k + \alpha_k \leq (2^d - 1)2^k + 2^k - 1$, that is, $n2^k + \alpha_k \leq 2^{\widetilde{k}} - 1 < 2^{\widetilde{k}}$.
It immediately follows that $m2^{\widetilde{k}}\Gamma = \widetilde{k}_i = Q_{\widetilde{\Pi}}(x_i)$ and $n2^k + \alpha_k = \alpha_{\widetilde{k}} = Q_{\widetilde{\Pi}}(\alpha_{\widetilde{k}})$.
Similarly, $p2^{\widetilde{k}}\Gamma = Q_{\widetilde{\Pi}}(\dot{x}_i)$ and $p'2^{\widetilde{k}}\Gamma = Q_{\widetilde{\Pi}}(\widehat{\dot{x}_i})$.

The evolution for k-abstraction would be
$k'_i = k_i + \widehat{\dot{k}_i}t + \dot{k}_i(T - t)$

So, $k'_i = (m2^d + n)\Pi + (2^d p' + q')\Pi\frac{t}{\Delta} + (2^d p + q)\Pi\frac{(l\Delta - t)}{\Delta}$
$= (m2^d + n)\Pi + (2^d(p' - p) + (q' - q))\Pi\frac{t}{\Delta} + (2^d p + q)l\Pi$
$= (m + pl)2^d\Pi + (n + ql)\Pi + (2^d(p' - p) + (q' - q))\Pi\frac{t}{\Delta}$.

Since $0 \leq t < T$, k'_i lies in the interval

- $[(m + pl)2^d\Pi, (m + (p' + 1)l + 1)2^d\Pi)$ if $\dot{k}_i \geq \widehat{\dot{k}_i}$
- $[(m + p'l)2^d\Pi, (m + (p + 1)l + 1)2^d\Pi)$ if $\widehat{\dot{k}_i} \geq \dot{k}_i$.

$Q_{\Pi'}(k'_i)$ lies in

- $[(m + pl)\widetilde{\Pi}, m + (p' + 1)l\widetilde{\Pi}]$ if $\dot{k}_i \geq \widehat{\dot{k}_i}$
- $[(m + p'l)\widetilde{\Pi}, m + (p + 1)l\widetilde{\Pi}]$ if $\widehat{\dot{k}_i} \geq \dot{k}_i$

Using $m2^{\widetilde{k}}\Gamma = \widetilde{k}_i$, $p2^{\widetilde{k}}\Gamma = \widetilde{\dot{k}_i}$ and $p'2^{\widetilde{k}}\Gamma = \widetilde{\widehat{\dot{k}_i}}$ obtained above $Q_{\Pi'}(k'_i)$ lies in

- $[(\widetilde{k}_i + \widetilde{\dot{k}_i}l), \widetilde{k}_i + (\widetilde{\widehat{\dot{k}_i}} + 1)l]$ if $\dot{k}_i \geq \widehat{\dot{k}_i}$
- $[(\widetilde{k}_i + \widetilde{\widehat{\dot{k}_i}}l), \widetilde{k}_i + (\widetilde{\dot{k}_i} + 1)l]$ if $\widehat{\dot{k}_i} \geq \dot{k}_i$

Thus, $Q_{\Pi'}(k'_i) = \widetilde{k}_i$. Similar arguments for all the variables leads to the lemma $Q_{\Pi'}(c') = \widetilde{c}'$. □

Definition 6. *We define a partial order relation $\preceq$ between transition systems. If all the states reachable from the initial states in a transition system T have their corresponding abstract states reachable from the abstract initial states in T', then $T \preceq T'$.*

The hierarchy Theorem 3 follows from lemma 2 and Theorem 1.

Theorem 3. *k-abstraction $\preceq$ k'-abstraction if $0 \leq k < k'$. Thus, k-abstractions, where $k \geq 0$, form an hierarchical abstractions of the lazy linear hybrid automata. Further, 0-abstraction is the Γ-abstract transition system which bisimulates the lazy linear hybrid automata.*

The abstraction hierarchy presented above can be effective in practice in containing the state space explosion problem with the Γ-abstraction; while the latter is a finite bisimulation quotient of the lazy linear hybrid automata but has a huge state space size of $O(2^{2n}(\frac{x_{max} - x_{min}}{\Gamma})^n)$. Theorem 4 presents the relative reduction in state space size with k.

Theorem 4. *Let S_k be the state space size of k-abstraction and S'_k of k'-abstraction where $k' > k$. Then $S'_k/S_k = 2^{n(k'-k)}$.*

Theorem 3 provides a framework for use of progressive abstraction of lazy linear hybrid automata to develop a sound and complete abstraction-refinement paradigm for reachability analysis of LLHA. We demonstrate with our experimental results that the progressive abstraction can yield a significant reduction in model checking time.

5 Model Checking *k*-abstractions of LLHA

Our implementation of a symbolic verifier of LLHA is based on three techniques: bounded model checking, "k-induction", and an overall counterexample-guided abstraction-refinement [7] framework. We describe each of these below.

Bounded model checking. We describe how the BMC formula is constructed, starting with a useful definition.

Definition 7. *A frame (F) is a tuple (K, t_1, t_2, t, l) where $K = (k_1, k_2 \ldots k_n)$ represent the variables, t_1 is the sensing delay, t_2 is the actuation delay t_2, t is the time before transition to next frame, l denotes the control mode.*

The initial state of the hybrid automata is the predicate $Init(F_0) \equiv (l = v_{start}) \wedge \phi_0(K)$, where v_{start} denotes the initial control mode and ϕ_0 the initial predicate over continuous variables.

The transition T is defined as a predicate over the previous frame (F_{m-1}) and the present frame (F_m). It is a disjunction of all possible state switches (G_{ij}) and flow evolutions (E_i).

$$T(F_{m-1}, F_m) \equiv \bigvee_{(i,j)\in E} G_{ij}(F_{m-1}, F_m) \vee \bigvee_{i \in V} E_i(F_{m-1}, F_m)$$

The switch predicates G_{ij} and the time evolution predicates E_i are themselves defined in terms of three other quantities: I_i is a predicate that tests satisfiability of state invariant inv_i at control mode i, predicate g_{ij} tests satisfiability of guard ψ_{ij}, and e_{hi} deals with time evolution in control mode i with predecessor mode h.

Let us consider two functions - *compensated for sensing delay (csd)* and *compensated for actuation delay (cad).* These map a set of valuations of the continuous variables (K) to a set of possible corresponding valuations obtained after compensating for sensing and actuation delay respectively.

$csd(K, i, t_1) =$
$\{(k_1 - \dot{k}_1 t_1, \ldots, k_n - \dot{k}_n t_1) | (\dot{k}_1, \dot{k}_2, \ldots \dot{k}_n) \models flow(i)\}.$

$cad(K, h, i, t_2, t) =$
$\{(k_1 + (\dot{k}_{1h} - \dot{k}_{1i})t_2 + \dot{k}_{1i}t, \ldots, k_n + (\dot{k}_{nh} - \dot{k}_{ni})t_2 + \dot{k}_{ni}t) | (\dot{k}_{1h}, \dot{k}_{2h}, \ldots \dot{k}_{nh}) \models flow(h)$
and $(\dot{k}_{1i}, \dot{k}_{2i}, \ldots \dot{k}_{ni}) \models flow(i)\}.$

Let the current frame be $F_m = (K^m, t_1^m, t_2^m, t^m, l^m)$ and the previous frame be $F_{m-1} = (K^{m-1}, t_1^{m-1}, t_2^{m-1}, t^{m-1}, l^{m-1})$.

$$I_i(F_m) \equiv (i = l^m) \wedge \exists K'[K' \in csd(K^m, l^m, t_1^m) \wedge inv_i(K')]$$

$$e_{hi}(F_{m-1}, F_m) \equiv (i = l^{m-1} \wedge i = l^m) \wedge K^m \in cad(K^{m-1}, h, i, t_2^{m-1}, t^m)$$

$$g_{ij}(F_{m-1}, F_m) \equiv (i = l^{m-1} \wedge j = l^m) \wedge \exists K'[K' \in csd(K^{m-1}, l^{m-1}, t_1^{m-1}) \wedge \psi_{ij}(K')]$$

The switch and evolution predicates can now be defined as follows:

$$G_{ij}(F_{m-1}, F_m) \equiv I_i(F_{m-1}) \wedge I_j(F_m) \wedge g_{ij}(F_{m-1}, F_m) \wedge [K^m = update_{ij}(K^{m-1})]$$

$$E_i(F_{m-1}, F_m) \equiv I_i(F_{m-1}) \wedge I_i(F_m) \wedge [\bigvee_{h \in pred(i)} e_{hi}(F_{m-1}, F_m)]$$

where $pred(i)$ denotes the set of predecessor locations of i.

This completes the definition of the transition predicate.

Let the state to be checked for reachability be (s_r, ϕ_r). If reachability analysis is used to check safety properties, then (S_r, ϕ_r) would be the error state violating the safety property, Then, the predicate $reach(F) \equiv (l = s_r \wedge \phi_r(K))$ represents the error state, that is the *target* state for reachability analysis.

If d is the number of steps to which we want to check the k-abstraction for reachability of (s_r, ϕ_r), we need to check for the satisfiability of

$$BMC^d \equiv Init(F_0) \wedge \bigwedge_{n=1}^{d} (T(F_{n-1}, F_n)) \wedge reach(F_n).$$

If BMC^d is satisfied, then the target state $(s_r, \phi_r(K))$ is reachable in k-abstraction and the frames $F_0, F_1, \ldots, F_d$ gives a trace from the start state to the target state.

Further, it is sufficient to do BMC for p steps to prove that a target state is not reachable where p is the diameter of the transition system. If BMC^p is unsatisfiable, then the target state can not be reached in the transition system. Since the number of reachable states of the transition system provides an overestimate of the diameter, it is sufficient (though unrealistic) to do BMC for number of steps equal to the state space size of the k-abstraction.

Induction. We now describe an induction procedure to guarantee the unreachability of a state in a model. This can be used to prove the satisfaction of a safety property which can be expressed as a reachability query.

If N steps of BMC are found to be not satisfiable, that is, BMC^N is unsatisfiable, then we test the satisfiability of

$$\neg reach(F_0) \wedge \bigwedge_{k=1}^{N+1} (T(F_{k-1}, F_k)) \wedge reach(F_{N+1}).$$

If the above is unsatisfiable, no further bounded model checking is required and all the states of the model are guaranteed to satisfy the property. Based on this, we present below a BMC algorithm along with use of induction to check for safety properties in a LLHA. We define the following predicates to be used in the algorithm.

$$N^j(F_j) \equiv Init(F_0) \wedge \bigwedge_{k=1}^{j} T(F_{k-1}, F_k) \text{ and } S^{j+1}(F_{j+1}) \equiv \neg reach(F_0) \wedge \bigwedge_{k=1}^{j+1} T(F_{k-1}, F_k)$$

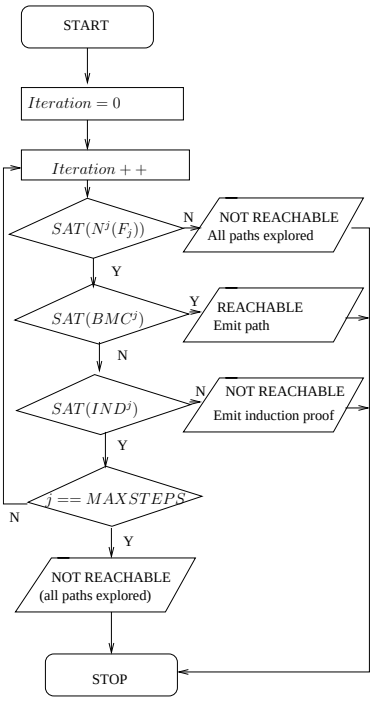


Fig. 1. Symbolic reachability analysis based on BMC and induction

If at any step of the BMC, we find that $N^j(F_j)$ is not satisfiable, it means that there does not exist a path of length j or more, and hence we can terminate with the output that the model satisfies the safety property.

The bounded model checking predicate and the induction step predicate are $BMC^j \equiv N^j(F_j) \wedge reach(F_j)$ and $IND^j \equiv S^{j+1}(F_{j+1}) \wedge reach(F_{j+1})$

The sub-routine INDBMC is presented in Figure 1. The technique is sound and complete due to the results of the preceding section; we present a detailed discussion of the abstraction-refinement framework in the next section.

6 Counterexample guided refinement of k-abstractions

We now describe an automated CEGAR [7, 6] technique presented in Figure 2 which exploits the abstraction hierarchy presented in section 4. An initial coarse abstraction can be arbitrary chosen as k_0-abstraction depending on the size of the state space.

In case the target state is not reachable in k_0-abstraction, the target state is also not reachable in the LLHA by Theorem 2. In case the target state is reachable in LLHA, then BMC will yield a path p_0 from the initial state to the target state in the k_0-abstraction. This needs to be validated with respect to

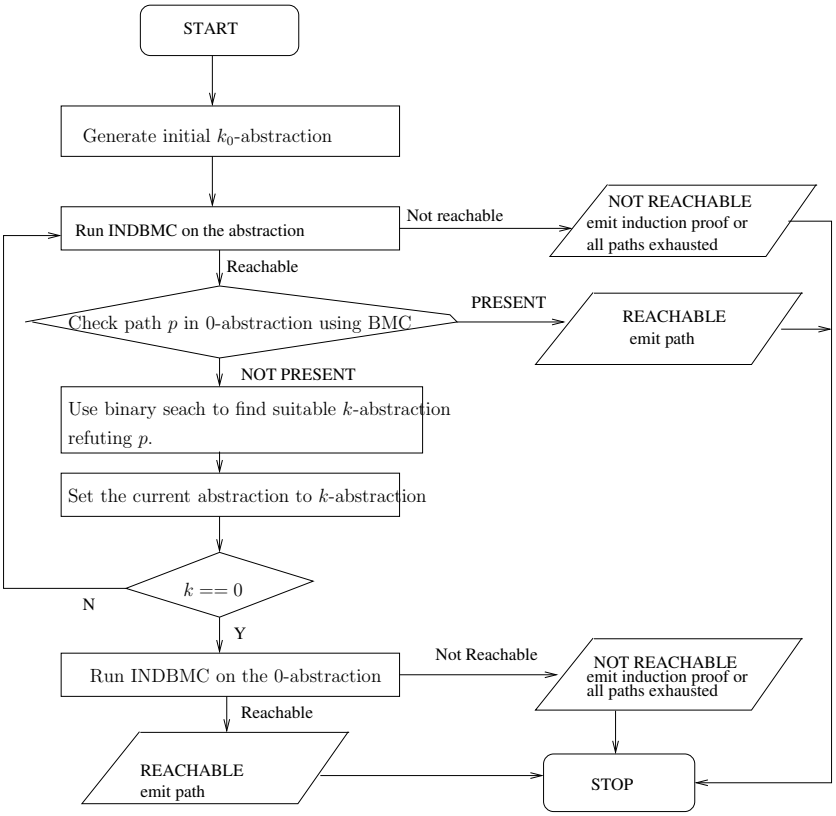


Fig. 2. Reachability analysis of LLHA using iterative refinement

the 0-abstraction. If the abstract path p_0 found in k_0-abstraction is present in 0-abstraction, then the target state is reachable in the LLHA too. If it is not present in 0-abstraction, then we select a more finer refinement k_i-abstraction which refutes the abstract spurious path. The same technique is repeated for progressively finer abstractions until the target states is shown to be unreachable or a valid path to the target sate is found.

The key components of this technique are counter-example validation technique and automated refinement step. The path obtained at any iteration is a satisfying assignment to BMC^j, it can be validated on 0-abstraction by doing a BMC on BMC^j to identify the first spurious transition. If BMC^j has a satisfying assignment for 0-abstraction too, then the path is valid. The hierarchy of abstractions allows the use of binary search to find the smallest value of l such that the l-abstraction refutes the path identified in the coarser abstraction. The complete technique for reachability analysis of LLHA based on iterative refinement and bounded model checking is presented as a flowchart in Figure 2.

In the rest of the section, we describe these two key components in more detail. The validation of counterexample obtained on k_i-abstraction is done by using BMC on the 0-abstraction. We also describe a binary search based refinement technique which exploits the linear hierarchy of abstractions to find the coarsest abstraction that refutes the spurious path. This ensures that the increase in state space at every step of iterative refinement is minimal and is sufficient to refute the spurious paths to the target state. The problem of finding the coarsest refinement to refute an abstract path is NP-hard [7]. Hence, the refinement technique presented below is only a best effort technique and does not guarantee that the state space requirement at every iteration step is minimum. This technique only ensures that the k_i-abstraction selected at every iteration is the coarsest of possible k-abstractions.

Using BMC to validate an abstract path: If j steps of BMC were needed to obtain a counterexample in k-abstraction, then the satisfying assignment to BMC^j is the path from initial state to the target state. This can be validated with respect to the concrete LLHA by checking the satisfiability of $BMC^j \equiv Init(F_0) \wedge \bigwedge_{k=1}^{j} T(F_{k-1}, F_k) \wedge reach(F_j)$ where the frames F_i have continuous variables discretized using Γ. This checks the presence of the abstract path in the 0-abstraction. This check is performed in an incremental manner so that we can locate the minimum value of j' such that $N^{j'}(F_{j'})$ can not be satisfied. This corresponds to finding the first abstract transition in the abstract state which is spurious. In case no such j' exists and BMC^j is satisfiable in 0-abstraction too, then the path is valid and the corresponding assignment can be given as an evidence of reachability of the target state.

Binary search based automated refinement: In case, the abstract path is found to be spurious, we need to consider a suitable refinement for the next iteration which refutes this spurious path. The present abstraction k_i contains the spurious path but the 0-abstraction does not contain any path corresponding to the spurious path. Since we know that there is a linear hierarchy of k-abstractions, we can use binary search between 0 and k_i to find the smallest value of l such that l-abstraction refutes the counterexample.

This will need only $log(k_i)$ iterations and will yield the smallest k-abstraction which refutes the spurious path. This can be used in the next iteration of bounded model checking based reachability analysis.

BMC using iterative refinement: The complete technique for reachability analysis of LLHA based on iterative refinement and bounded model checking is presented as a flowchart in Figure 2.

The soundness of this technique is ensured by Theorem 2 and 1. Since, we start with some initial k_0-abstraction and every step involves a progress in refinement, we take at most k_0 iterations before terminating.

Theorem 5. *The iterative abstraction refinement technique presented in Figure 2 is sound and complete.*

At any iteration if we conclude that the target state is never reached by doing reachability analysis at some k_i-abstraction, Theorem 2 guarantees that our conclusion is correct. If the path found in some k-abstraction is found to be present in the 0-abstraction, we conclude that the target state is reachable in LLHA. The correctness of this conclusion is ensured byh Theorem 1.

In every iteration, we either terminate concluding either way or we refine the abstraction to some finer k-abstraction from k'-abstraction ($k' < k$) in the abstraction hierarchy established in Theorem 3. Since, we start with some finite k_0 as our initial k_0-abstraction and every step involves a progress in refinement, we take at most k_0 iterations before terminating.

7 Experiments and Results

In this section, we present the results of experiments on two case studies. All experiments were performed on a workstation with Intel Xeon 3.06 GHz processors and 4GB RAM.[2]

Automated Highway Control System: AHS (Figure 3) is an arbiter which ensures that there is no collision between cars running on a highway by imposing legal speed ranges. AHS monitors the distance between cars and switches to recovery mode from cruise mode if there is a possibility of collision. The legal speeds of cars are then altered to bring the system back to cruise mode. If there is a collision between the cars then the AHS reaches error mode. We need to check whether error mode is reachable. This example has being widely used in literature [9, 12]. We use the description by Jha et al [12] and extend it to handle inertial delays. We take note of the following features of this example:

1. All constraints in this example are linear and hence, Phaver can be used for its reachability analysis.
2. The number of locations as well as variables increase linearly with the number of cars. Thus, we use the number of cars as a parameter to scale the example.

A set of legal parameter values is: (All distance measures are in km, time is in hr and all speeds are in km/hr)
$\alpha = .002$, $\alpha' = .0005$, $a = 10, rl = 20, b = 30, c = 40, d = 50, e = 60, ru = 70, f = 100$
Precision of distance (ϵ) is 10^{-5}, delay factor $g = 1e-3$, $h = 5e-4$, the uncertainties for delay are $\delta_g = 5e-4, \delta_h = 5e-5$ and the sampling is done with a period of .01. Corresponding to this the required quantization factors are $\Delta = 5e-5$ and $\Gamma = 5e-5$.

Thus, safety property to be verified was that the control mode is never the "error" mode. This model was built in UCLID to employ our technique and was also analyzed using Phaver to allow us to compare time and space performance.

[2] A complete set of UCLID, Phaver or HSolver modules as well as data pertaining to run-time and memory requirements can be obtained from the first author's webpage.

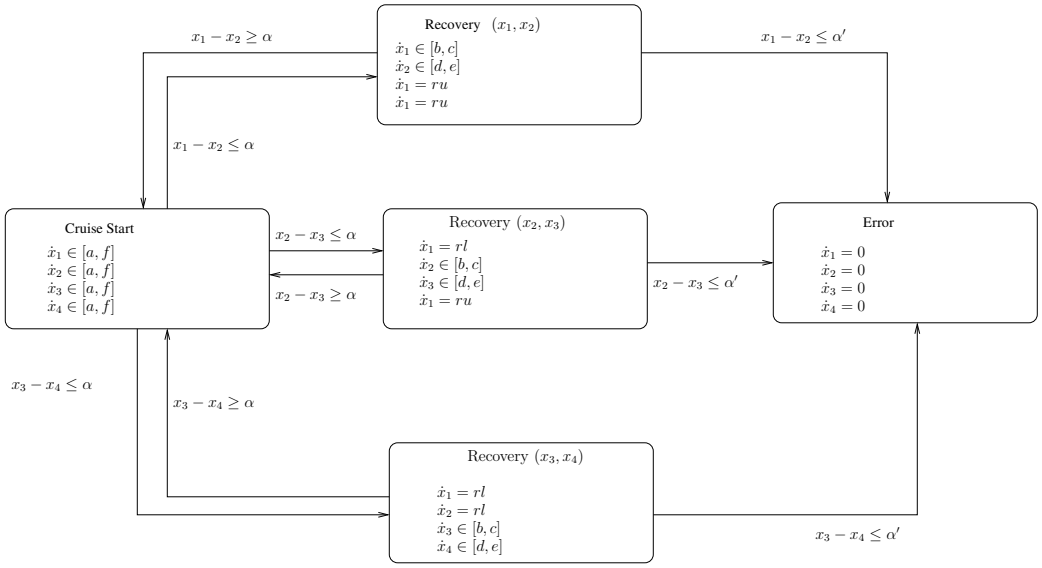


Fig. 3. Automated Highway Control System with 4 vehicles

In order to compare our results , we exploit the parametric nature of the example of AHS. We vary the number of cars to observe the variance in performance of our technique and Phaver, as the size of the model changes. We point out certain key observations made during this comparison.

1. Phaver takes more than 10 hours to analyze a model in which there are 15 cars.
2. The growth in time as well as space requirement for Phaver is exponential.
3. Our technique takes less than 2 minutes to analyze a model which has 150 cars.
4. The growth in time for our technique is sub-exponential.
5. The SAT solving time was observed to be less than 1 seconds for number of cars from 4 to 150. The major component of run time was the model-building time.

Figure 4 and Table 1 compare the runtime of our technique and that of Phaver on this example for different number of cars. It shows that our approach is more scalable than Phaver and hence, can handle AHS with realistic number of cars. We did not need to use any k-abstraction to obtain our results.

Conclusion from Figure 4: The plot shows that a very reasonable discretization based on delay and sampling frequency can make the problem of reachability much easier to resolve. The subexponential nature of run-time for our technique upto 15 cars in this plot (the same nature was observed till 150 cars) reflects an order of magnitude decrease in the run-time requirement from the super-exponential run-time of Phaver.

Number of cars	UCLID total run-time	UCLID SAT-time	Phaver run-time	Phaver memory
4	4.626	0.094	0.20	1424
5	5.685	0.118	0.31	5012
6	6.805	0.150	0.51	5528
7	8.100	0.172	0.92	6332
8	9.170	0.197	1.63	7512
9	10.639	0.225	3.57	9860
10	11.655	0.256	8.17	12608
11	12.990	0.297	27.96	20460
12	14.027	0.332	121.15	32724
13	15.071	0.371	655.36	56280
14	16.707	0.400	4520.35	103372

Table 1. Table comparing results with Phaver (time is in seconds, memory is in KB)

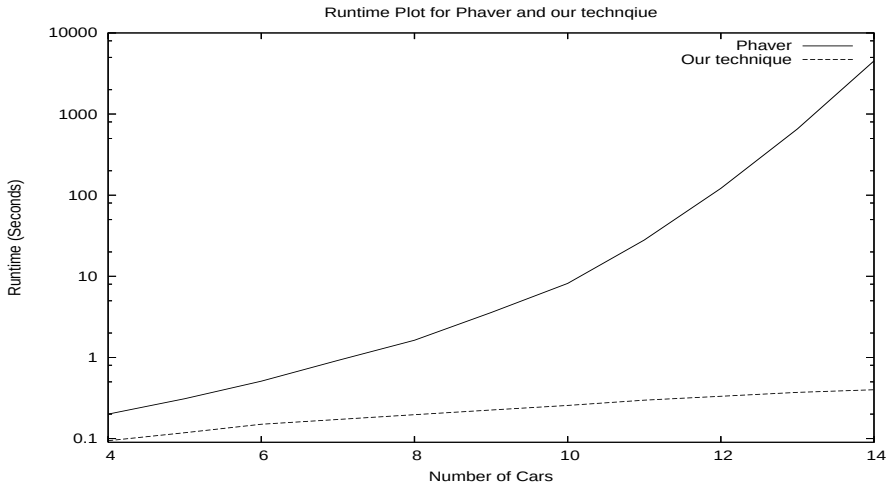


Fig. 4. Plot comparing runtimes of Phaver and our technique

Further, the bulk of the run-time taken by our technique is used up in building the model. The time taken to solve the corresponding SAT problems for BMC and Induction are a very small percentage of total run time. For 150 cars, the total run time is nearly 120 seconds, but the SAT solving time is of the order of just 1 second.

Air Traffic Alert and Collision Avoidance System: TCAS is a predictive warning system used for avoiding collision of aircrafts using a sequence of preventive and corrective resolution advisories. The model for TCAS resolution used here is similar to the one used by Pappas et al [14]. We make the following changes to the model to make it more realistic.

1. The TCAS specification [5] uses *expected time to collision* for detecting collision threats and not distance between aircrafts used in Pappas et al example

[14]. The max in the constraint avoids division by zero. The k/x_r term ensures that slow approaches are avoided by triggering threat if x_r is small. This makes the problem harder since these invariants are non-linear. Hence, LHA tools like Phaver can not be used for this example.

2. We allow the input for speed of aircrafts to be an interval. It is realistic to expect the speed of aircrafts to be in a range rather than assuming them to be a constant input.

3. We also allow inertial delays in actuation and sensing.

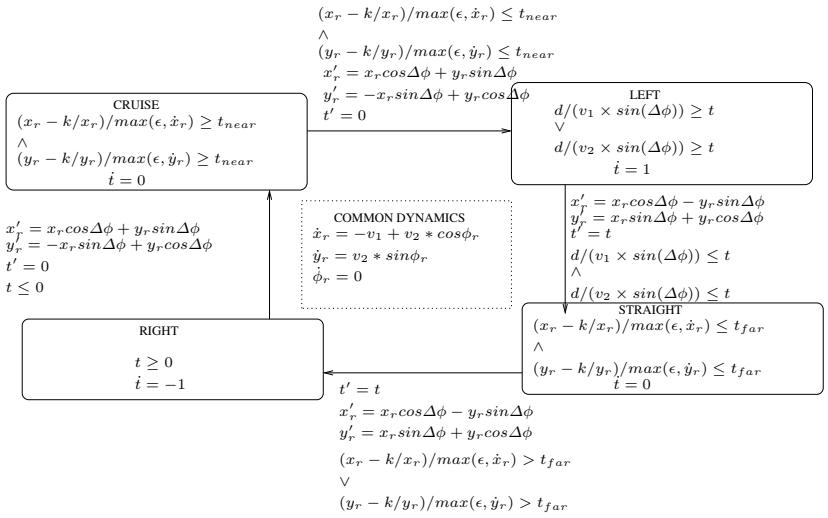


Fig. 5. Air Traffic Alert and Collision Avoidance System

The parameters used in the experiment were taken from the specifications in TCAS 2, Version 7 documentation [20] and TCAS-201 simulator [17] specifications. The time-zone considered for advisory is $30 - 120$ seconds (t_{near} and t_{far}, respectively). The distance d is taken to be 15 nautical miles (that is, 27.78 kms.). The range of speed for aircraft is allowed to range between 100 knots to 510 knots (nearly 200 km/hr to 1000 km/hr). It may be noted that the maximum speed of Airbus 380 is Mach 0.88 (nearly 505 knots). The LLHA parameters used in our example are $128\mu s \leq g, h \leq 256\mu s$ and $\epsilon = 2^{-15}$ nautical miles [17].

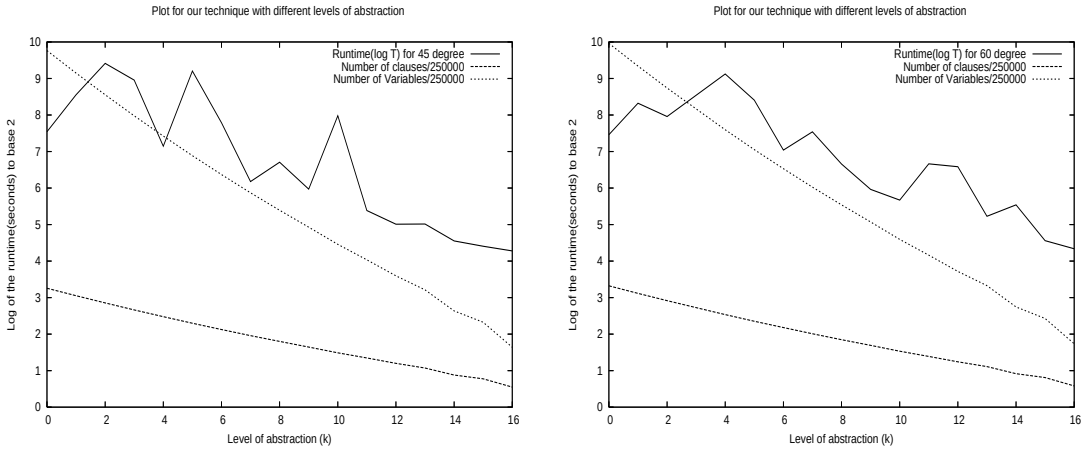
Table 2 shows the run-times of our model checker on the TCAS example with different values of k. Most of the run-time is spent in SAT solving (MiniSat).

Phaver cannot handle this example due to non-linear invariants and guards; HSolver [8] cannot either as the invariants and guards involve flow rates. We ran HSolver on a simplified model given by Pappas et al [14] but it did not finish after an hour.

k	Runtime (sec) for different angles		
	$\Delta\phi = 30^\circ$	$\Delta\phi = 45^\circ$	$\Delta\phi = 60^\circ$
0	400.50	181.47	177.49
2	300.01	732.24	253.96
4	904.76	136.39	544.97
8	117.25	101.01	55.45
16	27.45	18.21	17.64

Table 2. Runtimes of our model checker for TCAS with varying angles.

Conclusion from Figure 6: In this plot, we illustrate how the run-time varies for different levels of abstraction. The x-axis represents the value of k in the k-abstraction. We considered different levels of abstraction for the TCAS example with different set of parameters (changing the deviation angle). There is an initial increase in runtime due to addition of extra flows which increase the size of the SAT instance generated by BMC and induction techniques. The different flows are represented using disjunction and hence, the resulting SAT instance increases with increase in possible flows. But there is also a gain due to decrease in the number of bits required to represent each variable as the level of abstraction increases. For larger values of k, this decrease in bit-vector encoding offsets the increase in size due to more possible flows. Thus, the SAT instances for larger abstractions are easier to solve and take less time as shown by Figure 6.



(a) $\Delta\phi = 45$ degree

(b) $\Delta\phi = 60$ degree

Fig. 6. Plot comparing runtimes of our technique for different levels of abstraction

8 Future Work

A next step would be extending this technique to handle non-linear flows. We would also like to extend it to hybrid systems with flow dependence between the variables. It would also be interesting to explore a combination of predicate abstraction based techniques with our method to be able to analyze even larger examples.

References

1. M. Agrawal, F. Stephan, P. S. Thiagarajan, and S. Yang. Behavioural approximations for restricted linear differential hybrid automata. In *HSCC*, pages 4–18, 2006.
2. M. Agrawal and P. S. Thiagarajan. Lazy rectangular hybrid automata. In *HSCC*, pages 1–15, 2004.
3. M. Agrawal and P. S. Thiagarajan. The discrete time behavior of lazy linear hybrid automata. In *HSCC*, pages 55–69, 2005.
4. R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems I*, LNCS 736, pages 209–229, 1992.
5. W. Chan, R. Anderson, P. Beame, and D. Notkin. Combining constraint solving and symbolic model checking for a class of a systems with non-linear constraints. In *CAV*, pages 316–327, 1997.
6. E. Clarke, A. Gupta, J. Kukula, and O. Strichman. SAT based abstraction-refinement using ILP and machine learning techniques. In *CAV'02*, LNCS, 2002.
7. E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In *CAV '00*, pages 154–169, London, UK, 2000. Springer-Verlag.
8. W. Damm, G. Pinto, and S. Ratschan. Guaranteed termination in the verification of ltl properties of non-linear robust discrete time hybrid systems. In *ATVA*, pages 99–113, 2005.
9. A. Deshpande, D. N. Godbole, A. G, and P. Varaiya. Design and evaluation tools for automated highway systems. In *Hybrid Systems*, pages 138–148, 1995.
10. G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *HSCC*, pages 258–273, 2005.
11. T. A. Henzinger and P. W. Kopke. Discrete-time control for rectangular hybrid automata. *TCS*, 221(1–2):369–392, 1999.
12. S. K. Jha, B. H. Krogh, J. Weimer, and E. M. Clarke. Iterative relaxation abstraction (ira) for linear hybrid automata. In *HSCC'07*, Lecture Notes in Computer Science, 2007(to appear).
13. C. Livadas, J. Lygeros, and N. A. Lynch. High-level modeling and analysis of tcas. In *RTSS '99*, page 115, Washington, DC, USA, 1999. IEEE Computer Society.
14. G. Pappas, C. Tomlin, and S. Sastry. Conflict resolution for multi-agent hybrid systems, 1996.
15. B. Potocnik, A. Bemporad, F. Torrisi, G. Music, and B. Zupancic. Hysdel Modeling and Simulation of Hybrid Dynamical Systems. In *MATHMOD Conference*, Vienna, Austria, Feb. 2003.
16. S. Ratschan and Z. She. Constraints for continuous reachability in the verification of hybrid systems. In *AISC*, pages 196–210, 2006.

17. TCAS201 Specification Datasheet. http://www.aeroflex.com/products/avionics/rf/datasheets/tcas201.pdf.
18. A. Tiwari. Approximate reachability for linear systems. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control HSCC*, volume 2623 of *LNCS*, pages 514–525. Springer, Apr. 2003.
19. A. Tiwari and G. Khanna. Series of abstractions for hybrid automata. In *HSCC*, pages 465–478, 2002.
20. F. A. A. US Department of Transportation. Introduction to TCAS II Version , November, 2000. http://www.arinc.com/downloads/tcas/tcas.pdf.