Formal Semantics of Hybrid Automata
Technical Report

Manasvi Saxena[1], Nishant Rodrigues[1], Xiaohong Chen[1], and Grigore Roşu[1]

[1] University of Illinois at Urbana Champaign
[2] {msaxena2,nishant2,xc3,grosu}@illinois.edu

Abstract. Hybrid Automata (HA) form the backbone of modeling systems with both discrete and continuous components. However, the semantics of HA are usually described loosely using Labeled Transition Systems (LTS), and reasoning about HA involves informal proofs over LTS. In this paper, we propose a formally rigorous, concise, and semantically correct definition of HA in Matching Logic (ML), which is a uniform logic for programming languages design and verification. We show how our definition allows formal reasoning about HA using ML's proof system. Our approach exposes HA to a rich set of tools that operate over ML and provide a sound logical basis for various techniques like Deductive Verification, Monitoring and Runtime Verification.

1 Introduction

The framework of *Hybrid Automata* is widely used to model dynamical systems - systems that combine discrete dynamics with continuous dynamics. *Cyber Physical Systems* (CPS), or systems that combine cyber capabilities (computation, communication and control) with physical capabilities (motion or other physical processes) are examples of such dynamical systems. In recent years, CPS are increasingly being employed to tackle challenges in domains like medicine, transportation and energy. The use of CPS in safety critical applications, however, makes their correctness and reliability extremely important. Moreover, the inherent complexity of CPS make their analysis challenging. HA have been instrumental in addressing these challenges by providing an intuitive mechanism for modeling CPS, and have become central to a rich ecosystem of CPS analysis tools [5,9,6,10].

In this paper, we propose a *formally rigorous*, and *semantically correct* definition of HA. Our approach is centered around embedding HA in Matching Logic (ML) [4]. Given a HA H as described in [8], we derive a ML theory Γ^H from H such that Labeled Transition System (LTS) of H is an ML-model of Γ^H. Intuitively, Γ^H can be thought of as an *axiomatization* of H's transition relation. Since Γ^H is an ML theory, we can use ML's proof system to formally reason about H.

We now briefly describe the paper's layout. In 2 we describe relevant mathematical preliminaries. In 3 we describe the process of embedding HA in ML, and present its correctness in 1.A . In 4.2 we discuss monitoring and runtime

verification using the embedding from 3. In 5 we discuss directions for future work and conclude.

2 Preliminaries

2.1 Hybrid Automata

Hybrid Automaton H [8] consists of: (1) A finite set $X = \{x_1, \ldots, x_n\}$ of *real valued variables*, a finite set $\dot{X} = \{\dot{x_1}, \ldots, \dot{x_n}\}$ of *dotted variables* for derivatives, and a finite set $X' = \{x'_1, \ldots, x'_n\}$ of *primed variables* for final values of discrete changes. (2) A finite *directed multigraph* (V, E), with vertices referred to as *control modes* and edges as *control switches*. (3) For all $v \in V$, predicates $\mathit{init}(v), \mathit{inv}(v)$ and $\mathit{flow}(v)$ called *initial*, *invariant* and *flow* conditions respectively. $\mathit{init}(v)$ and $\mathit{inv}(v)$ are defined using variables in X, while $\mathit{flow}(v)$ is defined using those in $X \cup \dot{X}$. (4) For all $e \in E$, predicate $\mathit{jump}(e)$ called *jump* condition, defined over free variables in $X \cup X'$. (5) A finite set Σ_H of events, and an edge labeling function *event*: $E \to \Sigma_H$, assigning to each *control switch* $e \in E$ an event $s \in \Sigma_H$.

2.2 Labeled Transition Systems

For the purpose of this discussion, a *Labeled Transition System* (LTS) S consists of (1) A possibly infinite set of states Q, and set $Q_0 \subseteq Q$ of initial states. (2) A possibly infinite set A of labels, and for each $a \in A$, a binary relation $\xrightarrow{a}$ on Q.

For a given *Hybrid Automaton* H, the *timed transition system* S^t_H is a LTS that captures transition semantics of H [8], defined as: (1) $Q, Q_0 \subseteq (V \times \mathit{Real}^n)$. Say, $(v, \boldsymbol{x}) \in Q$ iff $\mathit{inv}(v)[\boldsymbol{x}/X]$ and $(v, \boldsymbol{x}) \in Q_0$ iff $\mathit{init}(v)[\boldsymbol{x}/X]$. Let label set $A = \Sigma_H \cup \mathbb{R}_{\geq 0}$. (2) For given $\sigma \in \Sigma_H$, say $(v, \boldsymbol{x}) \xrightarrow{\sigma} (v', \boldsymbol{x}')$ iff (a) there exists *control switch* e with source v and destination v'. (b) $\mathrm{jump}(e)[\boldsymbol{x}/X, \boldsymbol{x}'/X']$ holds, and $\mathrm{event}(e) = \sigma$. (3) For each non negative real $\delta \in \mathbb{R}_{\geq 0}$, say $(v, \boldsymbol{x}) \xrightarrow{\delta} (v', \boldsymbol{x}')$ iff (a) $v = v'$ (b) there exists a differentiable function $f\colon [0, \delta] \to \mathbb{R}^n$ where $\dot{f}\colon (0, \delta) \to \mathbb{R}^n$ is the first order derivative of f such that $f(0) = \boldsymbol{x}$ and $f(\delta) = \boldsymbol{x}'$ (c) $\forall \epsilon \in (0, \delta)$, $\mathit{inv}(v)[f(\epsilon)/X]$ and $\mathit{flow}(v)[f(\epsilon)/X, \dot{f}(\epsilon)/\dot{X}]$ hold.

2.3 Matching Logic

Matching Logic [14,4] is a logic for specifying static structures, logic constraints, and dynamic properties. ML serves as the foundation of the $\mathbb{K}$ framework [1], which is a language semantic framework where the formal semantics of languages are defined and all language tools of the given languages are automatically generated in a correct-by-construction manner. In this section, we give a compact overview of ML.

In short, ML is a many-sorted FOL variant combined with the least fixpoint μ-binder. Unlike FOL, ML makes no distinction between *functions* and *predicates*, but only has *symbols* that uniformly construct *patterns*, which can capture static structure, dynamic properties, and FOL and modal logic constraints.

ML *syntax* is parametric in a *signature* $\mathbb{\Sigma} = (S, \textsc{Var}, \Sigma)$ consisting of (1) a *sort set* S; (2) a disjoint union set $\textsc{Var} = \textsc{EVar} \cup \textsc{SVar}$ of *element variables* in $\textsc{EVar}$ denoted $x{:}s, y{:}s$ etc. and *set variables* in $\textsc{SVar}$ denoted $X{:}s, Y{:}s$ etc; and

(3) an $(S^* \times S)$-indexed set $\Sigma = \{\Sigma_{s_1 \ldots s_n, s}\}_{s_1,\ldots,s_n \in S}$ of *many-sorted symbols*. *Patterns* are inductively defined by the following grammar:

$$\varphi_s ::= x{:}s \in \text{EVar} \mid X{:}s \in \text{SVar} \mid \varphi_s \wedge \varphi_s \mid \neg\varphi_s \mid \forall x{:}s' \,.\, \varphi_s$$
$$\mid \sigma(\varphi_{s_1}, \ldots, \varphi_{s_n}) \text{ with } \sigma \in \Sigma_{s_1 \ldots s_n, s} \mid \mu X{:}s \,.\, \varphi_s \text{ where } \varphi_s \text{ is positive in } X{:}s$$

We say φ_s is positive in $X{:}s$ if all occurrences of $X{:}s$ are under an even number of negations in φ_s. Sorts are omitted when they are irrelevant or can be inferred from the context. Common propositional connectives can be define in the usual way. We define $\exists x \,.\, \varphi \equiv \neg\forall x \,.\, \neg\varphi$, and $\nu X \,.\, \varphi \equiv \neg\mu X \,.\, \neg\varphi[\neg X/X]$ (the greatest fixpoint), where "$\_[\_/\_]$" is the standard notion of capture-avoiding substitution as in FOL. Define $\top \equiv \exists x \,.\, x$ as the pattern that matches all elements and $\bot \equiv \neg\top$ the one that matches nothing. Let $\text{Pattern} = \{\text{Pattern}_s\}_{s \in S}$ be the S-indexed set of all patterns, modulo α-renaming of bound variables.

Given the signature $\mathbb{\Sigma}$ as above, a *model* $M = (\{M_s\}_{s \in S}, \_^M)$ consists of (1) a nonempty carrier set M_s for each $s \in S$; and (2) an interpretation $\sigma_M \colon M_{s_1} \times \cdots \times M_{s_n} \to \mathcal{P}(M_s)$ for each $\sigma \in \Sigma_{s_1 \ldots s_n, s}$, where $\mathcal{P}(M_s)$ is the powerset of M_s. Note that symbols are interpreted as relations. Standard FOL models are special cases of ML models where $|\sigma_M(a_1, \ldots, a_n)| = 1$ for all $a_i \in M_{s_i}$, $1 \leq i \leq n$. Partial FOL models are also special cases where $|\sigma_M(a_1, \ldots, a_n)| \leq 1$. We tacitly use the same σ_M to mean its *pointwise extension*, $\sigma_M \colon \mathcal{P}(M_{s_1}) \times \cdots \times \mathcal{P}(M_{s_n}) \to \mathcal{P}(M_s)$, defined as: $\sigma_M(A_1, \ldots, A_n) = \bigcup\{\sigma_M(a_1, \ldots, a_n) \mid a_1 \in A_1, \ldots, a_n \in A_n\}$ for all $A_i \subseteq M_{s_i}$, $1 \leq i \leq n$. An *(M-)valuation* $\rho \colon \text{Var} \to (M \cup \mathcal{P}(M))$ is a function s.t. $\rho(x) \in M_s$ for $x{:}s \in \text{EVar}$ and $\rho(X) \in \mathcal{P}(M_s)$ for $X{:}s \in \text{SVar}$, i.e., element variables are evaluated to elements and set variables are evaluated to sets. Its *extension* $\bar{\rho} \colon \text{Pattern} \to \mathcal{P}(M)$ is defined inductive over patterns as follows:

$$\bar{\rho}(x) = \{\rho(x)\} \quad \bar{\rho}(\neg\varphi) = M_s \setminus \bar{\rho}(\varphi) \quad \bar{\rho}(\varphi_1 \wedge \varphi_2) = \bar{\rho}(\varphi_1) \cap \bar{\rho}(\varphi_2)$$
$$\bar{\rho}(X) = \rho(X) \quad \bar{\rho}(\forall x \,.\, \varphi) = \bigcap_{a \in M_{s'}} \overline{\rho[a/x]}(\varphi) \quad \bar{\rho}(\sigma(\varphi_1, ..., \varphi_n)) = \sigma_M(\bar{\rho}(\varphi_1), ..., \bar{\rho}(\varphi_n))$$

$\bar{\rho}(\mu X \,.\, \varphi) = \mu\mathcal{F}^{\rho}_{\varphi,X}$, where $\mathcal{F}^{\rho}_{\varphi,X} \colon \mathcal{P}(M_s) \to \mathcal{P}(M_s)$ is the monotone function given as $\mathcal{F}^{\rho}_{\varphi,X}(A) = \overline{\rho[A/X]}(\varphi)$ for all $A \subseteq M_s$ and $\mu\mathcal{F}^{\rho}_{\varphi,X}$ denotes its the least fixpoint.

Here, "$\setminus$" is set difference; $\rho[a/x]$ (resp. $\rho[A/X]$) is the valuation ρ' with $\rho'(x) = a$ (resp. $\rho'(X) = A$) and $\rho'(y) = \rho(y)$ for $y \neq x$ (resp. $\rho'(Y) = \rho(Y)$ for $Y \neq X$). We define $M \vDash \varphi$ iff $\bar{\rho}(\varphi) = M$ for all ρ. Let Γ be a pattern set called *axiom set*. We define $M \vDash \Gamma$ iff $M \vDash \varphi$ for all $\varphi \in \Gamma$ and $\Gamma \vDash \varphi$ iff $M \vDash \varphi$ for all $M \vDash \Gamma$. A *theory* is a pair $(\mathbb{\Sigma}, \Gamma)$, often abbreviated as Γ, where Γ is a set of $\mathbb{\Sigma}$-axioms.

In this paper, we only need the FOL fragment of ML, meaning that most ML symbols occurred in this paper can be safely understood as FOL functions or predicates. The only exception is the "one-path next" symbol "$\bullet$" (defined in Section 3.2) which we use to capture the transition relations in transition systems. We will elaborate it in details when we define it.

3 Matching Logic Embedding of Hybrid Automata

Given an HA H, we derive an ML theory Γ^H from H, and prove, in Appendix 1.A, that the theory captures *faithfully* the behavior of H.

3.1 Augmented Theory of Reals

We assume Γ^{Real} - the ML theory of *Real Closed Fields* [15]. We define $\Gamma^{Real^\dagger}$ which extends Γ^{Real} with *sorts* $Real^n, Real \dashrightarrow Real^n$ representing n-ary products of *Reals* and relations between *Reals* to n-ary products of *Reals* respectively. and Γ^{Real}'s *signature* with *symbol* app $\in \Sigma_{Real \dashrightarrow Real^n\ Real, Real^n}$. We briefly describe the need for $\Gamma^{Real^\dagger}$, and its need for embedding H in ML.

Consider $\mathcal{M}_{Real^\dagger}$ to be an ML model of $\Gamma^{Real^\dagger}$, with $\mathbb{R}$ and $\mathbb{R}^n$ as the carrier sets of sorts *Real* and $Real^n$ respectively. Intuitively, the sort $Real \dashrightarrow Real^n$ can be thought of as the sort of relations between *Real* and $Real^n$. Since any binary relation R can be described as a set S_R, where $(a, b) \in S_R$ iff $a\,R\,b$, any relation between $\mathbb{R}$ and $\mathbb{R}^n$, is simply a subset of $\mathbb{R}^{n+1}$. Thus, the carrier set of sort $Real \dashrightarrow Real^n$ would be $\mathcal{P}(\mathbb{R}^{n+1})$. For more information on ML models, we refer the reader to [4]. The sort $Real \dashrightarrow Real^n$ and symbol $app \in \Sigma_{Real \dashrightarrow Real^n\ Real, Real^n}$ give us the ability to quantify over relations that behave functionally in an interval. For instance, consider reals a, b, and a variable $f{:}Real \dashrightarrow Real^n$. Say we have ML pattern $\varphi \equiv \forall a \leq t{:}Real \leq b\,.\,(\exists y{:}Real^n\,.\,(app(f, t) = y))$. In a model of $\Gamma^{Real^\dagger}$, the (extended) evaluation of φ will be $\top$ iff the evaluation of f is a relation that behaves as a function in the closed interval $[a, b]$. In section 3.2, we use aforementioned sort and symbol in the axiomatization of *flow* predicates.

We define syntactic sugar for frequently used $\Gamma^{Real^\dagger}$-terms. Say we have variables $f{:}Real \dashrightarrow Real^n$, $l{:}Real_{\geq 0}$, $u{:}Real_{\geq 0}$, $x{:}Real_{\geq 0}$, and $y{:}Real^n$, then we say $f(x)$ instead of $\text{app}(f, x)$; $f{:}[l, u]$ instead of $\forall l \leq t \leq u\,.\,(\exists y\,.\,f(t) = y)$ and $f{:}(l, u)$ instead of $\forall l < t < u\,.\,(\exists y\,.\,f(t) = y)$. Given $f, \dot{f}{:}Real \dashrightarrow Real^n$, we say $\dot{f}$ is the first order derivative of f in interval l, u as:

$$\varphi_{isDerivativeOf}(f, \dot{f}, l, u) \;\equiv\; f{:}[l, u] \wedge \dot{f}{:}(l, u) \wedge \forall l < t{:}Real_{>0} < u$$
$$\forall \epsilon{:}Real_{\geq 0}\,.\,\left(\exists \delta{:}Real_{\geq 0}\,.\,\left(\forall h{:}Real_{\geq 0}\; < \delta\,.\,\left(\left|\frac{f(t+h) - f(t)}{h} - \dot{f}(t)\right| \leq \epsilon\right)\right)\right) \tag{1}$$

Intuitively, the valuation of $\varphi_{isDerivativeOf}$ will be $\top$ iff (1) valuation of f is a relation that behaves functionally in closed interval $[a, b]$, and (2) valuation of $\dot{f}$ is a relation that behaves functionally in the open interval (a, b), and (3) at time $t \in (a, b)$, the valuation of $\dot{f}(t)$ is the first order derivative of f.

3.2 ML Theory of Hybrid Automata

In section 3.1 we defined $\Gamma^{Real^\dagger}$, an *augmented theory of Reals*. The ML embedding $\Gamma^H = (\mathbb{\Sigma}^H, A^H)$ of H as:

(Sorts) $\mathbb{\Sigma}^H = (\{V, E, State, Transition, Event\} \cup S^{Real^\dagger}), \text{VAR}^H, \Sigma^H)$, where $S^{Real^\dagger}$ is the sort-indexed set of $\Gamma^{Real^\dagger}$ sorts. Sorts V, E are understood to represent H's control graph. We assume Σ_V and Σ_E contain symbols σ_v and σ_e for every $v \in V$ and $e \in E$ respectively. Similarly, we say *Event* is the sort of H-events, and assume Σ_{Event} to be the corresponding (finite) signature of H-events. Sort State is assumed to a subsort of $V \times Real^n$. The sort *Transition* represents both continuous and discrete transitions of H. We assume appropriate axioms such that in any ML-model, the carrier sets of sorts *Event* and $Real_{\geq 0}$ are subsets of the carrier set of sort *Transition*. Intuitively, *State* and $Real_{\geq 0}$ can be thought of as subsorts of *Transition*.

(Symbols) We define symbols first $\in \Sigma_{State,V}$ and second $\in \Sigma_{State,Real^n}$, with appropriate axioms such that they behave as projection functions. Say we have s:*State*, then we use s_v and s_x as syntactic sugar to refer to first(s) and second(s) respectively. Intuitively, the sort *State* can be thought of as a subsort of $V \times Real^n$, where first and second are projections into V and $Real^n$. In the remainder of this work, we may use (v, x):*State* instead of s:*State*, where it is understood that $v = s_v$ and $x = s_x$. We also define symbol $\bullet \in \Sigma_{Transition\,State,State}$. The $\bullet$ (one path next) symbol is the idiomatic way of capturing transition sytems in ML. We refer the reader to [4] for futher information. We tacitly assume that *flow*,*inv*,*init* and *jump* behave similarly for ML-symbols corresponding to vertices and edges in H's *control graph*. In other words, given $v \in V$ and corresponding ML symbol $\sigma_v \in \Sigma_V$, $\mathit{flow}(v) = \mathit{flow}(\sigma_v)$.

(Axioms) A^H, the set of Γ^H axioms is defined as:

(Invariant) For each $\sigma_v \in \Sigma_V$, we define an axiom:

$$\forall x : Real^n \,.\, (\mathit{inv}(\sigma_v)[x \,/\, X] \leftrightarrow \exists (v', x'){:}State \,.\, v' = \sigma_v \wedge x' = x) \tag{2}$$

(Jump) For each $\sigma_e \in \Sigma_E$, we define an axiom:

$$\begin{aligned}\forall (v, x), (v', x'){:}State \,.\, ((v, x) \rightarrow \bullet(\mathit{event}(\sigma_e), (v', x'))) \leftrightarrow \\ (\mathit{jump}(\sigma_e)[x \,/\, X, x' \,/\, X'] \wedge (\text{source}(\sigma_e) = v) \wedge (\text{destination}(\sigma_e) = v'))\end{aligned} \tag{3}$$

(Flow) For each $\sigma_v \in \Sigma_V$, we define an axiom:

$$\begin{aligned}\forall \delta{:}Real_{\geq 0} \,.\, \forall (v, x), (v', x'){:}State \,.\, (v = \sigma_v) \wedge ((v, x) \rightarrow \bullet(\delta, (v', x'))) \leftrightarrow (v = v') \\ \wedge \exists f, \dot{f}{:}Real \dashrightarrow Real^n \,.\, \varphi_{isDerivativeOf}(f, \dot{f}, 0, \delta) \wedge (f(0) = x \wedge f(\delta) = x') \\ \wedge \forall 0 < \epsilon < \delta \,.\, (\mathit{inv}(\sigma_v)[f(\epsilon) \,/\, X] \wedge \mathit{flow}(\sigma_v)[f(\epsilon) \,/\, X, \dot{f}(\epsilon) \,/\, \dot{X}])\end{aligned} \tag{4}$$

(Init) For each $\sigma_v \in \Sigma_V$ we also define aliases:

$$\mathit{init}_{\sigma_v} \equiv \exists (v, x){:}State \,.\, \mathit{init}(\sigma_v)[x' \,/\, X] \wedge (v = \sigma_v) \tag{5}$$

4 Applications

4.1 Verification

HA are commonly used to model safety-critical hybrid systems like cars, planes and medical devices, where safety is important. As a prelude to talking about safety, we briefly describe *trace semantics* of HA, and the *Reachability Problem*. Given a HA H and its LTS $S^t_H = (Q, Q_0, A)$, a *duration* is assigned to each transition $a \in A$. For discrete events $\sigma \in \Sigma_H$, the duration is 0. For reals $\delta \in \mathbb{R}_{\geq 0}$, the duration of $q \xrightarrow{\delta} q'$ is δ, where $q, q' \in Q$ [8]. If q_0 is the initial state and $q_0 \xrightarrow{a_0} q_1$, then $\langle a_i, q_i \rangle_{i \geq 1}$ is an *initialized* trajectory.

In [8], the *reachability problem* a fundamental subtask for safety verification, but is only described informally as "existence of an initialized trajectory that visits a state of the form (v', x')". Our embedding however, allows us to formally

and concisely define the *Reachability Problem*, for a given HA H, its ML embedding Γ^H, and initial state (v, x) as $(v, x) \rightarrow \Diamond(v', x')$ while safety can be then be stated in terms of the reachability problem as $\psi_{\text{safety}} \equiv \neg((v, x) \rightarrow \Diamond\varphi_{\text{unsafe}})$ Intuitively, the above formalization says that given an initial state (v, x), it must not be the case that an initialized trajectory will *eventually* reach an unsafe state (denoted by φ_{unsafe}). For more information on modal operators in ML, we refer the reader to [4]. Deductive verification is then proving $\Gamma_H \vdash \psi_{\text{safety}}$. Deductive verification of Hybrid Systems isn't a novel idea. In [13,16], the authors present *Differential Dynamic Logic* and *Hybrid Hoare Logic* respectively - logics for verification of Hybrid Systems. These approaches however use *Hybrid Programs* and *Hybrid CSP* respectively to model hybrid systems. This places an additional requirement of understanding a different modeling approach instead of widely used HA. On the other hand our approach is centered around HA which eliminates the need for learning new modeling approaches for deductive verification.

4.2 Monitoring and Runtime Verification

Given a hybrid system S, and a HA H that models S, we can use Γ^H, the embedding of H in ML, and the ML proof system to deductively verify properties (like safety, liveness, et.c.) about H. However, verification doesn't guarantee that S behaves as expected, since H may not be an adequate model of S, as it is impossible to completely model real word physics. Thus it is vital, even after deductive verification, to monitor S's execution for properties like *model compliance* and *safety*. In [12,3], the authors build on [13] to present monitoring mechanisms for stronger safety guarantees. However, since the aforementioned frameworks can only generate monitors for *Hybrid Programs*, and cannot be directly used with HA.

We now attempt to formally define monitoring in the context of HA. In Section 5, we briefly discuss approaches for efficient monitoring. We assume an 0-*duration* event is generate on every discrete transition, and at time interval t_{SR}, some pre determined sampling rate for continuous transitions. We also assume that for each 0-*duration* event, the discrete transition associated with the event is known.

Model Compliance Consider an initialized trajectory $\pi = \langle a_i, q_i \rangle_{i \geq 1}$. Recall a_i is a *duration*, and q_i a state in the LTS of HA. We say $\pi \vDash_{trajectory} \Gamma_H$ iff (a) Let $\pi = \pi' \langle a_i, q_i \rangle$, where π' is sub-trajectory. If π' is non empty then $\pi' \vDash_{trajectory} \Gamma_H$ (b) For events with *duration* $a_i \in \mathbb{R}_{>0}$, $\Gamma_H \vdash (q_{i-1} \rightarrow \bullet(a_i, q_i))$. (c) For 0-*duration* (discrete) events, with dicrete transition σ, there must exists an intermediate event $\langle \delta, q_i' \rangle$ with *duration* $\delta \in \mathbb{R}_{\geq 0}$ such that $\Gamma_H \vdash (q_{i-1} \rightarrow \bullet(\delta, q_i')) \wedge (q_i' \rightarrow \bullet(\sigma, q_i))$.

Safety In section 4.1 we presented ψ_{safety} - a formalization of safety using the *Reachability Problem*. Intuitively, φ_{unsafe} is a unsafe state that an initialiazed trajectory must not reach to be considered safe. Formally, we say an initialiazed trajectory $\pi = \langle a_i, q_i \rangle_{i \geq 1}$ is a safe iff (a) Initial state $\Gamma_H \vdash \neg\lceil q_0 \rightarrow \varphi_{\text{unsafe}} \rceil$. (b) Let $\pi = \pi' \langle a_i, q_i \rangle$, where π' is sub-trajectory. If π' is non empty then π' must be a safe trajectory. (c) The most recent observed event $\langle a_i, q_i \rangle$ must not match the unsafe pattern, i.e. $\Gamma_H \vdash \neg\lceil q_i \rightarrow \varphi_{\text{unsafe}} \rceil$.

Note that while monitoring, we can produce ML *proof objects* certifying how each event adheres to the the property being monitored.

5 Future Work and Conclusion

In the immediate future, we plan to develop efficient algorithms that accomplish monitoring as described in section 4.2. In [7] the authors presented an algorithm for efficiently monitoring propositional LTL formulas by using the LTL property itself as a monitor, and rewriting the property w.r.t. every event to obtain a derived formula to be used as a monitor. We plan to extend their approach to our ML-embedding of Transition Systems, and derive efficient algorithms for monitoring Transition Systems expressed in ML, of which our HA embedding is a special case.

In the long term, we plan to identify classes of HA for which generation of effecient monitors can be automated. We plan to use and extend the existing infrastructure of RV-Monitor [11,2], to implement and test our approach using examples. Our choice is driven by RV-Monitor's existing support for instrumenting LLVM and ROS. LLVM support is needed to instrument simulation of Hybrid Systems, while ROS (widely used for embedded systems) support allows monitoring actual ROS based Hybrid Systems.

In conlusion, this short paper presents a faithful (Appendix 1.A) embedding of HA in ML. Unlike other logical formalizations, our embedding can express can express all classes of HA. Importantly our work provides a framework for deductive verification of Hybrid Systems without going through alternative modeling approach and logics.

We also provided concise and formal definitions of important but informally defined problems from [8] such as the *Reachability Problem* and *safety*. We then formally defined monitoring *model compliance* and *safety*. Since we our definitions are expressed in a well understood logic, we hope to be able to produce proof objects certifying compliance of an observed trajectories with properties of interest. Finally, as future work, we present ideas for efficient online monitoring techniques that can be used with offline deductive verification to provide strong safety guarantees about Hybrid Systems.

References

1. K framework. https://www.kframework.org.com/
2. Runtime verification monitor. https://www.runtimeverification.com/monitor/
3. Bohrer, B., Tan, Y.K., Mitsch, S., Myreen, M.O., Platzer, A.: Veriphy: Verified controller executables from verified cyber-physical system models. In: Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (2018)
4. Chen, X., Roşu, G.: Matching μ-logic. In: Proceedings of the 34th Annual IEEE/ACM Symposium on Logic in Computer Science (2019)
5. Duggirala, P.S., Mitra, S., Viswanathan, M., Potok, M.: C2e2: A verification tool for stateflow models. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 68–82. Springer (2015)

6. Frehse, G.: Phaver: Algorithmic verification of hybrid systems past hytech. In: International workshop on hybrid systems: computation and control. pp. 258–273. Springer (2005)
7. Havelund, K., Rosu, G.: Monitoring programs using rewriting. In: Proceedings of the 16th IEEE International Conference on Automated Software Engineering. ASE '01 (2001)
8. Henzinger, T.A.: The Theory of Hybrid Automata, pp. 265–292. Springer Berlin Heidelberg (2000)
9. Henzinger, T.A., Ho, P.H., Wong-Toi, H.: Hytech: A model checker for hybrid systems. In: International Conference on Computer Aided Verification. pp. 460–463. Springer (1997)
10. Jin, X., Donzé, A., Deshmukh, J.V., Seshia, S.A.: Mining requirements from closed-loop control models. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **34**(11), 1704–1717 (2015)
11. Luo, Q., Zhang, Y., Lee, C., Jin, D., Meredith, P.O., Serbanuta, T.F., Rosu, G.: Rv-monitor: Efficient parametric runtime verification with simultaneous properties. In: Proceedings of the 14th International Conference on Runtime Verification (September 2014)
12. Mitsch, S., Platzer, A.: Modelplex: verified runtime validation of verified cyber-physical system models. Formal Methods in System Design **49**(1), 33–74 (Oct 2016). https://doi.org/10.1007/s10703-016-0241-z
13. Platzer, A.: Logics of dynamical systems. In: Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science. pp. 13–24 (2012)
14. Roşu, G.: Matching logic. Logical Methods in Computer Science **13**(4), 1–61 (December 2017)
15. Tarski, A.: A decision method for elementary algebra and geometry. Journal of Symbolic Logic **14**(3) (1949). https://doi.org/10.2307/2267068
16. Zhan, N., Wang, S., Zhao, H.: Hybrid Hoare Logic, pp. 91–105. Springer International Publishing, Cham (2017)

Appendix 1.A Proof of Faithfulness

Let $S^t_H = (Q, Q_0, A)$ be the LTS expressing the transition semantics of H, and Γ^H be the ML embedding of H. We refer the reader to section 2.2 and [8] for more information on transition semantics of Hybrid Automata.

Lemma 1. *For every $q, q' \in Q$ and $a \in A$ $(v, x) \xrightarrow{a} (v', x')$ iff $\Gamma^H \vdash (v, x) \to \bullet(a, (v', x'))$.*

Since $A = \Sigma_H \cup \mathbb{R}_{\geq 0}$, where Σ_H is finite and $\Sigma_H \cap \mathbb{R}_{\geq 0} = \varnothing$, we have the following cases:

Discrete When $a \in \Sigma_H$, $(v, x) \xrightarrow{a} (v', x')$ iff there is a control switch $e \in E$ such that

(d_1) source$(e) = v$ and destination$(e) = v'$
(d_2) event$(e) = a$
(d_3) $\mathit{jump}(e)[x \,/\, X, x' \,/\, X]$ holds

We first prove ($\to$) direction. Assume $(v, x) \xrightarrow{a} (v', x')$. By assumption ($d_1$), ($d_2$) and ($d_3$) hold. Using the 3 (*Jump* axiomatization) from section 3.2, since (d_1), (d_2) and (d_3) hold, we get $\forall(v, x), (v', x'){:}\mathit{State}\,.\,(v, x) \to \bullet(\mathit{event}(e), (v', x'))$, which proves ($\to$) direction. Next we prove ($\leftarrow$) direction. Assume event$(e) = a$ and $\forall(v, x), (v', x'){:}\mathit{State}\,.\,(v, x) \to \bullet(\text{event}(e), (v', x'))$. Using 3 from section 3.2, we get
$(\mathit{jump}(e)[x \,/\, X, x' \,/\, X'] \wedge (\text{source}(e) = v) \wedge (\text{destination}(e) = v'))$. Thus, ($d_1$), ($d_2$) and ($d_3$) hold, which proves ($\leftarrow$) direction.

Continuous When $a \in \mathbb{R}_{\geq 0}$. $(v, x) \xrightarrow{a} (v', x')$ iff

(c_1) $v = v'$
(c_2) There exist functions $f : [0, a] \to \mathbb{R}^n$ and $\dot{f} : (0, a) \to \mathbb{R}^n$ such that $\dot{f}$ is the first-order derivative of f.
(c_3) $f(0) = x$ and $f(a) = x'$
(c_4) $\forall 0 < \epsilon{:}\mathbb{R}^n < a\,.\,\mathit{inv}(v)[f(\epsilon) \,/\, X]$ and $\mathit{flow}(v)[f(\epsilon) \,/\, X, \dot{f}(\epsilon) \,/\, \dot{X}]$ hold.

We first prove ($\to$) direction. Assume $(v, x) \xrightarrow{a} (v', x')$. By assumption ($c_1$), ($c_2$), ($c_3$) and ($c_4$) hold. Using 4 (*Flow* axiomatization) from section 3.2, we get, (c_1), (c_2) , (c_3) and (c_4) hold iff $(v, x) \to \bullet(a, (v', x'))$. For proof of ($\leftarrow$) we assume $(v, x) \to \bullet(a, (v', x'))$. Using *Flow*, we get , (c_1), (c_2), (c_3) and (c_4) hold, leading to $(v, x) \xrightarrow{a} (v', x')$.