Computing Bounded ϵ-Reach Set with Finite Precision Computations for a Class of Linear Hybrid Automata

Kyoung-Dae Kim Sayan Mitra P. R. Kumar

Department of Electrical and Computer Engineering and Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801 USA
{kkim50, mitras, prkumar}@illinois.edu

ABSTRACT

In a previous paper [6] we have identified a special class of linear hybrid automata, called *Deterministic Transversal Linear Hybrid Automata*, and shown that an ϵ-reach set up to a finite time, called a *bounded* ϵ-reach set, can be computed using infinite precision calculations. However, given the linearity of the system and the consequent presence of matrix exponentials, numerical errors are inevitable in computation. In this paper we address the problem of determining a bounded ϵ-reach set using variable finite precision numerical approximations. We present an algorithm for computing it that uses only such numerical approximations. As a consequence, the corresponding safety problem is decidable. We further develop an architecture for such bounded ϵ-reach set computation which decouples the fundamental computational algorithm for an ϵ-reach set with given parameter values from both the choice of several runtime adaptation needed by several parameters in the variable precision approximations.

Categories and Subject Descriptors

G.M [**Mathematics of Computing**]: Miscellaneous

General Terms

Theory, Algorithm

Keywords

Linear hybrid automata, reachability, transversal discrete transition, deterministic discrete transition

1. INTRODUCTION

It is well known that computing the exact reach set of a Hybrid Automaton (HA) is undecidable, in general. In [5, 9] several class of decidable HA have been identified. In the recent years, research in hybrid system verification has since focused on algorithms computing over-approximations of the reachable states of various classes of HAs. [1, 8]. In [4], for examples, two techniques, called *clock translation* and *linear phase-portrait approximation*, are proposed to compute an over-approximation of the reach set when the continuous dynamics of a HA is more general than a rectangular HA. In [3], a conservative over-approximation of a reach set of a HA is computed through the on-the-fly over-approximation of the phase portrait, which is a variation of the approximation in [4]. In [2], to solve a verification problem of a class of HA, called a polyhedral-invariant HA (PIHA), a finite state transition system, which is a conservative approximation of the original HA, is constructed through a polyhedral approximation of each sampled segment of the continuous state evolution between switching planes.

In [6], we have identified a class of hybrid automata, called *Deterministic Transversal Linear Hybrid Automate (DTLHA)*[1], and shown a new approach to compute an over-approximate reach set, with arbitrarily small approximation error ϵ up to a finite time from an initial state, which we shall refer to as a *bounded ϵ-reach set*. The class of DTLHA consists of linear systems with constant inputs, for which the linear dynamics as well as the constant input switch along the boundaries of polyhedra, and for which the discrete transitions involved are *deterministic* and *transversal* at each discrete transition time. Since the linear systems involve matrix exponentials, one however needs to carefully take into account the issue of numerical approximations. In this paper we address the problem of computing with variable finite precision numerical schemes. We show that one can determine a bounded ϵ-reach set, and thus the corresponding safety problem is decidable. We also present an algorithm that decouples the fundamental computation of the bounded ϵ-reach set from the numerical approximation issues. This algorithm is additionally more flexible in comparison to the method used in [6] in terms of allowing more efficient computational strategies.

2. PRELIMINARIES

We consider the problem of the computation of an approximate reach set of a special class of Hybrid Automaton (HA) under some assumptions on the discrete transitions. More precisely, for a given approximation parameter $\epsilon > 0$, and a terminal time t_f defined below, we wish to compute an ϵ-reach set for a class of HAs called Deterministic Transversal

[1]Abbreviated simply as DLHA in [6]

Linear Hybrid Automata. These are linear hybrid automata that satisfy a certain deterministic and transversal discrete transition assumption, where $t_f := \min\{\tau_N, T\}$, N is a given upper bound of the number of discrete transitions, τ_N is the time of the N-th discrete transition, and T is a given upper bound of time. We now describe the class of automata in greater detail.

We assume that the continuous state space $\mathcal{X} \subset \mathbb{R}^n$ is closed and bounded, and is partitioned into a collection of polyhedral regions, called *cells*, that is

$$\bigcup_{i=1}^{m} \mathcal{C}_i = \mathcal{X}, \quad s.t. \;\; \mathcal{C}_i^\circ \cap \mathcal{C}_j^\circ = \emptyset \quad for\ i \neq j, \tag{1}$$

where m is the size of the partition, each $\mathcal{C}_i$ is a polyhedron such that $\mathcal{C}_i^\circ \neq \emptyset$, where $\mathcal{C}_i^\circ$ is the interior of $\mathcal{C}_i$. Two cells $\mathcal{C}_i$ and $\mathcal{C}_j$ are said to be *adjacent* if the affine dimension of $\partial\mathcal{C}_i \cap \partial\mathcal{C}_j$ is $(n-1)$, or, equivalently, cells $\mathcal{C}_i$ and $\mathcal{C}_j$ intersect in an $(n-1)$-dimensional facet. Here $\partial\mathcal{C}_i$ denotes the boundary of $\mathcal{C}_i$. Two cells $\mathcal{C}_i$ and $\mathcal{C}_j$ are said to be *connected* if there exists a sequence of adjacent cells between $\mathcal{C}_i$ and $\mathcal{C}_j$.

DEFINITION 1. *An n-dimensional* Deterministic Transversal Linear Hybrid Automaton (DTLHA) $\mathcal{A}$ *is a tuple* $(\mathbb{L}, l_0, x_0, Inv, A, u)$ *satisfying the following properties. (a)* $\mathbb{L}$ *is a finite set of* locations *or* discrete states*;* $l_0 \in \mathbb{L}$ *is the initial location; and* $x_0 \in \mathbb{R}^n$ *is the initial* continuous state*: The state space is* $\mathbb{L} \times \mathbb{R}^n$*, and an element* $(l, x) \in \mathbb{L} \times \mathbb{R}^n$ *is called a* state *of* $\mathcal{A}$*. (b)* $Inv : \mathbb{L} \to 2^{\mathcal{C}}$ *is a function that maps each location to a set of cells* [2]*, called an* invariant set *of a location, such that (i) for each* $l \in \mathbb{L}$*, all the cells in* $Inv(l)$ *are connected, (ii) for any two locations* $l, l' \in \mathbb{L}$*,* $Inv(l)^\circ \cap Inv(l')^\circ = \emptyset$*, and (iii)* $\cup_{l\in\mathbb{L}} Inv(l) = \mathcal{X}$*. (c)* $A : \mathbb{L} \to \mathbb{R}^{n\times n}$ *is a function that maps each location to an* $n \times n$ *matrix, and (d)* $u : \mathbb{L} \to \mathbb{R}^n$ *is a function that maps each location to an* n*-dimensional vector.*

In the sequel, for each $l_i \in \mathbb{L}$, we use A_i, u_i, Inv_i to denote $A(l_i)$, $u(l_i)$, and $Inv(l_i)$, respectively.

DEFINITION 2. *For a location* $l_i \in \mathbb{L}$*, a* trajectory *of duration* $t \in \mathbb{R}_{\geq 0}$ *for* $\mathcal{A}$ *with* n *continuous dimensions (or variables) is a continuous map* η *from* $[0, t]$ *to* $\mathbb{R}^n$*, such that (a)* $\eta(\tau)$ *satisfies the differential equation*

$$\dot{\eta}(\tau) = A_i\eta(\tau) + u_i, \tag{2}$$

(b) $\eta(\tau) \in Inv_i$ *for every* $\tau \in [0, t]$*.*

For such a trajectory η, its *duration* is t, and it is denoted by $\eta.dur$. We use Σ_i to denote the linear time invariant (LTI) system defined at a location l_i as in (2).

DEFINITION 3. *An* execution x *of* $\mathcal{A}$ *is defined as a continuous map* $x : [0, t] \to \mathbb{R}^n$ *which is the concatenation of a finite or infinite sequence of trajectories* $x = \eta_0\eta_1\eta_2\ldots$ *such that (a)* $t = \sum_k \eta_k.dur$*, (b)* $x(0) = \eta_0(0) = x_0 \in Inv_0$*,*

[2]Actually, to be precise, the invariant of a location is the union of such cells; however, we abuse the terminology slightly for ease of reading.

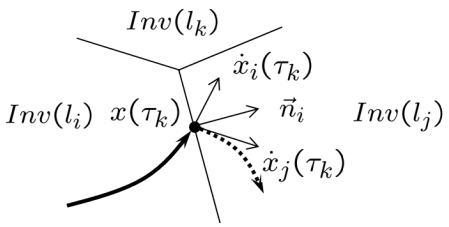

Figure 1: A deterministic and transversal discrete transition from a location l_i to a location l_j occurring at $x(\tau_k) \in \partial Inv(l_i) \cap \partial Inv(l_j)$.

(c) $x(\tau_k) = \eta_k(0) = \eta_{k-1}(\eta_{k-1}.dur)$ *for* $k \geq 1$*, (d)* $x(\tau) = \eta_{k-1}(\tau - \tau_{k-1})$ *for* $\tau \in [\tau_{k-1}, \tau_k]$*, where* $\tau_0 = 0$*, and* $\tau_k = \sum_{i=0}^{k-1} \eta_i.dur$ *for* $k \geq 1$*. Note that* τ_k *for* $k \geq 1$ *represents the* k*-th discrete transition between locations.*

DEFINITION 4. *For* $l_i, l_j \in \mathbb{L}$*, a* discrete transition *from* l_i *to* l_j *occurs at a continuous state* $x(\tau')$ *at time* τ'*, whenever* $x(\tau') \in Inv_i \cap Inv_j$ *and* $x(\tau') = \lim_{\tau \nearrow \tau'} x(\tau)$ *where* $x(\tau) \in (Inv_i)^\circ$ *for* $\tau \in (\tau' - \delta, \tau')$ *for some* $\delta > 0$*.*

DEFINITION 5. *A discrete transition is called a* deterministic discrete transition *if there is only one location* $l_j \in \mathbb{L}$ *to which* $x(\tau_k)$ *can make a discrete transition from* l_i*. Furthermore, for* $\epsilon > 0$*, we call a discrete transition a* transversal discrete transition *if the following condition is satisfied at* $x(\tau_k)$*:*

$$\langle \dot{x}_i(\tau_k), \vec{n}_i \rangle \geq \epsilon \quad \wedge \quad \langle \dot{x}_j(\tau_k), \vec{n}_i \rangle \geq \epsilon, \tag{3}$$

where $\vec{n}_i$ *is an outward normal vector of* ∂Inv_i *at* $x(\tau_k)$*, and* $\dot{x}_i(\tau_k) = A_i x(\tau_k) + u_i$*, and* $\dot{x}_j(\tau_k) = A_j x(\tau_k) + u_j$ *are the vector fields at* $x(\tau_k)$ *evaluated with respect to the continuous dynamics of location* l_i *and* l_j*, respectively.*

Fig. 1 illustrates a case when $x(\tau_k)$ satisfies such a deterministic and transversal discrete transition conditions. Note that if $x(\tau_k)$ satisfies a deterministic and transversal discrete transition condition, then $x(\tau_k)$ must make a discrete transition from a location l_i to the other unique location l_j. Furthermore, the *Zeno behavior*, an infinite number of discrete transitions within a finite amount of time, does not occur if a discrete transition is transversal discrete transition.

DEFINITION 6. *A continuous state in* $\mathcal{X}$ *is* reachable *if there exists some time* t *at which it is reached by some execution* x*.*

DEFINITION 7. *A* bounded reach set*, denoted as* $\mathcal{R}_t(x_0)$*, of a DTLHA* $\mathcal{A}$ *is defined to be the set of continuous states* $x(\tau)$ *reachable for some time* $\tau \in [0, t]$ *by some execution* x*, from* $x_0 \in Inv_0$*.*

DEFINITION 8. *Given* $\epsilon > 0$*, a set of continuous states* S *is called a* bounded ϵ-reach set *of a DTLHA* $\mathcal{A}$ *over a time interval* $[0, t]$ *from an initial state* x_0 *if* $\mathcal{R}_t(x_0) \subseteq S$ *and*

$$\forall y \in S, \quad \exists z \in \mathcal{R}_t(x_0) \; s.t. \; \|y - z\| \leq \epsilon. \tag{4}$$

The specific norm that we use in (4) as well as the sequel is the ℓ_∞-norm. Its advantage is that the neighborhoods it induces are polyhedra, in fact hypercubes.

Our results and computational procedures also address the following *Safety Problem*: Does the state enter the "unsafe" set of cells corresponding to a specified location within a specified finite time T? Our results show that except for the degenerate case where the state hits the boundary of the unsafe set for the first time at exactly T, the problem is decidable, and that our algorithm resolves this question.

Throughout this paper, we use $\mathcal{D}_t(\mathcal{P})$ to denote the set of states reached at time t from a set $\mathcal{P}$ at time 0. We also use $\mathcal{D}_t(\mathcal{P},\gamma)$ to denote an over-approximation of $\mathcal{D}_t(\mathcal{P})$ with a approximation parameter $\gamma > 0$, and calling it a γ-approximation of $\mathcal{D}_t(\mathcal{P})$ if it satisfies (i) $\mathcal{D}_t(\mathcal{P}) \subset \mathcal{D}_t(\mathcal{P},\gamma)$ and (ii) $d_H(\mathcal{D}_t(\mathcal{P}), \mathcal{D}_t(\mathcal{P},\gamma)) \leq \gamma$ where $d_H(\mathcal{P},\mathcal{Q})$ denotes the Hausdorff distance between two sets $\mathcal{P}$ and $\mathcal{Q}$. Note that $\mathcal{D}_0(\mathcal{P},\gamma)$ is simply a γ-approximation of the set $\mathcal{P}$.

3. BOUNDED ϵ-REACH SET OF A DTLHA

In this section, we first present the theoretical results for bounded ϵ-reach set computation developed in [6]. Then, we consider the issue of finite precision computation in computing such a bounded ϵ-reach set.

3.1 With Infinite Precision Calculations

The approach that we use to compute a bounded ϵ-reach set of a DTLHA from an initial state x_0 in [6] is that for given parameters δ and γ, and a sampling period h, the reach set is over-approximated by the union of $\mathcal{D}_t(\mathcal{B}_\delta(x_0),\gamma)$, a γ-approximation of a polyhedron of $\mathcal{D}_t(\mathcal{B}_\delta(x_0))$ at each sample time t where $\mathcal{B}_\delta(x_0)$ is a polyhedral neighborhood around x_0 and $\mathcal{D}_t(\mathcal{B}_\delta(x_0))$ is a set of states reached at time t from $\mathcal{B}_\delta(x_0)$. Here, $\mathcal{D}_t(\mathcal{B}_\delta(x_0),\gamma)$ is in fact an over-approximation of the reach set from $\mathcal{B}_\delta(x_0)$ in which the reach set from x_0 is in turn contained.

In this approach, the parameters δ, γ, and h are critical in computing a bouned ϵ-reach set. More precisely, h has be chosen for a given γ so that $\mathcal{D}_\tau(\mathcal{B}_\delta(x_0)) \subset \mathcal{D}_t(\mathcal{B}_\delta(x_0),\gamma)$ for $\tau \in [t, t+h]$. Furthermore, δ and γ have to be chosen so that (i) If there is a deterministic and transversal discrete transition, it can be determined by $\mathcal{D}_t(\mathcal{B}_\delta(x_0))$ and $\mathcal{D}_t(\mathcal{B}_\delta(x_0),\gamma)$ and, (ii) The diameter of a polyhedron $\mathcal{D}_t(\mathcal{B}_\delta(x_0),\gamma)$ at each sample time t has to be smaller than a given ϵ. In this regard, the following results were shown in [6].

LEMMA 1. *For a given $\gamma > 0$, if a sampling period h satisfies the following inequality in (5), then $\mathcal{D}_\tau(\mathcal{B}_\delta(x_0)) \subset \mathcal{D}_t(\mathcal{B}_\delta(x_0),\gamma)$ for $\tau \in [t, t+h]$ for each sample time t.*

$$h < \frac{\gamma}{\bar{v}} \tag{5}$$

where $\bar{v} := \max_{l_i \in \mathbb{L}}\{\|A_i\|\bar{x} + \|u_i\|\}$ and $\bar{x} := \max_{x\in\mathcal{X}} \|x\|$

LEMMA 2. *For a given $\epsilon > 0$, a given DTLHA $\mathcal{A}$, and a given location $l_i \in \mathbb{L}$, there exist $\delta > 0$, $\gamma > 0$, and $h > 0$ such that the followings hold:*

(i) $dia(\mathcal{D}_t(\mathcal{B}_\delta(x_0),\gamma)) < \epsilon$ for each sample time t,

(ii) If $x(\tau) \in (Inv_i \cap Inv_j)$ for some $l_j \in \mathbb{L}$, then $\mathcal{D}_t(\mathcal{B}_\delta(x_0)) \subset Inv_i^C$ where $x(\tau)$ is the reach set of $\mathcal{A}$ from x_0 and $\tau \in [t-h, t]$, and

(iii) Let $\mathcal{J}_{i,j} := \mathcal{D}_t(\mathcal{B}_\delta(x_0),\gamma) \cap Inv_i \cap Inv_j$ for some $l_j \in \mathbb{L}$. Suppose $\mathcal{J}_{i,j} \neq \emptyset$. If $x(\tau) \in \partial Inv_i$ such that $\tau \in (t-h, t)$, then there exists a $\delta' > 0$ such that $\mathcal{B}_{2\delta'}(x(\tau)) \subset (Inv_i \cup Inv_j)$. Furthermore, there exists a $\Delta > 0$ such that $h < \Delta$, $\mathcal{J}_{i,j} \subset \mathcal{J}'_{i,j}$, and

$$\bigcup_{y\in\mathcal{J}'_{i,j}} \mathcal{D}_t(y) \subset (Inv_j)^\circ \quad \forall t \in (\tau, \tau+\Delta), \tag{6}$$

where $\mathcal{J}'_{i,j} := \mathcal{B}_{\delta'}(x(\tau)) \cap Inv_i \cap Inv_j$.

In summary, the above results state the following: (i) For any given $\gamma > 0$, a sampling period $h > 0$ can be determined so that the reach set can be over-approximated. (ii) If there is a deterministic and transversal discrete transition, then this event can be determined by the over-approximation of the reach set with some sampling period h and some over-approximation parameters δ and γ. Now, we elaborate in more detail on the result given in Lemma 2, especially on (ii) and (iii), to develop some conditions which are used in Section 5.

LEMMA 3. *For a given location l_c, if $\mathcal{D}_t(\mathcal{B}_\delta(x_0)) \subset Inv_c^C$ and $\mathcal{D}_{t-h}(\mathcal{B}_\delta(x_0)) \subset Inv_c$ for some $\delta > 0$ and $h > 0$ where $\mathcal{B}_\delta(x_0)$ is a δ-neighborhood of the initial state x_0, then there is a discrete transition from the location l_c.*

PROOF. Note $\mathcal{D}_t(x_0) \in \mathcal{D}_t(\mathcal{B}_\delta(x_0))$ where $\mathcal{D}_t(x_0)$ is the reached state at time t from x_0. Similarly, $\mathcal{D}_{t-h}(x_0) \in \mathcal{D}_{t-h}(\mathcal{B}_\delta(x_0))$. From the hypothesis, $\mathcal{D}_t(x_0) \in Inv_c^C$ and $\mathcal{D}_{t-h}(x_0) \in Inv_c$. This implies that there exists $\tau \in (t-h, t)$ such that $\mathcal{D}_s(x_0) \in \partial Inv_c^\circ$ for $s \in [t-h, \tau)$ and $\mathcal{D}_s(x_0) \in \partial(Inv_c^C)^\circ$ for $s \in (\tau, t]$. Hence there is a discrete transition at some time $\tau \in [t-h, t]$. □

LEMMA 4. *For a given $\mathcal{P}_t$, suppose that there is a discrete transition from a location l_c to some other locations, i.e., $\mathcal{P}_t \subset Inv_c^C$ and $\mathcal{P}_{t-h} \subset Inv_c$ for a given polyhedron $\mathcal{P}_t$ and for some $h > 0$. Then the discrete transition is deterministic if there exists a location l_n such that $l_n \neq l_c$ and $\mathcal{P}_t \subset Inv_n$.*

PROOF. By Definition 5, the result is trivially true. □

LEMMA 5. *For a given $\mathcal{P}_t$, $\gamma > 0$, and $h > 0$ satisfying (5), suppose that there is a deterministic discrete transition from a location l_c to a location l_n. Then for any $\epsilon > 0$, the discrete transition is transversal if the following conditions hold.*

(i) $h < (dia(\mathcal{J}_{c,n})/2)/(2\bar{v})$,

(ii) $\mathcal{D}_0(\mathcal{J}_{c,n}, dia(\mathcal{J}_{c,n})/2) \subset Inv_c \cup Inv_n$,

(iii) $\langle \dot{x}_c, \vec{n}\rangle \geq \epsilon \wedge \langle \dot{x}_n, \vec{n}\rangle \geq \epsilon, \quad \forall x \in \mathcal{V}(\mathcal{J}'_{c,n})$

where $\mathcal{J}_{c,n} := \mathcal{D}_0(\mathcal{P}_t, \gamma) \cap Inv_c \cap Inv_n$, $\mathcal{J}'_{c,n} := \mathcal{D}_0(\mathcal{J}_{c,n}, dia(\mathcal{J}_{c,n})/2) \cap Inv_c \cap Inv_n$, $\bar{v}$ *is as defined in (5),* $\mathcal{V}(\mathcal{P})$ *is a set of vertices of a polyhedron* $\mathcal{P}$*, and* $\dot{x}_i$ *is the vector flow evaluated with respect to the LTI dynamics of location* $l_i \in \mathbb{L}$.

PROOF. Under the assumption, we first show that $\mathcal{J}_{c,n} \neq \emptyset$ and it is in fact an over-approximation of the deterministic discrete transition state $x_{\tau_k} \in Inv_c \cap Inv_n$. Since there is a deterministic discrete transition from l_c to l_n, $\mathcal{P}_{t-h} \subset Inv_c$ and $\mathcal{P}_t \subset Inv_n$. Thus $\mathcal{J}_{c,n} \neq \emptyset$. Notice that $\mathcal{P}_{t-h} \subset \mathcal{D}_0(\mathcal{P}_t, \gamma)$. In fact, $\cup_{z \in \mathcal{P}_{t-h}} x(\tau; z) \subset \mathcal{D}_0(\mathcal{P}_t, \gamma)$ for $\tau \in [t-h, t]$ where $x(\tau; z) := e^{A_c t} z + \int_0^\tau e^{A_c s} u_c ds$. Hence, if we let $\mathcal{D}_\tau(x_0)$ be a state reached from an initial state x_0 at time τ, then $\mathcal{D}_\tau \in \mathcal{D}_0(\mathcal{P}_t, \gamma)$ for $\tau \in [t-h, t]$. Furthermore, $\mathcal{D}_{\tau'}(x_0) \in \mathcal{D}_0(\mathcal{P}_t, \gamma) \cap Inv_c \cap Inv_n$ for some $\tau' \in (t-h, t)$ since $\mathcal{D}_{t-h}(x_0) \in Inv_c$ and $\mathcal{D}_t(x_0) \in Inv_n$.

Note that $\|x(h; z) - z\| \leq \bar{v}h$ for any $z \in \mathcal{J}_{c,n}$ and $h > 0$, where $x(h; z)$ is the state reached from z at time h under the LTI dynamics and $\bar{v}$ is as defined in (5). Then, (i) implies that $\|x(h; z) - z\| < dia(\mathcal{J}_{c,n})/2$. This in turn implies that $x(\tau; z) \in \mathcal{D}_0(\mathcal{J}_{c,n}, dia(\mathcal{J}_{c,n})/2)$ for $\tau \in [0, h]$. Therefore, $x(\tau; z) \in Inv_n$ for any $z \in \mathcal{J}_{c,n}$ and for all $\tau \in [0, h]$ by (ii) and (iii). □

3.2 With Finite Precision Calculations

The results in Section 3.1 rely on the assumption that the following quantities can be computed exactly:

- $x(t; x_0) = e^{At} x_0 + \int_0^t e^{As} u ds$.
- The intersection between a polyhedron and a hyperplane.

However, these exact computation assumptions can not be satisfied in practice and we can only compute each of these with arbitrarily small computation error. Hence, in this section, we extend the theory to incorporate the numerical computation errors.

3.2.1 Approximate Numerical Computations

In the sequel, we use $a(x, y)$ to denote an approximate computation of x with $y \in \mathbb{R}^+$ as an upper bound of the approximation error. The precise definition depends on the types of x:

- If x is a vector or a matrix, then $\|x - a(x, y)\| \leq y$.
- If x is a set, then $d_H(x, a(x, y)) \leq y$ where $d_H(x, z)$ is the Hausdorff distance.

In computing $x(t; x_0)$, we assume that a set of approximate computations, specifically, of $a(e^{At}, \sigma_e)$, $a(\int_0^t e^{A\tau} d\tau, \sigma_i)$, $a(A \cdot b, \sigma_p)$, and $a(u+v, \sigma_a)$, are available for given approximation errors $\sigma_e, \sigma_i, \sigma_p$, and σ_a. From these approximate computational capabilities, we can derive an upper bound on the approximation error, denoted as μ_x, for $x(t; x_0)$. We first note that, for all approximate computations $a(x, y)$ that are used for computing $x(t; x_0)$, we have $(x - y \cdot \mathbf{1}_{n \times m}) \leq a(x, y) \leq (x + y \cdot \mathbf{1}_{n \times m})$ where $x \in \mathbb{R}^{n \times m}$ and $\mathbf{1}_{n \times m}$ is an n by m matrix whose every element is 1. With this, we derive μ_x as follows.

$$e^{At} - \sigma_e \cdot \mathbf{1}_{n \times n} \leq a(e^{At}, \sigma_e) \leq e^{At} + \sigma_e \cdot \mathbf{1}_{n \times n},$$

$$\begin{aligned}\{(e^{At} - \sigma_e \cdot \mathbf{1}_{n \times n})|x_0| - \sigma_p \cdot \mathbf{1}_{n \times 1}\} &\leq a(e^{At} x_0, \sigma_p) \\ &\leq \{(e^{At} + \sigma_e \cdot \mathbf{1}_{n \times n} \cdot)|x_0| + \sigma_p \cdot \mathbf{1}_{n \times 1}\}.\end{aligned}$$

Similarly,

$$\begin{aligned}\{(\int_0^t e^{As} ds - \sigma_i \cdot \mathbf{1}_{n \times n}) \cdot |u| - \sigma_p \cdot \mathbf{1}_{n \times 1}\} &\leq a(\int_0^t e^{As} ds \cdot u, \sigma_p) \\ &\leq \{(\int_0^t e^{As} ds + \sigma_i \cdot \mathbf{1}_{n \times n}) \cdot |u| + \sigma_p \cdot \mathbf{1}_{n \times 1}\}.\end{aligned}$$

Hence, we have

$$x(t; x_0) - \delta_x \leq a(x(t; x_0), \delta_x) \cdot u \leq x(t; x_0) + \delta_x,$$

where $\delta_x := (2\sigma_p + \sigma_a) \cdot \mathbf{1}_{n \times 1} + (\sigma_e |x_0| + \sigma_i |u|) \cdot \mathbf{1}_{n \times n}$.

Now, we define μ_x as the maximum of δ_x over the continuous state space $\mathcal{X}$ and the control input domain $\mathcal{U}$,

$$\mu_x := \max_{x \in \mathcal{X}, u \in \mathcal{U}} \{\delta_x\}. \tag{7}$$

In addition to the above approximate computation $a(x, y)$ of $x(t; x_0)$, we also assume that an approximate intersection computation between two polyhedra $\mathcal{P}$ and $\mathcal{Q}$, $a(\mathcal{P} \cap \mathcal{Q}, \mu_c)$, is also available for given approximation error μ_c. Since $\mathcal{P} \cap \mathcal{Q}$ is also a polyhedron, $d_H(\mathcal{P} \cap \mathcal{Q}, a(\mathcal{P} \cap \mathcal{Q}, \mu_c)) \leq \mu_c$.

3.2.2 Some Preliminaries on Finite Precision Calculations

In this section, we extend the result given in Section 3.1 to relax the infinite precision computation assumption. Especially, we extend the results in Lemmas 1, 3, 4, and 5. We first discuss how the result in Lemma 1 can be extended under finite precision computation.

LEMMA 6. *Let* $\rho > 0$ *be an upper bound on the approximation errors of* $a(x(t; x_0), \rho)$ *and* $a(x(t+h; x_0), \rho)$ *for an LTI system* $\dot{x} = Ax + u$ *for some time* $t > 0$ *and* $h > 0$. *Then if* h *satisfies* $h < (\gamma - \rho)/(\|A\|\bar{x} + \|u\|)$ *for any given* $\gamma > \rho$*, where* $\bar{x}$ *is as defined in (5), then the following property holds:*

$$\bigcup_{z \in \mathcal{B}_\rho(x(t; x_0))} x(\tau; z) \subset \mathcal{B}_\gamma(x(t; x_0)), \quad \forall \tau \in [t, t+h]. \tag{8}$$

PROOF. Note that $a(x(t; x_0), \rho) \in \mathcal{B}_\rho(x(t; x_0))$ where $\mathcal{B}_\rho(y)$ is a ρ-neighborhood around y. Also note that for any $x(t) \in \mathcal{X}$,

$$\begin{aligned}\|x(t+h) - x(t)\| &\leq \int_t^{t+h} \|\dot{x}(s)\| ds \\ &\leq (\|A\|\bar{x} + \|u\|)h.\end{aligned}$$

Since $h < (\gamma - \rho)/(\|A\|\bar{x} + \|u\|)$, we see that $\|x(t+h) - x(t)\| < \gamma - \rho$ for any $x(t) \in \mathcal{X}$. Hence for any $z \in \mathcal{B}_\rho(x(t; x_0))$, $x(s; z) \in \mathcal{B}_{\gamma-\rho}(z)$ for $s \in [t, t+h]$. This implies that for any $z \in \mathcal{B}_\rho(x(t; x_0))$, $\|x(t; x_0 - x(s; z))\| \leq \|x(t; x_0) - z\| + \|z - x(s; z)\| \leq \gamma$. □

It is straightforward to extend the result of (i) in Lemma 2 as shown in the following Lemma.

LEMMA 7. *For given $\hat{\mathcal{P}}$, $\epsilon > 0$, and $\rho > 0$, if $dia(\hat{\mathcal{P}}) < \epsilon - \rho$, then $dia(\mathcal{P}) < \epsilon$.*

PROOF. $\mathcal{P} \subset \mathcal{D}_0(\hat{\mathcal{P}}, \rho)$ since $d_H(\mathcal{P}, \hat{\mathcal{P}}) < \rho$. Also $dia(\mathcal{D}_0(\hat{\mathcal{P}}, \rho)) < \epsilon$ since $dia(\hat{\mathcal{P}}) < \epsilon - \rho$. Hence $dia(\mathcal{P}) < \epsilon$. □

Now we address the issue of numerical computation error in determining a deterministic and transversal discrete transition. The conditions developed in the following lemmas are sufficient in that if they are satisfied by a given polyhedron $\hat{\mathcal{P}}$ with a given approximation error ρ at time t, then there is a deterministic and transversal discrete transition at some time $\tau \in [t-h, t]$.

LEMMA 8. *For given $\hat{\mathcal{D}}_t(\mathcal{B}_\delta(x_0))$, $\rho > 0$, and a location l_c, if $\hat{\mathcal{D}}_t(\mathcal{B}_\delta(x_0), \rho) \subset Inv_c^C$ and $\hat{\mathcal{D}}_{t-h}(\mathcal{B}_\delta(x_0), \rho) \subset Inv_c$ for some $\delta > 0$ and $h > 0$, then there is a discrete transition from the location l_c.*

PROOF. Since $d_H(\mathcal{D}_t(\mathcal{B}_\delta(x_0)), \hat{\mathcal{D}}_t(\mathcal{B}_\delta(x_0))) \leq \rho$, $\mathcal{D}_t(\mathcal{B}_\delta(x_0)) \subset \hat{\mathcal{D}}_t(\mathcal{B}_\delta(x_0), \rho)$. Similarly, $\mathcal{D}_{t-h}(\mathcal{B}_\delta(x_0)) \subset \hat{\mathcal{D}}_{t-h}(\mathcal{B}_\delta(x_0), \rho)$. Hence $\mathcal{D}_t(\mathcal{B}_\delta(x_0)) \subset Inv_c^C$ and $\mathcal{D}_{t-h}(\mathcal{B}_\delta(x_0)) \subset Inv_c$. Then the result follows immediately from Lemma 3. □

LEMMA 9. *For given $\hat{\mathcal{P}}_t$, l_c, and $\rho > 0$, suppose that $\mathcal{D}_0(\hat{\mathcal{P}}_t, \rho) \subset Inv_c^C$ and $\mathcal{D}_0(\hat{\mathcal{P}}_{t-h}, \rho) \subset Inv_c$ for some $h > 0$. Then there is a deterministic discrete transition from l_c to l_n if there exists a location l_n such that $l_n \neq l_c$ and $\mathcal{D}_0(\hat{\mathcal{P}}_t, \rho) \subset Inv_n$.*

PROOF. The assumption implies that there is a discrete transition from the given location l_c by Lemma 8. Since $\mathcal{P}_t \subset \mathcal{D}_0(\hat{\mathcal{P}}_t, \rho)$, and $\mathcal{D}_0(\hat{\mathcal{P}}_t, \rho) \subset Inv_n$, we have $\mathcal{P} \subset Inv_n$. Then by Lemma 4, the conclusion holds. □

LEMMA 10. *For a given $\hat{\mathcal{P}}_t$ and given $\rho > 0$, γ, and $h > 0$ satisfying the hypothesis given in Lemma 6, suppose that there is a deterministic discrete transition from a location l_c to a location l_n. Then for any $\epsilon > 0$, the discrete transition is transversal if the following conditions hold:*

(i) $h < (dia(\hat{\mathcal{J}}_{c,n})/2)/(2\bar{v})$,

(ii) $\mathcal{D}_0(\hat{\mathcal{J}}_{c,n}, dia(\hat{\mathcal{J}}_{c,n})/2) \subset (Inv_c \cup Inv_n)$,

(iii) $\langle \dot{x}_c, \vec{n} \rangle > \epsilon \wedge \langle \dot{x}_n, \vec{n} \rangle > \epsilon, \quad \forall x \in \mathcal{V}(\hat{\mathcal{J}}'_{c,n})$,

where $\hat{\mathcal{J}}_{c,n} := \mathcal{D}_0(\hat{\mathcal{P}}_t, \gamma+\rho) \cap Inv_c \cap Inv_n$, $\hat{\mathcal{J}}'_{c,n} := \mathcal{D}_0(\hat{\mathcal{J}}_{c,n}, dia(\hat{\mathcal{J}}_{c,n})/2) \cap Inv_c \cap Inv_n$, and $\dot{x}_i, \vec{n}$ are as defined in Lemma 5.

PROOF. Note that $\mathcal{D}_0(\mathcal{P}_t, \gamma) \subset \mathcal{D}_0(\hat{\mathcal{P}}_t, \gamma+\rho)$ since $d_H(\mathcal{P}_t, \hat{\mathcal{P}}_t) \leq \rho$. Then, by the definition of $\mathcal{J}_{c,n}$ given in Lemma 5 and $\hat{\mathcal{J}}_{c,n}$, we know $\mathcal{J}_{c,n} \subset \hat{\mathcal{J}}_{c,n}$. Hence, $\hat{\mathcal{J}}_{c,n} \neq \emptyset$ and furthermore, it is an over-approximation of the deterministic discrete transition state as shown in the proof of Lemma 4.

By the same argument used in the proof of Lemma 5, if (i) holds, then $x(\tau; z) \in \mathcal{D}_0(\hat{\mathcal{J}}_{c,n}, dia(\hat{\mathcal{J}}_{c,n})/2)$ for $\tau \in [0, h]$ and for any $z \in \hat{\mathcal{J}}_{c,n}$. Hence, (ii) and (iii) imply that $\mathcal{D}_\tau(\hat{\mathcal{J}}_{c,n}) \subset Inv_n$ for $\tau \in [0, h]$. Therefore, the conclusion holds since $\mathcal{J}_{c,n} \subset \hat{\mathcal{J}}_{c,n}$. □

4. ARCHITECTURE

In [6], we have shown that an approximate bounded reach set of a DTLHA from an initial condition x_0 in an initial location l_0 can be computed with arbitrarily small approximation error ϵ under the assumption that every discrete transition is deterministic and transversal. We also have proposed an algorithm for such bounded ϵ-reach set computation. Algorithm 1 shows the main computation steps of the proposed algorithm where $\mathcal{B}_\delta(x_0)$ is an δ-neighborhood of a given initial state x_0, $\mathcal{D}_t(\mathcal{B}_\delta(x_0), \gamma)$ is a γ-approximation of $\mathcal{D}_t\mathcal{B}_\delta(x_0)$, and h is a sampling period corresponding to the value of γ satisfying an over-approximation condition derived in [6]. Note that $\mathcal{R}$ returned from the algorithm is in fact a bounded ϵ-reach set as stated in the following theorem.

THEOREM 1. *Given input $(\mathcal{A}, N, T, l_0, x_0, \epsilon)$, Algorithm 1 terminates in a finite number of iterations and returns $\mathcal{R}$, a bounded ϵ-reach set of $\mathcal{A}$ from $x_0 \in Inv_0$ up to time $t_f := \min\{\tau_N, T\}$, if $x(\tau_k)$ satisfies a deterministic and transversal discrete transition condition for every $\tau_k \leq t_f$.*

$$\mathcal{R} := \bigcup_{m=0}^{M} \mathcal{D}_{mh}(\mathcal{B}_\delta(x_0), \gamma) \tag{9}$$

where δ, γ, h are the values when Algorithm 1 returns and M is a value such that $Mh \in [t_f, t_f + h]$.

Algorithm 1: An algorithm proposed in [6] for a bounded ϵ-reach set of a DTLHA $\mathcal{A}$ from an initial state $x_0 \in Inv_0$.

Input: $\mathcal{A}, N, T, l_0, x_0, \epsilon$

Initialize δ and γ with arbitrary positive real values.
• Initialize $t = 0$, $jump = 0$, and $\mathcal{R} = \emptyset$.
while true **do**
  Compute $\mathcal{B}_\delta(x_0)$ and h from γ.
  Compute $\mathcal{D}_t(\mathcal{B}_\delta(x_0))$ and $\mathcal{D}_t(\mathcal{B}_\delta(x_0), \gamma)$.
  if $dia(\mathcal{D}_h(\mathcal{B}_\delta(x_0), \gamma)) > \epsilon$ **then** Reduce δ, γ and goto •.
  if *discrete transition* **then**
    if *deterministic* $\wedge$ *transversal* **then**
      $t \leftarrow t + h$, $jump \leftarrow jump + h$,
      $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{D}_h(\mathcal{B}_\delta(x_0), \gamma)$
    else Reduce δ, γ and goto •
  else $t \leftarrow t + h$ and $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{D}_h(\mathcal{B}_\delta(x_0), \gamma)$.
  if $(t > T) \vee (jump > N)$ **then return** $\mathcal{R}$
end

In the above algorithm and accompanying theoretical results for bounded ϵ-reach set computation in [6], we have not addressed the issue of computation with finite precision. Moreover, even though the proposed algorithm can compute a

bounded ϵ-reach set correctly, it is far from efficient in terms of computational efficiency since the algorithm restarts its ϵ-reach set computation from an initial state whenever the values of δ and γ are changed. Moreover, the algorithm does not provide any flexibility in choosing the values for δ and γ whenever the algorithm needs to be continued with different δ and γ values, since one specific decision rule resulting in δ and γ which are monotonically decreasing is tightly embedded within the algorithm. We address all these issues in this section.

It is helpful to take a top-down approach since we want to address modularity, flexibility, and architecture. Hence we begin by presenting an architecture for bounded ϵ-reach set computation followed by a new bounded ϵ-reach set algorithm which is based on the theoretical framework developed in Section 3 so that the overall computation process can be better optimized in terms of the computational efficiency and flexibility.

We first discuss the overall structure of computation of a bounded ϵ-reach set of a DTLHA from an initial state. One of the main objectives of the design, which also includes a new algorithm, is to provide flexibility. We argue that this can be achieved by decoupling the part where decisions are made, called *Policy*, and the part where some specific steps of computation are performed, which is called *Algorithm* in our context but called *Mechanism* in some other contexts. Fig. 2 shows a proposed architecture based on this design principle.

In general, a bounded ϵ-reach set computation of a DTLHA can be described by a modeling or description language. The System Description module contains information about the system, described by a modeling language; it consists of $\mathcal{X}$ the domain of continuous state space, the DTLHA $\mathcal{A}$, and an initial continuous state x_0, an initial location l_0 (i.e., discrete state) where x_0 is contained. Also, to specify the required computation, we need an upper bound T on terminsal, an upper bound N on the total number of discrete transitions, and an approximation parameter ϵ. The data generated by the System Description module, called `SystemData`, is stored in the *Data* module which can then be used by the rest of the modules in the architecture. With the inputs from the *Policy* module, the *Main Algorithm* continues ϵ-reach set computation utilizing Sub-functions and functions from the *Condition Checking* module until it either successfully finishes its computation or cannot make further progress which happens when some required conditions are not met. If the algorithm encounters the latter situation, then it returns to the *Policy* module indicating the problems that the Policy module has to resolve so as to continue the computation. In making a decision, the Policy module may need to access functions in the *Condition Checking* module. During the bounded ϵ-reach set computation, the *Main Algorithm* stores its computational state in two data structures, called `ReachSetHistory` and `DiscreteTransitio- nHistory`. `ReachSetHistory` contains the computation results and the information used to produce the results at each step of computation, described in more detail in Section 5.1. One of the most import benefits of maintaining this information is that the computational efficiency can be reduced significantly since the computation

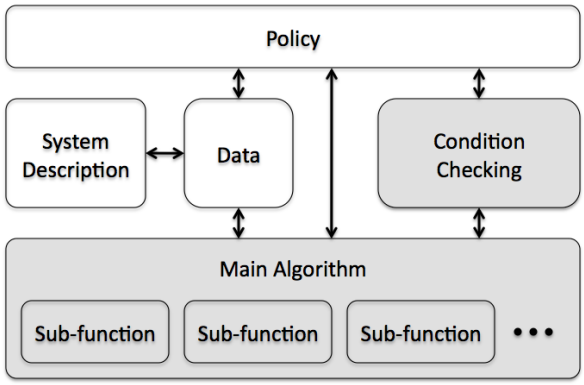

Figure 2: Architecture for bounded ϵ-reach set computation

does not need to be restarted from the initial state whenever new parameter values, such as a smaller δ and γ, have to be used to continue the computation. Under this architecture, the *Policy* module can go back to any past computational step and make the *Main Algorithm* restart the computation from that point. Maintaining this information in `DiscreteTransitionHistory`, also contributes to improving efficiency of the overall bounded ϵ-reach set computation process.

5. ALGORITHM

In this section, we first propose an algorithm for bounded ϵ-reach set computation of a DTLHA with the infinite precision computation assumption. Then, we extend this algorithm for the finite precision case.

5.1 Infinite Precision Calculations

In computing a bounded ϵ-reach set, the following set of questions need to be answered at each step of computation in the Main Algorithm to produce a correct result.

- Is the diameter of a reach set at current time step less than the given ϵ?
- Is the given over-approximation parameter γ large enough so as to contain all the state evolution during the given time interval h?
- Is there a discrete transition?
- If there is a discrete transition, is the discrete transition a deterministic discrete transition?
- If there is a deterministic discrete transition, is the discrete transition a transversal discrete transition?

If any of these questions except the third one is answered 'no', then the Main Algorithm cannot make any further progress and the Policy module will need to be invoked to resolve the problem by determining the values for the parameter δ, γ, and h. It also needs to specify from which past computation step the computation should resume.

The proposed algorithm for bounded ϵ-reach set computation is decomposed into roughly two parts, the *Main Algorithm* module and the *Condition Checking* module.

The Condition Checking module consists of the following set of functions which are based on the results in Section 3.1.

`IsEpsilonSmall`$(\mathcal{P}, \epsilon)$. This function returns `true` if $dia(\mathcal{P}) < \epsilon$, where $dia(\mathcal{P})$ denotes the diameter of a given polyhedron $\mathcal{P}$.

`IsOverApproximate`(γ, h). This function uses the result in (5) to check the over-approximation condition, i.e., `true` if $h < \gamma/\bar{v}$.

`IsTransition`$(\mathcal{P}, l_c)$. This function returns `true` if $\mathcal{P} \subset Inv_c^C$ and `false` if $\mathcal{P} \subset Inv_c$. If a given $\mathcal{P}$ has nonempty intersection with both Inv_c and Inv_c^C, then other values of δ or h need to be used to resolve the ambiguity.

`IsDeterministic`$(\mathcal{P}, l_c)$. For given input, this function returns the location l_n if there is a location $l_n \in \mathbb{L}$ which satisfies the hypothesis given in Lemma 4. Otherwise it returns `error`.

`IsTransversal`$(\mathcal{P}, \gamma, h, l_c, l_n)$. For given input, this function returns a set $\mathcal{D}_h(\mathcal{J}_{c,n}, \gamma) \setminus Inv_c$ if the conditions (i), (ii), and (iii) given in Lemma 5 are satisfied. Otherwise, it returns `error`. Note that $\mathcal{D}_h(\mathcal{J}_{c,n}, \gamma) \setminus Inv_c$ is an over-approximation of the exact discrete transition state and the bounded ϵ-reach set computation continues its computation from this set at a new location l_n.

These functions are used in the Main Algorithm as described in below. Roughly, the Main Algorithm consists of two part. The first part is a function called `ReachSet`$(\cdot)$ which is the main function to compute a bounded ϵ-reach set and the second part is a set of function called *Sub-functions* which are called by `ReachSet`$(\cdot)$ during its computation. We first describe the functions defined as a Sub-function.

`ReachNext`$(\mathcal{P}, \gamma, h)$. For given input, this function returns $\mathcal{D}_h(\mathcal{P})$, the linear image of a polyhedron $\mathcal{P}$ under a linear dynamics, and $\mathcal{D}_h(\mathcal{P}, \gamma)$, an over-approximation of $\mathcal{D}_h(\mathcal{P})$ for a given over-approximation parameter γ.

To compute these polyhedra, it exploits the fact that the polyhedral structure is preserved under a linear dynamics in the following way. For a given polyhedron $\mathcal{P}$, it first computes $\mathcal{V}(\mathcal{P})$ which is set that contains the vertices of $\mathcal{P}$, and possibly some other points in $\mathcal{P}$. (The reason for allowing some other points that are possibly not vertices is because $\mathcal{P}$ is itself computed as the linear image of a finite number of points, and we would like to avoid the need to computationally determine precisely which remain extreme points under the linear map). Then for each $v_i \in \mathcal{V}(\mathcal{P})$, it computes $v_i(h) := e^{Ah} + \int_0^h e^{As} u ds$ where A and u is given by a linear dynamics of a location on which the linear image of $\mathcal{P}$ is computed. If we let $\mathcal{V}_h(\mathcal{P}) := \{v_i(h) : v_i \in \mathcal{V}(\mathcal{P})\}$, then the convex hull of $\mathcal{V}_h(\mathcal{P})$ defines a $\mathcal{D}_h(\mathcal{P})$. From $\mathcal{V}_h(\mathcal{P})$, this function can also compute $\mathcal{D}_h(\mathcal{P}, \gamma)$ easily. For each $v_i(h) \in \mathcal{V}_h(\mathcal{P})$, it first constructs a hypercubic γ-neighborhood of $v_i(h)$. If we denote such a neighborhood by $\mathcal{B}_\gamma(v_i(h))$, then the convex hull of the set of vertices of $\mathcal{B}_\gamma(v_i(h))$ for all $v_i(h) \in \mathcal{V}_h(\mathcal{P})$ defines a $\mathcal{D}_h(\mathcal{P}, \gamma)$. In [6], it has been shown that $\mathcal{D}_h(\mathcal{P}, \gamma)$ is in fact the exact γ-neighborhood of $\mathcal{D}_h(\mathcal{P})$.

`AtTransition`$(\mathcal{P}, \mathcal{D}_0(\mathcal{P}, \gamma), \gamma, h, l_c)$. If there is a deterministic and transversal discrete transition, this function returns a location l_n to which a discrete transition occurs from the given location l_c, and also returns a polyhedral approximation of the exact discrete transition state $\mathcal{P}'$ from which the bounded ϵ-reach set can continue at a location l_n. This function calls `IsDeterministic`$(\mathcal{P}, l_c)$ first to determine a location l_n and then calls `IsTransversal`$(\mathcal{D}_0(\mathcal{P}, \gamma), \gamma, h, l_c, l_n)$ to determine $\mathcal{P}'$. If any of these functions returns `error`, this function returns the same `error`.

`ImageAt`(t, δ). Even though the overall computational efficiency of a bounded ϵ-reach set computation can be improved by the proposed architecture, it is unavoidable to restart the computation from an initial state when the value of parameter δ which defines an initial neighborhood around an initial state is changed. If the algorithm encounters such a situation, `ImageAt`$(\cdot)$ can be used to reduce the number of computational steps. The basic idea of the algorithm involved is the following.

For given input t and δ, the goal is to compute $\mathcal{D}_t(\mathcal{B}_\delta(x_0))$ which is a polyhedron reached at time t from the δ-neighborhood around an initial state x_0. Note that if we know the time interval between discrete transitions, then we can directly compute of the linear image of a polyhedron from another polyhedron as it is computed in `ReachNext`$(\cdot)$. As an example, in Fig. 3, we can easily compute $\mathcal{D}_{\tau_1}(\mathcal{B}_\delta(x_0))$ from $\mathcal{B}_\delta(x_0)$ and $\mathcal{D}_t(\mathcal{B}_\delta(x_0))$ from $\mathcal{D}'_{\tau_1}(\mathcal{B}_\delta(x_0))$ in this way. Now, to compute $\mathcal{D}'_{\tau_1}(\mathcal{B}_\delta(x_0))$ which is an over-approximation of $\mathcal{D}_{\tau_1}(\mathcal{B}_\delta(x_0))$ at time τ_1, this function call `AtTransition`$(\cdot)$ with $(\mathcal{D}_{\tau_1}(\mathcal{B}_\delta(x_0)), \mathcal{D}_{\tau_1}(\mathcal{B}_\delta(x_0), \gamma), \gamma, h, l_c)$ as input to compute $\mathcal{D}'_{\tau_1}(\mathcal{B}_\delta(x_0))$. If `AtTransition`$(\cdot)$ returns `error`, then `ImageAt`$(\cdot)$ returns the same `error`. Otherwise, this function continues its image computation from the returned polyhedron. Note that γ, h, and l_c are stored in `DiscreteTransitionHistory` data structure at each time of discrete transition by `ReachSet`$(\cdot)$.

Now, we describe the main function, called `ReachSet`$(\cdot)$, in the Main Algorithm.

`ReachSet`$(k, \delta_k, \gamma_k, h_k)$. : This function computes a bounded ϵ-reach set by utilizing all other functions in Main Algorithm and Condition Checking modules, given an input $(k, \delta_k, \gamma_k, h_k)$ from the Policy module. For the given input, this function first retrieves computation data at the $(k-1)$-th computation step from the `ReachSetHistory` and starts its k-th computation step using the this data. As shown in Algo-

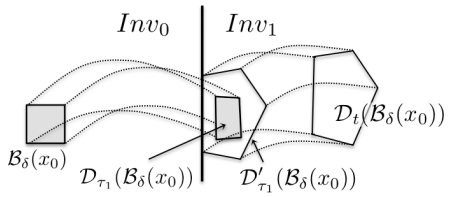

Figure 3: Illustration of the computation in `ImageAt`$(\cdot)$

rithm 2, it continues its computation until it either successfully computes a bounded ϵ-reach set or encounters some `error`. If there is an `error` from any of the functions that are called, then this function returns the same `error` to the Policy module to indicate the cause of the `error`. For each types of `error`, `ReachSet`$(\cdot)$ expects to have a new input from the Policy module to continue its computation.

`ReachSet`$(\cdot)$ stores its computation results in addition to the information used for its computation in `ReachSetHistory` data structure,

$$\{k, t_k, l_k, \delta_k, \gamma_k, h_k, \mathcal{D}_{t_k}(\mathcal{B}_{\delta_k}(x_0)), \mathcal{D}_{t_k}(\mathcal{B}_{\delta_k}(x_0), \gamma_k)\}. \quad (10)$$

5.2 Finite Precision Calculations

As shown in Fig. 4, an additional module, called *Numerical Computation*, is added to the architecture of the proposed algorithm. It can be thought as a collection of numerical computation functions based on some computation algorithms. As an example, $a(e^{At}, \sigma_e)$ can be computed in many different ways as shown in [7]. Each of these can computes e^{At} with different accuracy. Hence, the computational accuracy of a bounded ϵ-reach set computation inevitably depends on the choice of the algorithms for computation of each of the $a(x, y)$'s assumed in above. We decouple such issues arising in the low level numerical computations from our proposed computation algorithm , which is the reason for the separate module for numerical computation in our architecture.

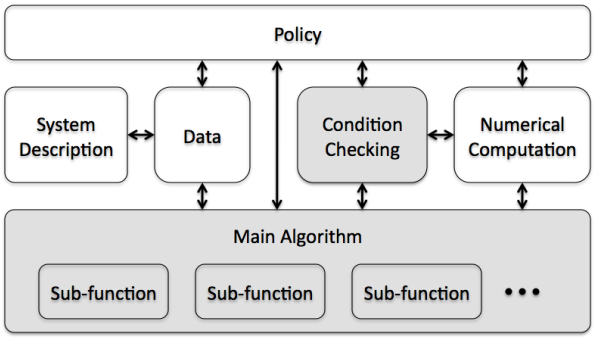

Figure 4: Architecture for bounded ϵ-reach set computation with numerical computation module

In the sequel, we denote by $\hat{x}$ the approximately computed value of x using a function in the Numerical Computation

Algorithm 2: An algorithm of `ReachSet`$(\cdot)$

Input: $k, \delta_k, \gamma_k, h_k$
Result: ReachSetHistory, TransitionHistory

while true **do**
  if `IsOverApproximate` $(\gamma_k, h_k) =$ false **then**
    return error
  end
  `Get` (ReachSetHistory, $k-1$) $\rightarrow$ data$(k-1)$
  if $\delta_k \neq \delta_{k-1}$ **then**
    `ImageAt`$(t_{k-1}, \delta_k) \rightarrow \mathcal{P}'$
    if $\mathcal{P}' \neq$ error **then** $\mathcal{D}_{t_{k-1}}(\mathcal{B}_{\delta_k}(x_0)) \leftarrow \mathcal{P}'$
    else return error
  end
  `ReachNext` $(\mathcal{D}_{t_{k-1}}(\mathcal{B}_{\delta_k}(x_0)), \gamma_k, h_k) \rightarrow$
  $\mathcal{D}_{t'}(\mathcal{B}_{\delta_k}(x_0)), \mathcal{D}_{t'}(\mathcal{B}_{\delta_k}(x_0), \gamma_k)$
    where $t' = t_{k-1} + h_k$
  if `IsEpsilonSmall` $(\mathcal{D}_{t'}(\mathcal{B}_{\delta_k}(x_0), \gamma_k), \epsilon) =$ false **then**
    return error
  end
  `IsTransition`$(\mathcal{D}_{t'}(\mathcal{B}_{\delta_k}(x_0)), l_{k-1}) \rightarrow$ out
  if out $=$ error **then return** error
  else if out $=$ false **then** $l_k \leftarrow l_{k-1}$
  else if out $=$ true **then**
    `AtTransition`
    $(\mathcal{D}_{t'}(\mathcal{B}_{\delta_k}(x_0)), \mathcal{D}_{t'}(\mathcal{B}_{\delta_k}(x_0), \gamma_k), \gamma_k, h_k, l_{k-1})$
    $\rightarrow (l_k, \mathcal{P})$
    if $l_k =$ null $\vee\ \mathcal{P} =$ null **then**
      return error
    end
    jump $\leftarrow$ jump $+ 1$
    $\mathcal{D}_{t'}(\mathcal{B}_{\delta_k}(x_0)) \leftarrow \mathcal{P}$
    $\mathcal{D}_{t'}(\mathcal{B}_{\delta_k}(x_0), \gamma_k) \leftarrow \mathcal{D}_0(\mathcal{P}, \gamma_k)$
    `Add` $\{t', l_k, h_k\}$ to TransitionHistory.`tail`
  end
  $t_k \leftarrow t_{k-1} + h_k$
  data$(k) =$
  $\{k, t_k, l_k, \delta_k, \gamma_k, h_k, \mathcal{D}_{t_k}(\mathcal{B}_{\delta_k}(x_0)), \mathcal{D}_{t_k}(\mathcal{B}_{\delta_k}(x_0), \gamma_k)\}$
  `Add` data (k) to ReachSetHistory.`tail`
  $k \leftarrow k + 1$
  if $(t_k > T) \vee$ (jump $> N$) **then return** done
end

module.

All of the conditions to be checked in the Condition Checking module in Section 5.1 are based on the assumption that there are no computation errors in computing a polyhedron $\mathcal{D}_t(\mathcal{B}_\delta(x_0))$ reachable from an initial polyhedron $\mathcal{B}_\delta(x_0)$ at time t. However, we cannot use the same conditions in the current context where some computation errors are involved.

The functions in the Condition Checking module in Section 5.1 can be redefined based on the theories in Section 3.2 as follows. In the sequel, a polyhedron $\hat{\mathcal{P}}$ should be understood as an approximate of a polyhedron $\mathcal{P}$ with some approximation parameter $\rho \in \mathbb{R}^+$, i.e., $\hat{\mathcal{P}}$ is an $a(\mathcal{P}, \rho)$.

`IsEpsilonSmall`$(\hat{\mathcal{P}}, \rho, \epsilon)$. This function determines whether $dia(\mathcal{P}) < \epsilon$ or not by checking the condition $dia(\hat{\mathcal{P}}) < \epsilon - \rho$.

`IsOverApproximate`(γ, h, ρ). This function checks whether the given input values satisfy $h < (\gamma - \rho)/\bar{v}$ where $\bar{v}$ is as defined in (5).

`IsTransition`$(\hat{\mathcal{P}}, \rho, l_c)$. This function returns `true` if $\mathcal{D}_0(\hat{\mathcal{P}}, \rho) \subset Inv_c^C$ and `false` if $\mathcal{D}_0(\hat{\mathcal{P}}, \rho) \subset Inv_c$. In other case such as $(\mathcal{D}_0(\hat{\mathcal{P}}, \rho) \cap Inv_c \neq \emptyset) \wedge (\mathcal{D}_0(\hat{\mathcal{P}}, \rho) \cap Inv_c^C \neq \emptyset)$, this function returns `error`.

`IsDeterministic`$(\hat{\mathcal{P}}, \rho, l_c)$. For given input, this function determines whether a discrete transition is a deterministic transition or not, using the conditions (i) and (ii) in Lemma 9. If it is a deterministic discrete transition, it returns a new location l_n to which a deterministic discrete transition occurs.

`IsTransversal`$(\hat{\mathcal{P}}, \rho, \gamma, h, l_c, l_n)$. For given input, this function determines whether a discrete transition from a location l_c to other location l_n is a transversal discrete transition or not using the conditions (i), (ii), (iii), and (iv) in Lemma 10. If it is a transversal discrete transition, it returns $\mathcal{D}_0(\hat{\mathcal{D}}_h(\hat{\mathcal{J}}_{c,n}), \mu_x) \setminus Inv_c$ where $\hat{\mathcal{J}}_{c,n}$ is as defined in Lemma 10. Note that $d_H(\mathcal{D}_h(\hat{\mathcal{J}}_{c,n}), \hat{\mathcal{D}}_h(\hat{\mathcal{J}}_{c,n})) \leq \mu_x$.

To incorporate the computation error in the computation of a bounded ϵ-reach set, each functions in the Main Algorithm module is modified as follows. Note that the basic algorithm in each of these functions is the same as in Section 5.1.

`ReachNext`$(\hat{\mathcal{P}}, \rho, \gamma, h)$. Here, an input ρ is an approximate error bound such that $d_H(\mathcal{P}, \hat{\mathcal{P}}) \leq \rho$. For the given input, the function returns $\hat{\mathcal{D}}_h(\hat{\mathcal{P}})$ and $\hat{\mathcal{D}}_h(\hat{\mathcal{P}}, \gamma)$ where $\hat{\mathcal{D}}(\cdot)$ ia approximation of $\mathcal{D}(\cdot)$. Hence, it is necessary to determine an approximation error bound ρ' such that $d_H(\mathcal{D}_h(\mathcal{P}), \hat{\mathcal{D}}_h(\hat{\mathcal{P}}) < \rho'$. Since there is only one step of image computation in this computation, the additional computation error introduced is just μ_x. Therfore, $\rho' = \rho + \mu_x$.

`AtTransition`$(\hat{\mathcal{P}}, \mathcal{D}_0(\hat{\mathcal{P}}, \gamma), \rho, \gamma, h, l_c)$. Note that the input ρ is an approximation error bound for both $\hat{\mathcal{P}}$ and $\mathcal{D}_0(\hat{\mathcal{P}}, \gamma)$. If these polyhedra satisfy the conditions given in Lemma 9 and 10, then it returns a new location l_n to which a discrete transition occurs, and a polyhedron $\mathcal{P}'$ from which the ϵ-reach set computation can continue. An additional output ρ' as an approximation error bound corresponding to $\mathcal{P}'$ is also returned along with $\mathcal{P}'$.

At each time of discrete transition, the algorithm introduces an additional computation error ρ'' in computing $\mathcal{P}'$. The specific value of ρ'' can be determined by considering the specific steps of computation of $\mathcal{P}'$ in `IsTransversal`$(\cdot)$ function. In fact, $\rho'' = \mu_x + 3\mu_c$ since there are one image computation and three intersection computations in computing $\mathcal{P}'$. $\rho' = \rho + (\mu_x + 3\mu_c)$.

`ImageAt`(t, δ). This function starts its computation from an exact initial polyhedron $\mathcal{B}_\delta(x_0)$ with the given δ and returns $\hat{\mathcal{D}}_t(\mathcal{B}_\delta(x_0))$ which is an approximate of $\mathcal{D}_t(\mathcal{B}_\delta(x_0))$ at time t. If we follow the algorithm of this function shown in Section 5.1, then we can see that it introduces a computation error $2\mu_x + 3\mu_c$ at each time of discrete transition. One μ_x is caused by the image computation within a single location and the rest $\mu_x + 3\mu_c$ is caused by the computation steps in `AtTransition`$(\cdot)$ function. Hence, if the number of discrete transitions up to time t is m, then $\hat{\mathcal{D}}_t(\mathcal{B}_\delta(x_0))$ may have computation error up to $\rho := \mu_x + m(2\mu_x + 3\mu_c)$. Note that an additional μ_x comes from the image computation at the last location. This ρ is also an output of this function.

An additional function is added in the Main Algorithm as one of the sub-functions.

`ErrorLTIImage`$(\sigma_e, \sigma_i, \sigma_p, \sigma_a)$. This function computes μ_x as defined in (7) for a given approximation error bound for each of the approximate computation functions defined in the *Numerical Computation* module.

`ReachSet`$(k, \delta_k, \gamma_k, h_k, \sigma_e, \sigma_i, \sigma_p, \sigma_a, \mu_c)$. Besides the input $(k, \delta_k, \gamma_k, h_k)$, an additional set of inputs $(\sigma_e, \sigma_i, \sigma_p, \sigma_a, \mu_c)$ is provided by the *Policy* module. Now, when a certain types of error occurs during the execution of the algorithm, the Policy module needs to make a decision about the choice of the numerical computation algorithm and its corresponding computation accuracy in the *Numerical Computation* module, in addition to deciding the values for $(k, \delta_k, \gamma_k, h_k)$. Table 1 shows an example of such a Policy. As an example, the Policy module can reduce either the values of γ and δ or use a different numerical computation function with better computation accuracy when an `error` is raised in `IsEpsilonSmall`$(\cdot)$ function.

The overall structure and steps of the algorithm are almost the same as the algorithm shown in Section 5.1 except that: (i) At the beginning of the loop, μ_x is determined by `ErrorLTIImage`$(\cdot)$, and (ii) An upper bound of computation error ρ_k is determined and it is stored at each step of iteration. The complete algorithm of `ReachSet`$(\cdot)$ is shown in Algorithm 3.

Table 1: A proposed policy

Error	Policy
`IsEpsilonSmall`$(\cdot)$	$(\gamma \downarrow, \delta \downarrow) \vee (\sigma_e, \sigma_i, \sigma_p, \sigma_a, \mu_c) \downarrow$
`IsOverApproximate`$(\cdot)$	$(\gamma \uparrow) \vee (h \downarrow) \vee (\sigma_e, \sigma_i, \sigma_p, \sigma_a, \mu_c) \downarrow$
`IsDeterministic`$(\cdot)$	$(\delta \downarrow) \vee (\sigma_e, \sigma_i, \sigma_p, \sigma_a, \mu_c) \downarrow$
`IsTransversal`$(\cdot)$	$(\gamma \downarrow, \delta \downarrow) \vee (\sigma_e, \sigma_i, \sigma_p, \sigma_a, \mu_c) \downarrow$

We now have the overall main result:

THEOREM 2. *For a given* `SystemData` $:= (\mathcal{X}, \mathcal{A}, l_0, x_0, T, N, \epsilon)$, *if* `ReachSet`$(\cdot)$ *in Algorithm 3 returns* `done`, *then a bounded ϵ-reach set of a DTLHA $\mathcal{A}$ over the continuous do-*

Algorithm 3: An algorithm of `ReachSet`$(\cdot)$ under numerical computation error.

Input: $k, \delta_k, \gamma_k, h_k, \sigma_e, \sigma_i, \sigma_p, \sigma_a, \mu_c$
Result: ReachSetHistory, TransitionHistory

while true **do**
  `Get` (ReachSetHistory, $k-1$) $\rightarrow$ data$(k-1)$
  `ErrorLTIImage`$(\sigma_e, \sigma_i, \sigma_p, \sigma_a) \rightarrow \mu_x$
  if $\delta_k \neq \delta_{k-1}$ **then**
    `ImageAt`$(t_{k-1}, \delta_k) \rightarrow \{\rho', \mathcal{P}'\}$
    if $\mathcal{P}' \neq$ error **then** $\hat{\mathcal{D}}_{t_{k-1}}(\mathcal{B}_{\delta_k}(x_0)) \leftarrow \mathcal{P}'$
    else return error
  end
  $\rho' \leftarrow \max\{\rho_{k-1}, \rho'\}$
  if `IsOverApproximate` $(\gamma_k, h_k, \rho' + \mu_x) =$ false **then return** error
  `ReachNext` $(\hat{\mathcal{D}}_{t_{k-1}}(\mathcal{B}_{\delta_k}(x_0)), \rho', \gamma_k, h_k)$
  $\rightarrow \{\hat{\mathcal{D}}_{t'}(\mathcal{B}_{\delta_k}(x_0)), \hat{\mathcal{D}}_{t'}(\mathcal{B}_{\delta_k}(x_0), \gamma_k), \rho_k\}$
    where $t' = t_{k-1} + h_k$ and $\rho_k = \rho' + \mu_x$
  if `IsEpsilonSmall` $(\hat{\mathcal{D}}_{t'}(\mathcal{B}_{\delta_k}(x_0), \gamma_k), \rho_k, \epsilon) =$ false **then return** error
  `IsTransition`$(\hat{\mathcal{D}}_{t'}(\mathcal{B}_{\delta_k}(x_0)), \rho_k, l_{k-1}) \rightarrow$ out
  if out $=$ error **then return** error
  else if out $=$ false **then** $l_k \leftarrow l_{k-1}$
  else if out $=$ true **then**
    `AtTransition`
    $(\hat{\mathcal{D}}_{t'}(\mathcal{B}_{\delta_k}(x_0)), \hat{\mathcal{D}}_{t'}(\mathcal{B}_{\delta_k}(x_0), \gamma_k), \rho_k, \gamma_k, h_k, l_{k-1})$
    $\rightarrow (l_k, (\rho'', \mathcal{P}))$
    if $l_k =$ null $\vee\, \mathcal{P} =$ null **then return** error
    $\rho_k \leftarrow \rho''$
    jump $\leftarrow$ jump $+ 1$
    $\hat{\mathcal{D}}_{t'}(\mathcal{B}_{\delta_k}(x_0)) \leftarrow \mathcal{P}$
    $\hat{\mathcal{D}}_{t'}(\mathcal{B}_{\delta_k}(x_0), \gamma_k) \leftarrow \hat{\mathcal{D}}_0(\mathcal{P}, \gamma_k)$
    `Add` $\{t', l_k, h_k\}$ to TransitionHistory.`tail`
  end
  $t_k \leftarrow t_{k-1} + h_k$
  data$(k) =$
  $\{k, t_k, l_k, \delta_k, \gamma_k, h_k, \rho_k, \hat{\mathcal{D}}_{t_k}(\mathcal{B}_{\delta_k}(x_0)), \hat{\mathcal{D}}_{t_k}(\mathcal{B}_{\delta_k}(x_0), \gamma_k)\}$
  `Add` data (k) to ReachSetHistory.`tail`
  $k \leftarrow k + 1$
  if $(t_k > T) \vee$ (jump $> N$) **then return** done
end

main $\mathcal{X}$ from an initial state $x_0 \in Inv_0$, denoted as $\mathcal{R}_{t_f}(x_0, \epsilon)$, is the following.

$$\mathcal{R}_{t_f}(x_0, \epsilon) := \bigcup_{k=1}^{K} \hat{\mathcal{D}}_{t_k}(\mathcal{B}_{\delta_k}(x_0), \gamma_k), \tag{11}$$

where K is the number of `data` *element in* `ReachSetHistory`*, $t_f := \min\{T, \tau_N\}$, and τ_N is the N-th discrete transition.*

Proof. Let data(k) be the k-th computation information stored in `ReachSetHistory` as shown in Algorithm 3.

Then, for each $k \leq K$, (i) (γ_k, h_k, ρ_x) satisfies Lemma 6, (ii) $(\hat{\mathcal{D}}_{t_k}(\mathcal{B}_{\delta_k}(x_0), \gamma_k), \rho_k, \epsilon)$ satisfies Lemma 7. These imply that $\hat{\mathcal{D}}_{t_k}(\mathcal{B}_{\delta_k}(x_0), \gamma_k)$ is guaranteed to be a correct γ_k-approximation of $\mathcal{D}_{t_k}(\mathcal{B}_{\delta_k}(x_0))$ by γ_k and h_k, i.e.,

$$\bigcup_{\tau \in [0, h_k]} \mathcal{D}_{t_k+\tau}(\mathcal{B}_{\delta_k}(x_0)) \subset \hat{\mathcal{D}}_{t_k}(\mathcal{B}_{\delta_k}(x_0), \gamma_k). \tag{12}$$

and moreover $dia(\hat{\mathcal{D}}_{t_k}(\mathcal{B}_{\delta_k}(x_0)\ , \gamma_k)) < \epsilon$.

If there is a discrete transition at $k \in K$, then (iii) $(\hat{\mathcal{D}}_{t_k}(\mathcal{B}_{\delta_k}(x_0)), \rho_k, l_k)$ satisfies Lemmas 8 and 9, (iv) $(\hat{\mathcal{D}}_{t_k}(\mathcal{B}_{\delta_k}(x_0), \gamma_k), \rho_k, \gamma_k, h_k, l_k, l'_k)$ satisfies Lemma 10 where l'_k is a value returned by `IsDeterministic`$(\cdot)$ in (iii). These impliy that a deterministic and transversal discrete transition event is correctly determined by $\hat{\mathcal{D}}_{t_k}(\mathcal{B}_{\delta_k}(x_0))$, $\hat{\mathcal{D}}_{t_k}(\mathcal{B}_{\delta_k}(x_0), \gamma_k)$, and ρ_k.

Since `ReachSet`$(\cdot)$ returns `done`, either $t_k > T$ or `jump` $> N$. This means that t_f is $\min\{T, \tau_N\}$.

Therefore, we conclude that $\mathcal{R}_{t_f}$ is a bounded ϵ-reach set of $\mathcal{A}$ from x_0. □

6. CONCLUSIONS

We have proposed an algorithm for a bounded ϵ-reach set of a Deterministic Transversal Linear Hybrid Automaton. We have also proposed an architecture for such computation. The proposed architecture can reduce the overall computational cost of a bounded ϵ-reach set computation by increasing the level of flexibility in the parameter adaption process.

We have also addressed the issue of numerical computation errors during a bounded ϵ-reach set computation. We have proposed an extended architecture that decouples numerical computation issues from the bounded ϵ-reach set algorithm itself.

7. REFERENCES

[1] R. Alur, T. Dang, and F. Ivančić. Counterexample-guided predicate abstraction of hybrid systems. *Theor. Comput. Sci.*, 354(2):250–271, 2006.

[2] A. Chutinan and B. H. Krogh. Computational techniques for hybrid system verification. *ITAC*, 48(1):64–75, 2003.

[3] G. Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. *International Journal on Software Tools for Technology Transfer*, 10(3):263–279, 2008.

[4] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2):110–122, 1997.

[5] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *ACM Symposium on Theory of Computing*, pages 373–382, 1995.

[6] K.-D. Kim, S. Mitra, and P. R. Kumar. Bounded ϵ-reachability of linear hybrid automata with a deterministic and transversal discrete transition condition. In *IEEE Conference on Decision and Control*, December 2010.

[7] C. Moler and C. V. Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five yeasr later. *SIAM Review*, 20(4):801–836, 1978.

[8] P. Tabuada, G. J. Pappas, and P. U. Lima. Composing abstractions of hybrid systems. In *HSCC 2002*, pages 436–450, 2002.

[9] V. Vladimerou, P. Prabhakar, M. Viswanathan, and G. E. Dullerud. Stormed hybrid systems. In *ICALP (2)*, volume 5126 of *LNCS*, pages 136–147. Springer, 2008.