ABSTRACT

In a previous paper [7] we have identified a special class of linear hybrid automata, called Deterministic Transversal Linear Hybrid Automata, and shown that an $\epsilon$-reach set up to a finite time, called a bounded $\epsilon$-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 this computation. In this paper we address the problem of determining a bounded $\epsilon$-reach set using variable finite precision numerical approximations. We present an algorithm for computing it that uses only such numerical approximations. We further develop an architecture for such bounded $\epsilon$-reach set computation which decouples the basic algorithm for an $\epsilon$-reach set with given parameter values from 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, Verification

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

1. INTRODUCTION

It is well known that computing the exact reach set of general Hybrid Automaton (HA) is undecidable. In [6, 11] several class of decidable HA with restricted expressive power have been identified. In recent years, research in hybrid system verification has focused on algorithms computing over-approximations of the reachable states of various classes of HA [2, 10]. In [5], 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 [4], a conservative over-approximation of a reach set of a HA is computed through on-the-fly overapproximation of the phase portrait, which is a variation of the approximation in [5]. In [3], 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 [7], we have identified a class of hybrid automata, called Deterministic Transversal Linear Hybrid Automata (DTLHA)\(^1\). We also have shown a new approach to compute an over-approximation of the reach set, with arbitrarily small approximation error $\epsilon$, up to a finite time, from an initial state. We refer to such a set as a bounded $\epsilon$-reach set. The class of DTLHA consists of linear systems with constant inputs (i.e., where the right hand sides of the differential equations consist of the superposition of a term that is linear in the state and a constant input), 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 solutions of 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 and show that one

\(^1\)Abbreviated simply as DLHA in [7]. In the hybrid system literature [1, 5] the word “linear automaton” has been used to denote a system where the differential equations and inequalities involved have constant right hand sides. However, this does not conform to the standard notion of linearity where the right hand side is allowed to be a function of state. We use the term “linear” in this latter more mathematically standard way that therefore encompasses a larger class of systems.
Preparatory Background
We consider the problem of the calculation of an approximate reach set of a special class of Hybrid Automaton (HA) under some assumptions on the discrete transitions. More precisely, given an initial state \( x_0 \), an approximation parameter \( \epsilon \), a time bound \( T \), and a jump bound \( N \), we would like to compute a set \( S \) such that it contains the actual set of states that are reachable from \( x_0 \) in \( T \) time or \( N \) jumps (whichever happens earlier), and does not contain any states which are more than \( \epsilon \) away from the actual reachable states. 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 \( \mathcal{C} := \{ C_1, \ldots, C_m \} \), that is

\[
\bigcup_{i=1}^m C_i = \mathcal{X}, \quad \text{s.t.} \quad C_i \cap C_j = \emptyset \quad \text{for} \quad i \neq j,
\]

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

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

In the sequel, for each \( l \in \mathcal{L} \), we use \( A(l), u(l), \text{Inv}(l) \), and \( \text{Inv}(l) \), respectively.

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

\[
\dot{\eta}(t) = A(l)\eta(t) + u(t), \quad (2)
\]

(b) \( \eta(t) \in \text{Inv}(l) \) for every \( t \in [0, t] \).

For such a trajectory \( \eta \), its duration is \( t \), and it is denoted by \( \eta \text{dur} \).

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 \( l_k \) to a location \( l_j \), occurring at \( x(\tau_k) \in \partial\text{Inv}(l_j) \cap \partial\text{Inv}(l_i) \).](image)

**Definition 3.** An execution \( x \) of an LHA \( A \) from a starting state \( (l_0, x_0) \in \mathcal{L} \times \mathbb{R}^n \) 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) \( t = \sum_{k=0}^\infty \eta_k \text{dur} \), (b) \( x(0) = \eta_0(0) = x_0 \in \text{Inv}(l_0) \), (c) \( x(\tau_k) = \eta_k(\tau_k) = \eta_{k-1}(\tau_{k-1} \text{dur}) \) for \( k \geq 1 \), (d) \( x(\tau) = \eta_{k-1}(\tau_{k-1} \text{dur}) \) for \( \tau \in [\tau_{k-1}, \tau_k) \), where \( \tau_0 = 0 \), and \( \tau_k = \sum_{i=0}^{k-1} \eta_i \text{dur} \) for \( k \geq 1 \). Note that \( \tau_k \) for \( k \geq 1 \) represents the time at the \( k \)-th discrete transition between locations and the continuous state is not reset during discrete transitions.

**Definition 4.** For \( l_i, l_j \in \mathcal{L} \), a discrete transition from \( l_i \) to \( l_j \) occurs at a continuous state \( x(\tau') \) at time \( \tau' \), whenever \( x(\tau') \in \text{Inv}(l_i) \cap \text{Inv}(l_j) \) and \( x(\tau') = \lim_{\tau \to \tau'} x(\tau) \) where \( x(\tau) \in (\text{Inv}(l_j)^c) \) 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 \( l_j \in \mathcal{L} \) to which a discrete transition state \( x(\tau_k) \) 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 \( x(\tau_k) \):

\[
\langle x(\tau_k), \vec{n}_i \rangle \geq \epsilon \quad \land \quad \langle x(\tau_k), \vec{n}_j \rangle \geq \epsilon,
\]

where \( \vec{n}_i \) is an outward normal vector of \( \partial\text{Inv}(l_i) \) at \( x(\tau_k) \), and \( \vec{x}(\tau_k) = A(l_i) x(\tau_k) + u_i \), and \( \vec{x}(\tau_k) = A_j x(\tau_k) + u_j \) 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 condition. 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 does not occur if a discrete transition is transversal discrete transition.

**Definition 6.** Given an LHA \( A \), a starting state \( (l_0, x_0) \in \mathcal{L} \times \mathbb{R}^n \), a time bound \( T \), and a jump bound \( N \), we call an LHA \( A \) as a Deterministic and Transversal Linear Hybrid Automaton (DTLHA) if all discrete transitions in the execution starting from \( x_0 \) up to time \( T \) or up to \( N \) transitions (whichever is earlier) are deterministic and transversal.

**Definition 7.** A continuous state in \( \mathcal{X} \) is reachable if there exists some time \( t \) at which it is reached by some execution \( x \).
Definition 8. Given a time $t$, the bounded reach set up to time $t$, denoted as $R_t(x_0)$, of a DTLHA $A$ is defined to be the set of continuous states that are reachable for some time $\tau \in [0, t]$ by some execution $x$ starting from $x_0 \in Inv_0$.

Definition 9. Given $\epsilon > 0$, a set of continuous states $S$ is called a bounded $\epsilon$-reach set of a DTLHA $A$ over a time interval $[0, t]$ from an initial state $x_0$ if $R_t(x_0) \subseteq S$ and
\[
d_H(R_t(x_0), S) \leq \epsilon.
\] where $d_H(P, Q)$ denotes the Hausdorff distance between two sets $P$ and $Q$.

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

Our results and algorithm 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 $D_t(P)$ to denote the set of states reached at time $t$ from a set $P$ at time $0$. We also use $D_t(P, \gamma)$ to denote an over-approximation of $D_t(P)$ with an approximation parameter $\gamma > 0$, and calling it a $\gamma$-approximation of $D_t(P)$ if it satisfies (i) $D_t(P) \subseteq D_t(P, \gamma)$ and (ii) $d_H(D_t(P), D_t(P, \gamma)) \leq \gamma$. Note that $D_0(P, \gamma)$ is simply a $\gamma$-approximation of the set $P$.

3. THEORY

In this section, we first present the theoretical results for bounded $\epsilon$-reachability of a DTLHA under the assumption of infinite precision calculation made in [7]. We then derive a set of conditions that can be used to determine the event of a deterministic and transversal discrete transition in computing a bounded $\epsilon$-reach set of a DTLHA which is discussed in more detail in Section 5. In the last part of this section, these results are extended to show that a bounded $\epsilon$-reach set of a DTLHA can be computed without infinite precision calculation capability.

3.1 Bounded $\epsilon$-Reachability of a DTLHA

The approach to compute a bounded $\epsilon$-reach set of a DTLHA from an initial state $x_0$ in [7] is to over-approximate the bounded reach set through sampling and polyhedral over-approximation. More precisely, for given parameters $\delta$ and $\gamma$, and a sampling period $h$, the bounded reach set of a DTLHA from $x_0$ up to time $t_f$ is over-approximated by
\[
\bigcup_{m=0}^{M} D_{mk}(B_\delta(x_0), \gamma)
\] where $B_\delta(x_0)$ is a polyhedral $\delta$-neighborhood of $x_0$, $\gamma$ is a parameter which defines the size of over-approximation of $D_t(B_\delta(x_0))$ for $\tau \in [0, t_f]$, and $M := \lfloor t_f/h \rfloor$.

In this approach, the existence of appropriate values for parameters $\delta$, $\gamma$, and $h$ is in fact critical in computing a bounded $\epsilon$-reach set of a DTLHA from $x_0$. In [7], we showed that for any given $\epsilon > 0$, there exist values for those parameters such that the set in (5) is indeed a bounded $\epsilon$-reach set of a DTLHA from $x_0$ if every discrete transition is deterministic and transversal.

We present the main results from [7] as follows:

Lemma 1. Given $\gamma > 0$, if a sampling period $h$ satisfies the following inequality in (6), then $D_{t_f}(B_\delta(x_0)) \subseteq D_t(B_\delta(x_0), \gamma)$ for $\tau \in [t, t + h]$ for each sample time $t$:
\[
h < \frac{\gamma}{\bar{v}}
\] where $\bar{v} := \max_{i \in \mathbb{L}} \{\|A_i(\|\bar{x} + \|u_i\|\)} \text{ and } \bar{x} := \max_{x \in X} \|x\|

Lemma 2. Given $\epsilon > 0$, a DTLHA $A$, an initial state $(l_0, x_0) \in \mathbb{L} \times \mathbb{R}^n$, and a time bound $t_f$, there exist $\delta > 0$, $\gamma > 0$, and $h > 0$ such that the following hold:

(i) $R_{t_f}(x_0) \subseteq \bigcup_{m=0}^{M} D_{mk}(B_\delta(x_0), \gamma),$

(ii) $\text{dia}(D_{mk}(B_\delta(x_0), \gamma)) < \epsilon \ \forall m \in \{0, 1, \ldots, M\},$

(iii) Suppose $x(\tau_0) \in \partial Inv_\psi$, $x(\tau_k) = \lim_{\tau \to \tau_k} x(\tau)$, and $\tau_k < t_f$ where $x(\tau) \in (Inv_\psi)^0 \forall \tau \in (\tau_k - \eta, \tau_k)$ for some $\eta \in \mathbb{L}$ and $\eta > 0$. Then $D_{t_f-h}(B_\delta(x_0)) \subseteq (Inv_\psi)^0$, $D_t(B_\delta(x_0)) \subseteq (Inv_\psi)^0$, $\tau_k \in (t-k-h, t)$, and $t < t_f$ where $D_t(B_\delta(x_0))$ is computed under the LTI dynamics of $l_i \in \mathbb{L} \forall \tau \in [0, t_f],$

(iv) Suppose (iii) holds and $x(\tau_k)$ makes a discrete transition from a location $l_i$ to some other location $l_{i-1} \in \mathbb{L}$. Then $(D_{t_f-h}(B_\delta(x_0), \gamma) \cap Inv_\psi) \subseteq J_i \cap h < \Delta$ for some appropriate $\delta > 0$ and $\Delta > 0$ such that $B_\psi(x(\tau_k)) \subseteq (Inv_\psi \cup Inv_j)$ and
\[
\bigcup_{y \in J_i \cap h < \Delta} D_t(y) \subseteq (Inv_\psi)^0 \forall \tau \in (\tau_k, \tau_k + \Delta),
\] where $J_i \subseteq B_\psi(x(\tau_k)) \cap Inv_\psi \cap Inv_j$.

In summary, for a given bounded reach set $R_{t_f}(x_0)$ of a DTLHA $A$ from $x_0$, the above results state the following:
(1) A sampling period $h > 0$ can be determined for any given $\gamma > 0$ so that the bounded reach set can be over-approximated. (2) If there is a discrete transition, then this event can be determined through the over-approximation of sampled states with an appropriate values of $\delta$ and $h$.

3.2 Conditions for Determination of Deterministic and Transversal Discrete Transition

Now, we elaborate in more detail on the result given in Lemma 2, especially on (iii) and (iv), to develop some conditions which are used in Section 5.
Lemma 3. Given a location \( l_c \), if \( D_{t-h}(B_h(x_c)) \subset (Inv_{c})^o \) and \( D_{t}(B_h(x_c)) \subset Inv_{c}^o \) for some \( \delta > 0 \) and \( h > 0 \) where \( B_h(x_c) \) is a \( \delta \)-neighborhood of the initial state \( x_c \), then there is a discrete transition from the location \( l_c \), within time \((t-h, t)\).

Proof. Note \( D_{t}(x_c) \in D_{t}(B_h(x_c)) \), where \( D_{t}(x_c) \) is the reached state at time \( t \) from \( x_c \). Similarly, \( D_{t-h}(x_c) \in D_{t-h}(B_h(x_c)) \). From the hypothesis, \( D_{t}(x_c) \in Inv_{c}^o \) and \( D_{t-h}(x_c) \in Inv_{c}^o \). This implies that there exists \( t \in (t-h, t) \) such that \( D_{t}(x_c) \in Inv_{c}^o \) for \( s \in [t-h, t) \) and \( D_{t-h}(x_c) \in Inv_{c}^o \) for \( s \in (t, t-h) \). Hence there is a discrete transition at some time \( t \in (t-h, t) \). \( \square \)

Lemma 4. Given a polyhedron \( P_t \) at a time \( t \), suppose that there is a discrete transition from a location \( l_c \) to another locations, i.e., \( P_{t-h} \subset (Inv_{c})^o \) and \( P_t \subset Inv_{c}^o \) for some \( h > 0 \). Then the discrete transition is deterministic if there exists a location \( l_d \) such that \( l_d \neq l_c \) and \( P_t \subset (Inv_{c})^o \).

Proof. By Definition 5, the result is trivially true. \( \square \)

Lemma 5. Given polyhedron \( P_t \) at time \( t \), \( \gamma > 0 \), and \( h > 0 \) satisfying (6), suppose that there is a deterministic discrete transition from a location \( l_c \) to a location \( l_d \), i.e., \( P_{t-h} \subset (Inv_{c})^o \) and \( P_t \subset Inv_{c}^o \) for some \( h > 0 \). Then for any \( \epsilon > 0 \), the discrete transition is transversal if the following conditions hold:

(i) \( h < (\text{dia}(J_{c,n})/2)/(2\mu) \)
(ii) \( D_{0}(J_{c,n}, \text{dia}(J_{c,n})/2) \subset \{ Inv_c \cup Inv_{c'} \} \)
(iii) \( \langle \bar{x}, n \rangle \geq \epsilon \land \langle \bar{x}, n \rangle \geq \epsilon, \forall x \in V(J_{c,n}) \)

where \( J_{c,n} := D_{0}(P_t, \gamma) \cap Inv_{c} \cap Inv_{c'} \), \( J_{c,n} := D_{0}(J_{c,n}, \text{dia}(J_{c,n})/2) \cap Inv_{c} \cap Inv_{c'} \), \( \bar{v} \) is as defined in (6), \( V(P) \) is a set of vertices of a polyhedron \( P \), \( n \) is an outward normal vector of \( \partial Inv_{c} \), and \( \bar{x} \) is a vector flow evaluated with respect to the LTI dynamics of location \( l_c \in L \).

Proof. First note that \( P_{t-h} \subset (Inv_{c})^o \) and \( P_t \subset (Inv_{c})^o \), since there is a deterministic discrete transition from \( l_c \) to \( l_d \). Since \( \gamma \) and \( h \) satisfy (6), \( P_{t-h} \subset D_{0}(P_t, \gamma) \). In fact, \( \cup_{\gamma < h} x(z) \subset D_{0}(P_t, \gamma) \) for \( r \in [0, h] \) where \( x(z, \sigma) := e^{At-r}z + \int_0^{r} e^{A(t-s)}w ds \). Since \( D_{t-h}(x_c) \in P_{t-h}(x_c) \) and \( D_{t}(x_c) \in P_t(x_c) \), \( D_{t-h}(x_c) \subset (Inv_{c} \cap Inv_{c'}) \cup (Inv_{c} \cap Inv_{c'}) \) for some \( z \in (t-h, t) \) where \( D_{t}(x_c) \) is a discrete transition state from \( l_c \) to \( l_d \) at time \( t \). Thus \( J_{c,n} \not= \emptyset \) (more precisely, \( J_{c,n} \neq \emptyset \)) and it is in fact an over-approximation of the deterministic discrete transition \( x(z) \subset Inv_{c} \cap Inv_{c'} \).

Notice that if (i) holds, then \( \|x(h, z) - z\| < \text{dia}(J_{c,n})/4 < \text{dia}(J_{c,n})/2 \) for any \( z \in J_{c,n} \), since \( \|x(h, z) - z\| < \epsilon h \) where \( x(h, z) \) is the state reached from \( z \) at \( h \) time under the LTI dynamics of the location \( l_c \) and \( \bar{v} \) is defined in (6). Also notice that if (ii) and (iii) hold, then for any \( z' \in J_{c,n} \), \( z' \) satisfies the deterministic and transversal discrete transition condition in Definition 5. If we now consider the fact that \( \text{dia}(J_{c,n}) \geq 2 \cdot \text{dia}(J_{c,n}) \), then \( x(z) \subset Inv_{c}^o \) for \( z \in (0, h) \). Since \( z \in J_{c,n} \) is arbitrary, it is easy to see that \( D_{t}(J_{c,n}) \subset Inv_{c}^o \) for \( z \in (0, h) \). \( \square \)

3.3 Bounded \( \epsilon \)-Reachability of a DTALHA with Finite Precision Calculations

The results in Section 3.1 and 3.2 rely on the assumption that the following quantities can be computed exactly:

- \( x(t, x_c) = e^{At}x + \int_0^t e^{A(t-s)}uds \)
- \( H \cap P \) where \( H \) is a hyperplane and \( P \) is a polyhedron.
- \( \text{hull}(V) \) where \( \text{hull}(V) \) is the convex hull of \( V \) which is a finite set of points in \( \mathbb{R}^n \).

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

3.3.1 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 on 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.

We assume that a set of subroutines or functions are available for approximately computing these quantities, which we use to compute a bounded \( \epsilon \)-reach set. More precisely, for given \( \mu_c \) and \( \sigma_c \), \( a(H \cap P, \mu_c) \) and \( a(\text{hull}(V), \mu_v) \) are available. Moreover, we also assume that a set of approximate computations, specifically, of \( a(e^{At}, \sigma_c) \), \( a(\int_0^t e^{At}dr, \sigma_c) \), \( a(A \cdot \sigma_c) \), and \( a(u + v, \sigma_a) \), are available in computing \( x(t, x_0) \) for given approximation errors \( \sigma_c, \sigma_v, \sigma_a \), and \( \sigma_c \). From these approximate computational capabilities, we can derive an upper bound on the approximation error, denoted as \( \mu_z \), 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 \( x(x_0 - 1_{n \times m}) \leq a(x, y) \leq x + y \cdot 1_{n \times m} \) where \( x \in \mathbb{R}^{n \times m} \) and \( 1_{n \times m} \) is an \( n \times m \) matrix whose every element is 1.

With this, we derive \( \mu_z \) as follows:

- \( e^{At} - \sigma_c \cdot 1_{n \times n} \leq a(e^{At}, \sigma_c) \leq e^{At} + \sigma_c \cdot 1_{n \times n} \)
- \( e^{At}x_0 - (\sigma_c|x_0| + \sigma_p) \cdot 1_{n \times 1} \leq a(e^{At}x_0, \sigma_c) \leq e^{At}x_0 + (\sigma_c|x_0| + \sigma_p) \cdot 1_{n \times 1} \)

Similarly,

\[ \int_0^t e^{As}ds \cdot u + (\sigma_i|u| + \sigma_p) \cdot 1_{n \times 1} \leq \int_0^t e^{As}ds \cdot u + (\sigma_i|u| + \sigma_p) \cdot 1_{n \times 1} \]

Hence, we have

- \( x(t, x_0) - \delta_z \leq a(x(t, x_0), \delta_z) \leq x(t, x_0) + \delta_z \)

where \( \delta_z := (2\sigma_v + \sigma_a + \sigma_v|x_0| + \sigma_i|u|) \cdot 1_{n \times 1} \).

Now, we define \( \mu_z \) as the maximum of \( \delta_z \) over the continuous state space \( X \) and the control input domain \( U \),

\[ \mu_z := \max_{x \in X, u \in U} |\delta_z| \]

3.3.2 Incorporation of Finite Precision Calculations in Bounded \( \epsilon \)-Reachability of a DTALHA

In this section, we extend the result given in Sections 3.1 and 3.2 to relax the infinite precision computation assumption. Especially, we extend the results in Lemmas 1, 3, 4,
and $h$. We first discuss the relation between $h$ and $\gamma$ in Lemma 1 can be changed under finite precision computation.

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

$$\bigcup_{x(\tau)} x(\tau; z) \subset B_\gamma(x(t)), \quad \forall \tau \in [0, h],$$

where $x(\tau; z) = e^{At} z + \int_0^\tau e^{As} u ds$ and $B_\gamma(x)$ is a $\gamma$-neighborhood around $x$.

**Proof.** Notice that $a(x(t), \rho) \in B_\rho(x(t))$ and for any $x(t) \in \mathcal{X}$,

$$||x(t + h) - x(t)|| \leq \int_t^{t+h} ||x(s)||ds,$$

Since $h < (\gamma - \rho)/(||A||x + ||u||)$, $||x(t + h) - x(t)|| < \gamma - \rho$ for any $x(t) \in \mathcal{X}$. In fact, $||x(t + s) - x(t)|| < \gamma - \rho$ for all $s \in [0, h]$. Hence for any $z \in B_\rho(x(t))$, $x(s; z) \in B_{\gamma - \rho}(x)$ for $s \in [0, h]$. This implies that for any $z \in B_\rho(x(t))$, $||x(t) - x(s; z)|| \leq ||x(t) - z|| + ||z - x(s; z)|| \leq \gamma$.

**Lemma 7.** Given $\rho > 0$ for $a(D_t(B_\rho(x_0)), \rho)$, let $P_t := D_t(B_\rho(x_0), \rho)$. Then if $h$ satisfies the inequality in (10) for a given $\gamma > \rho$, then $D_t(P_t) \subset D_t(B_\rho(x_0), \gamma)$ for all $\tau \in [0, h]$.

$$h < \frac{\gamma - \rho}{\bar{v}}$$

where $\bar{v}$ is as defined in (6).

**Proof.** Let $\mathcal{Y}$ and $\mathcal{Y}'$ be the set of extreme points of $D_t(B_\rho(x_0))$ and $P_t$, respectively. Since (10) hold, we know from Lemma 6 that for each $x(t) \in \mathcal{Y}$, $D_t(B_\rho(x(t))) \subset B_\gamma(x(t))$ for all $\tau \in [0, h]$. For each $z \in \mathcal{Y}'$, there exists $x(t) \in \mathcal{Y}$ such that $||x(t) - z|| \leq \rho$. Notice that for each $z \in \mathcal{Y}'$, $x(s; z) \in P_t(x(t))$ for some $s \in [0, h]$. Therefore, $D_t(P_t) \subset D_t(B_\rho(x_0), \gamma)$ for all $\tau \in [0, h]$.

In the sequel, we use $\dot{x}$ to denote $a(x, \rho)$ for a given approximation error bound $\rho$ for simplicity of notation.

The condition (ii) in Lemma 2 enforces the size of an overapproximation of each sampled state along $\mathcal{R}_{\tau}^{t}(x_0)$ to be less than the given $\epsilon > 0$. Under the finite precision calculations, it is straightforward to extend the result of (ii) in Lemma 2 as shown in the following Lemma.

**Lemma 8.** Given $\epsilon > 0$, $\rho > 0$, a polyhedron $\mathcal{P}$, and $\mathcal{P}$, if $\text{diag}(\mathcal{P}) < \epsilon - \rho$, then $\text{diag}(\mathcal{P}) < \epsilon$.

**Proof.** Recall that $\hat{\mathcal{P}} := a(\mathcal{P}, \rho)$. This implies $d_H(\mathcal{P}, \rho) < \rho$. Hence $\mathcal{P} \subset D_\rho(\hat{\mathcal{P}}, \rho)$. Notice that $\text{diag}(D_\rho(\hat{\mathcal{P}}, \rho)) \leq \text{diag}(\hat{\mathcal{P}}) + \rho$. Hence $\text{diag}(\hat{\mathcal{P}}) < \epsilon - \rho$ implies $\text{diag}(\hat{\mathcal{P}}) < \epsilon$.

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 $\rho$ at time $t$, then there is a deterministic and transversal discrete transition at some time $\tau \in [t - h, t]$.

**Lemma 9.** Given $\rho > 0$, a location $l_c$, and $D_t(B_\rho(x_0))$ at time $t$, if $D_{t-h}(B_\rho(x_0), \rho) \subset \text{Int}_c$ and $D_t(B_\rho(x_0), \rho) \subset \text{Int}_c$ for some $\delta > 0$ and $h > 0$, then there is a discrete transition from the location $l_c$.

**Proof.** Since $d_H(D_t(B_\rho(x_0)), D_t(B_\rho(x_0), \rho)) \leq \rho$, $D_t(B_\rho(x_0), \rho) \subset D_t(B_\rho(x_0), \rho)$. Similarly, $D_{t-h}(B_\rho(x_0)) \subset D_{t-h}(B_\rho(x_0), \rho)$. Hence $D_t(B_\rho(x_0)) \subset \text{Int}_c$ and $D_{t-h}(B_\rho(x_0), \rho) \subset \text{Int}_c$.

Then the result follows immediately from Lemma 3.

**Lemma 10.** Given $\rho > 0$, a location $l_c$, and a polyhedron $\mathcal{P}_t$ at time $t$, suppose that there is a discrete transition from a location $l_c$ to some other locations, i.e., $D_\rho(l_c, \rho) \subset \text{Int}_c$ and $D_\rho(l_c, \rho) \subset \text{Int}_c$ for some $h > 0$. Then there is a deterministic discrete transition from $\mathcal{P}_t$ to $l_c$ if there exists a location $l_n$ such that $l_n \neq l_c$ and $D_\rho(l_n, \rho) \subset \text{Int}_c$.

**Proof.** Since $P_{t-h} \subset D_\rho(l_{t-h}, \rho), P_{t-h} \subset \text{Int}_c$. Similarly, $P_t \subset \text{Int}_c$ since $P_t \subset D_\rho(l_t, \rho)$. Then by Lemma 4, the conclusion holds.

**Lemma 11.** Given $\rho > 0$, $\gamma > 0$ and $h > 0$ satisfying (10), and a polyhedron $\mathcal{P}_t$ at time $t$, suppose that there is a deterministic discrete transition from a location $l_c$ to a location $l_n$, i.e., $D_\rho(l_{t-h}, \rho) \subset \text{Int}_c$ and $D_\rho(l_t, \rho) \subset \text{Int}_c$ for some $h > 0$. Then for any $\epsilon > 0$, the discrete transition is transversal if the following conditions hold:

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

(ii) $D_\rho(\mathcal{J}_{c,n}, c_{n}) \subset \text{Int}_c \cup \text{Int}_n$

(iii) $\langle \hat{\mathcal{J}}, n \rangle \geq \epsilon \wedge \langle \mathcal{J}, n \rangle \geq \epsilon \forall \mathcal{J} \in \mathcal{J}_{c,n}$

where $\mathcal{J}_{c,n} := D_\rho(\mathcal{J}, \gamma) \cap \text{Int}_c \cap \text{Int}_n$, $\mathcal{J}_{c,n} := D_\rho(\mathcal{J}_{c,n}, c_{n}) \cap \text{Int}_c \cap \text{Int}_n$, and $\hat{\mathcal{J}}$ and $\tilde{n}$ are as defined in Lemma 5.

**Proof.** Notice that $D_\rho(\mathcal{J}, \gamma) \subset D_\rho(\mathcal{J}, \gamma + \rho)$ since $d_H(\mathcal{J}, \tilde{n}) \leq \rho$. Then, by the definition of $\mathcal{J}_{c,n}$ given in Lemma 5 and $\mathcal{J}_{c,n}$, we know $\mathcal{J}_{c,n} \subset \mathcal{J}_{c,n}$. Hence, $\mathcal{J}_{c,n} \neq \emptyset$ and in fact it is an over-approximation of the deterministic discrete transition state as is $\mathcal{J}_{c,n}$ in Lemma 5.

By the same argument used in the proof of Lemma 5, if (i) holds, then $D_\rho(\mathcal{J}_{c,n}, \gamma) \subset D_\rho(\mathcal{J}_{c,n}, \gamma + \rho)$ since $d_H(\mathcal{J}, \tilde{n}) \leq \rho$. Hence, (ii) and (iii) imply that $D_\rho(\mathcal{J}_{c,n}) \subset \text{Int}_c$ for $\tau \in [0, h]$. Therefore, the conclusion holds since $\mathcal{J}_{c,n} \subset \mathcal{J}_{c,n}$.

**4. ARCHITECTURE**

In [7], we have shown that an approximate bounded reach set of a DTLHA from an initial continuous state $x_0$ in an initial location $l_0$ can be computed with arbitrarily small approximation error $\epsilon$ under the assumption that every discrete transition is deterministic and transversal. We also have proposed an algorithm for such bounded $\epsilon$-reach set computation. Algorithm 1 shows the main computation steps of the proposed algorithm where $B_\rho(x_0)$ is a $\delta$-neighborhood of a given initial state $x_0$. $D_t(B_\rho(x_0), \gamma)$ is a $\gamma$-approximation of $D_t(B_\rho(x_0)), \rho$, and $h$ is a sampling period corresponding to the value of $\gamma$ satisfying an over-approximation condition in (6). Note that $\mathcal{R}$ returned from the algorithm is in fact a bounded $\epsilon$-reach set as stated in the following theorem.

**Theorem 1.** Given input $(A, N, T, l_0, x_0, \epsilon)$, Algorithm 1 terminates in a finite number of iterations and returns $\mathcal{R}$, a
bounded $\varepsilon$-reach set of $A$ from $x_0 \in Inv_0$ up to time $t_f := \min\{\gamma N, T\}$, if $A$ is a DTLHA up to $t_f$:

$$\mathcal{R} := \bigcup_{m=0}^{M} D_{mh}(B_m(x_0), \gamma)$$

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

**Algorithm 1:** An algorithm proposed in [7] for a bounded $\varepsilon$-reach set of a DTLHA $A$ from an initial state $x_0 \in Inv_0$.

**Input:** $A, N, T, l_0, x_0, \varepsilon$

- Initialize $\gamma$, $\delta$, and $\varepsilon$ with arbitrary positive real values.
- Initialize $t = 0$, $jump = 0$, and $\mathcal{R} = \emptyset$.

while true do

  Compute $B_m(x_0)$ and $h$ from $\gamma$.
  Compute $D_l(B_m(x_0))$ and $D_l(B_m(x_0), \gamma)$.
  if $\text{dist}(D_l(B_m(x_0), \gamma)) > \varepsilon$ then

    Reduce $\delta$, $\gamma$, and goto •

  end

  if discrete transition then

    if deterministic $\wedge$ transversal then

      $t \leftarrow t + h$
      $\text{jump} \leftarrow \text{jump} + 1$
      $\mathcal{R} \leftarrow \mathcal{R} \cup D_l(B_m(x_0), \gamma)$
    else Reduce $\delta, \gamma$, and goto •

  else $t \leftarrow t + h$ and $\mathcal{R} \leftarrow \mathcal{R} \cup D_l(B_0(x_0), \gamma)$.

  if $(t \geq T) \vee (\text{jump} \geq N)$ then return $\mathcal{R}$.

end

In the above algorithm and accompanying theoretical results for a bounded $\varepsilon$-reach set computation in [7], we have not addressed the issue of computation with finite precision. Moreover, even though the proposed algorithm can compute a bounded $\varepsilon$-reach set correctly, it is far from being computationally efficient since the algorithm restarts its $\varepsilon$-reach set computation from an initial state $x_0$ at time $t = 0$ whenever the values of $\delta$ and $\gamma$ are changed. Moreover, the algorithm does not provide any flexibility in choosing the values for $\delta$ and $\gamma$ whenever the algorithm needs to be continued with different $\delta$ and $\gamma$ values, since one specific decision rule resulting in $\delta$ and $\gamma$ which are monotonically decreasing is tightly embedded within the algorithm. We address all these issues in the remainder of 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 a bounded $\varepsilon$-reach set computation followed by a new bounded $\varepsilon$-reach set algorithm which is based on the theoretical results developed in Section 3 so that the overall computation process can be better optimized in terms of computational efficiency and flexibility.

One of the main objectives of the architecture design 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. As mentioned above, the proposed architecture consists of roughly four different parts which are Policy (Policy module), Algorithm (Main Algorithm and Condition Checking modules), Data (System Description and Data modules), and Numerical Calculation module. A more detail explanation of each of these modules is given in below.

**Policy.**

This module contains a user-defined rule to choose appropriate values of the parameters, especially $\delta$ and $\gamma$ as shown in Algorithm 1, that are needed to continue to compute a bounded $\varepsilon$-reach set of a DTLHA when an ambiguous situation is encountered in the Main Algorithm module. Furthermore, this module can make a decision about the choice of numerical calculation algorithms which affect to the computational accuracy for each approximate numerical function defined in Section 3.3.1.

**System Description.**

The System Description module contains information about the system, described by a modeling language; it consists of $X$, the domain of continuous state space, a DTLHA $A$, 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, an upper bound $T$ on terminal time, an upper bound $N$ on the total number of discrete transitions, and an approximation parameter $\varepsilon$, are described in the System Description module. In short, all information required to describe a problem of a bounded $\varepsilon$-reach set computation of a DTLHA is contained in the System Description module.

**Data.**

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. Furthermore, the data which are generated on-the-fly in a bounded $\varepsilon$-reach set computation by the Main Algorithm module, called ReachSetHistory and TransitionHistory, are also stored in this module.

**Condition Checking.**

To ensure a correct bounded $\varepsilon$-reach set computation, a bounded $\varepsilon$-reach set algorithm needs to correctly (i) detect a deterministic and transversal discrete transition if there is one, (ii) determine whether the size of the set computed as an over-approximation of the reach set between samples is smaller than the specified parameter $\varepsilon$, and (iii) check whether a sampling period $h$ and an over-approximation parameter $\gamma$ satisfy the relation for over-approximation guarantee. All functions which implement these condition checkings are contained in this module. More detail on this module is given in Section 5.

**Main Algorithm.**

With the inputs from the Policy module, the Main Algorithm computes a bounded $\varepsilon$-reach set 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. During a bounded $\varepsilon$-reach set computation, the Main Algorithm stores its computational state in two data structures, called ReachSetHistory.
5. ALGORITHM

The proposed algorithm for a bounded \( \epsilon \)-reach set computation is decomposed into roughly two parts, the Main Algorithm module and the Condition Checking module. In this section, we discuss these modules in more detail. Recall that we use \( \hat{x} \) to denote \( a(x, \rho) \) for some given approximation error bound \( \rho \in \mathbb{R}^+ \). In particular, a polyhedron \( \hat{P} \) in the sequel should be understood as an approximation of a polyhedron \( P \), i.e., \( \hat{P} := a(P, \rho) \).

5.1 Condition Checking Module

In computing a bounded \( \epsilon \)-reach set, the following set of questions needs to be answered at each step of computation in the Main Algorithm to produce a correct result:

1. Given \( \delta \) and \( \gamma \), is the diameter of \( D_0(B_s(x_0), \gamma) \) at current sample time \( t \) less than the given \( \epsilon \)?
2. Given \( \delta, \gamma, \) and \( h \), is \( D_0(B_s(x_0)) \subset D_0(B_s(x_0), \gamma) \) for all \( t \in [t, t+h] \) at current sample time \( t \)?
3. Given \( h, \delta, \) and \( D_0(B_s(x_0)) \), can we conclude that a discrete transition has occurred between \( t-h \) and \( t \)?
4. If there is a discrete transition as above, is it a deterministic discrete transition?
5. If there is a deterministic discrete transition above, is it a transversal discrete transition?

Corresponding to these questions, the Condition Checking module consists of the following set of functions which are based on the results in Section 3.3.2.

IsEpsilonSmall().

Given \( \epsilon > 0 \) and a polyhedron \( \hat{P} \), this function determines whether \( \text{dia}(P) < \epsilon \) or not, where \( \text{dia}(P) \) denotes the diameter of a polyhedron \( P \). As shown in Lemma 8, \( \text{dia}(P) < \epsilon \) if \( \text{dia}(\hat{P}) < \epsilon - \rho \). Hence, this function returns true if \( \text{dia}(\hat{P}) < \epsilon - \rho \).

IsOverApproximate().

Given \( \gamma \) and \( \rho \), this function determines whether a sampling period \( h \) and \( \gamma \) satisfy the condition (10). Hence, if \( h < (\gamma - \rho)/\bar{v} \), this function returns true where \( \bar{v} \) is as defined (6).

IsTransition().

Given a sampling period \( h \), a location \( l_c \), a polyhedron \( \hat{P}_t \) at time \( t \), this function checks if there is a discrete transition from a location \( l_c \) at some time in between \( t-h \) and \( t \). In Lemma 9, it is shown that if \( D_0(\hat{P}_{t-h}, \rho) \subset (\text{Inv}_v)^\delta \) and \( D_0(\hat{P}_t, \rho) \subset \text{Inv}_v^C \), then there is indeed a discrete transition at some time in \( (t-h, t) \). Assuming that \( D_0(\hat{P}_{t-h}, \rho) \subset (\text{Inv}_v)^\delta \) is satisfied at time \( t-h \), this function returns true if \( D_0(\hat{P}_t, \rho) \subset \text{Inv}_v^C \). If \( D_0(\hat{P}_t, \rho) \subset (\text{Inv}_v)^\delta \), then this function returns false. In the other cases that \( (D_0(\hat{P}_t, \rho) \cap \text{Inv}_v) \neq \emptyset \wedge (D_0(\hat{P}_t, \rho) \cap \text{Inv}_v^C) \neq \emptyset \), this function returns error to inform that other values of \( \delta \) or \( h \) need to be used to resolve the ambiguity.

IsDeterministic().

Given a location \( l_c \) and a polyhedron \( \hat{P} \), this function checks if a discrete transition from \( l_c \) is a deterministic transition to some other location \( l_a \). Based on the result in Lemma 10, this function returns the location \( l_a \) if there is a location \( l_a \in L \) such that \( l_a \neq l_c \) and \( D_0(\hat{P}_t, \rho) \subset (\text{Inv}_v)^\delta \). Otherwise it returns error.

IsTransversal().

Given \( h, \gamma \), and a polyhedron \( \hat{P} \), this function checks if a discrete transition from a location \( l_c \) to other location \( l_a \) is a transversal discrete transition or not, using the conditions (i), (ii), and (iii) in Lemma 11. If it is a transversal
5.2 Main Algorithm Module

Roughly, the Main Algorithm module consists of two parts. The first part is a function called \text{ReachSet}() which is the main function to compute a bounded \(\epsilon\)-reach set, and the second part is a set of functions called \text{Sub-functions} which are called by \text{ReachSet}() during its computation. We first describe the functions defined as Sub-functions.

\text{ReachNext}().

Given \(h, \gamma\), and a polyhedron \(\mathcal{P}\), this function returns \(\hat{D}_h(\mathcal{P})\), an approximation of the linear image of a polyhedron \(\mathcal{P}\) at time \(h\) under a linear dynamics, and \(\hat{D}_h(\mathcal{P}, \gamma)\), an over-approximation of \(D_h(\mathcal{P})\) for a given over-approximation parameter \(\gamma\). This function also returns estimates of the upper bound of computation errors \(\rho'\) and \(\rho''\) along with \(\hat{D}_h(\mathcal{P})\) and \(\hat{D}_h(\mathcal{P}, \gamma)\), so that \text{ReachSet}() function can keep track of the numerical errors accumulated from the initial time up to the current time \(t\). Notice that \(\rho'\) and \(\rho''\) are defined via \(d_H(\hat{D}_h(\mathcal{P}), \hat{D}_h(\mathcal{P})) \leq \rho'\) and \(d_H(\hat{D}_h(\mathcal{P}, \gamma), \hat{D}_h(\mathcal{P}, \gamma)) \leq \rho''\), respectively.

To compute \(\hat{D}_h(\mathcal{P})\) and \(\hat{D}_h(\mathcal{P}, \gamma)\), this function exploits the fact that the polyhedral structure is preserved under a linear dynamics in the following way. Given polyhedron \(\mathcal{P}\), this function first computes \(V(\mathcal{P})\) which is a 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 \(\mathcal{P}\) itself is 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 \(v_i \in V(\mathcal{P})\), it computes \(v_i(h) := e^{Ah}v_i + \int_0^h e^{Au}uds\) where \(A\) and \(u\) are given by the linear dynamics of a location on which the linear image of \(\mathcal{P}\) is computed. If we let \(V_h(\mathcal{P}) := \{v_i(h) : v_i \in V(\mathcal{P})\}\), then \(\hat{D}_h(\mathcal{P}) := \text{hull}(V_h(\mathcal{P}))\) where \(\text{hull}(V_h(\mathcal{P}))\) is the convex hull of \(V_h(\mathcal{P})\). Notice that what we really have here is \(\hat{D}_h(\mathcal{P})\), and not \(D_h(\mathcal{P})\), since there is a numerical calculation error in the \(\text{hull}(V_h(\mathcal{P}))\) computation. From \(V_h(\mathcal{P})\), this function can also compute \(\hat{D}_h(\mathcal{P}, \gamma)\) easily. For each \(v_i(h) \in V_h(\mathcal{P})\), it first constructs a hypercubic \(\gamma\)-neighborhood of \(v_i(h)\). If we denote such a neighborhood by \(B_{\gamma}(v_i(h))\), then the convex hull of the set of vertices of \(B_{\gamma}(v_i(h))\) for all \(v_i(h) \in V_h(\mathcal{P})\) defines a \(\hat{D}_h(\mathcal{P}, \gamma)\).

\text{AltTransition}().

This function is called by \text{ReachSet}() when a discrete transition from a given location \(l_i\) is detected by \text{IsTransition}(). Then this function internally calls \text{IsDeterministic}() and \text{IsTransversal}() functions to check if this discrete transition is deterministic and transversal. If it is, then this function returns a location \(l_i\) which is returned by \text{IsDeterministic}() and \(\hat{D}_h(\mathcal{J}_{e.n}, \rho')\) which is returned by \text{IsTransversal}(). However, if any of these functions returns \text{error}, this function returns the same \text{error} to indicate the necessity of a decision in the Policy module to resolve the erroneous situation.

\text{ImageAt}().

Even though the overall computational efficiency of a bounded \(\epsilon\)-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 \(\delta\) which defines an initial neighborhood around an initial state is changed. If the algorithm encounters such a situation, \text{ImageAt}() can be used to reduce the number of computational steps. Given \(t\) and \(\delta\), the goal of this function is to compute \(\hat{D}_t(B_h(x_0))\). More precisely, this function computes \(\hat{D}_t(B_h(x_0))\) and a corresponding numerical calculation error \(\rho\) such that \(d_H(\hat{D}_t(B_h(x_0)), \hat{D}_t(B_h(x_0))) \leq \rho\). To compute \(\hat{D}_t(B_h(x_0))\) from \(B_h(x_0)\), this function needs to know is the computational history of \text{ReachSet}() containing the time \(\tau_k\) when a discrete transition is detected and the values of the parameters \(h, \gamma\) that were used at the time \(\tau_k\). Note that all of these informations are stored in \text{TransitionHistory} by \text{ReachSet}().

Now, we describe the main function, called \text{ReachSet}(), in the Main Algorithm.

\text{ReachSet}().

Given an input \((k, \delta, \gamma, h)\) from the Policy module, where \(k\) indicates one of the past computation steps of \text{ReachSet}() from which this function starts its computation, this function computes a bounded \(\epsilon\)-reach set by utilizing all other functions in the Main Algorithm and the Condition Checking modules. This function first retrieves the computation data at the \((k - 1)\)-th computation step from the \text{ReachSetHistory} and starts its \(k\)-th computation step using this data. As shown in Algorithm 2, it continues its computation until it either successfully computes a bounded \(\epsilon\)-reach set or encounters some \text{error}. If there is an \text{error} from any of the functions that are called, then this function returns the same \text{error} to the Policy module to indicate the cause of the \text{error}. For each types of \text{error}, \text{ReachSet}() expects to have a new input from the Policy module to continue its computation. Besides the input \((k, \delta, \gamma, h)\), an additional set of inputs \((\sigma_x, \sigma_p, \sigma_h, \mu_c, \mu_h)\) can also be provided by the Policy module when there are numerical computational algorithms with better computational accuracies in the Numerical Calculation modules to resolve an erroneous situation occurring in \text{ ReachSet}().

As mentioned in Section 4, \text{ReachSet}() stores its computation results (or states) in \text{ReachSetHistory} data structure at every step of its computation. The information stored in \text{ReachSetHistory} includes \(k\) which is the step of its computation, and the time \(x_k\) at the \(k\)-th computation step, \((\delta_k, \gamma_k, h_k)\) that are used in the \(k\)-th computation step without causing any \text{error}, and \(\hat{D}_t(B_h(x_k))\) and \(\hat{D}_t(B_h(x_k)), \gamma_k\) along with their corresponding numerical calculation errors, \(\rho_k\) and \(\rho''_k\). In addition to \text{ReachSetHistory}, \text{ReachSet}() maintains another data structure, called \text{TransitionHistory} which contains computation information of \text{ReachSet}() only at the time of discrete transition between locations, intended to be used in \text{ImageAt}().

We now have the following overall main result:

**Theorem 2.** For a given SystemData := \((X, A, l_0, x_0, T, N, \epsilon)\), if \text{ReachSet}() in Algorithm 2 returns done, then a
Algorithm 2: Algorithm of ReachSet().

Input: $k, \delta_k, \gamma_k, h_k, \sigma_c, \sigma_i, \sigma_p, \sigma_a, \mu_c, \mu_h$

Result: ReachSetHistory, TransitionHistory

compute $\mu_k$ from $(\sigma_c, \sigma_i, \sigma_p, \sigma_a)$

while true do

Get $(k-1)$-th computation data from ReachSetHistory

if $\delta_k \neq \delta_{k-1}$ then

call ImageSet() \rightarrow $\hat{D}_{k-1}(B(h_k(x_0)))$

if error then return error
endif

if IsOverApproximate() = false then return error
endif

$t_k = t_{k-1} + h_k$

end

call ReachNext() \rightarrow $\{\hat{D}_k(B_h(x_0)), \hat{D}_k(B_k(x_0), \gamma_k)\}$

compute $\rho_k$ s.t.

d_H(\hat{D}_k(B_h(x_0), \gamma_k), \hat{D}_k(B_h(x_0), \gamma_k)) \leq \rho_k

if IsEpsilonSmall() = false then return error
endif

if out = error then return error
endif

else if out = false then $k \leftarrow k-1$
endif

end

then call AtTransition() \rightarrow $\{I_k, \hat{P}\}$

if error then return error
endif

$\hat{D}_k(B_h(x_0)) \leftarrow \hat{P}$

$\hat{D}_k(B_h(x_0), \gamma_k) \leftarrow \hat{D}_0(\hat{P}, \gamma_k)$

update $\rho_k$

jump \leftarrow jump + 1

store $\{t_k, I_k, h_k\}$ to TransitionHistory

end

store to ReachSetHistory the data of

\{k, t_k, I_k, \delta_k, \gamma_k, h_k, \rho_k, \hat{D}_k(B_h(x_0)), \hat{D}_k(B_h(x_0), \gamma_k)\}

$k \leftarrow k + 1$

if $(t_k \geq T) \lor (jump \geq N)$ then return done
end

6. IMPLEMENTATION

A prototype implementation of the proposed algorithm for a bounded $\epsilon$-reach set of a DTLHA has been developed on Matlab. We use the Multi-Parametric Toolbox [8] for polyhedral operations.

We now illustrate the results of the implementation on an example. The SystemData := $(X, A, l_0, x_0, T, N, \epsilon)$ of this example is the following: (i) $X := [-8,8] \times [-8,8] \subseteq \mathbb{R}^2$, (ii) $\epsilon = 0.1$ and 0.5, (iii) $T = 10 \text{ sec.}$, (iv) $N = 5$, (v) $A := (\text{L, Inv, A, u})$ where $\text{L} = \{\text{UP, DOWN, LEFT, RIGHT}\}$, and for each $l \in \text{L}$, $A(l)$ and $u(l)$ are defined as shown in Table 1 and Inv(l) is defined as shown in Fig. 3 and Fig. 4. As an example, the invariant set for the location UP, $\text{Inv}(UP)$, is defined as $X \cap (x_1 - x_2 \leq 0) \cap (x_1 + x_2 \geq 0)$, (vi) $x_0 = (2.5, 6)^2$, and (vii) $l_0 = \text{UP}$.

![Figure 3: A bounded $\epsilon$-reach set of $A$ with $\epsilon = 0.1$.](image)

Table 1: $A(l)$ and $u(l)$ for each $l \in \text{L}$ of $A$

<table>
<thead>
<tr>
<th>$l$</th>
<th>$A(l)$</th>
<th>$u(l)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>UP</td>
<td>$(-0.2, -1)$</td>
<td>$0.1$</td>
</tr>
<tr>
<td>DOWN</td>
<td>$(-0.2, -1)$</td>
<td>$-0.2$</td>
</tr>
<tr>
<td>LEFT</td>
<td>$(-0.2, -3)$</td>
<td>$0.15$</td>
</tr>
<tr>
<td>RIGHT</td>
<td>$(-0.2, -3)$</td>
<td>$0.2$</td>
</tr>
</tbody>
</table>

To compute a bounded $\epsilon$-reach set of $A$, we use a policy that (i) uses a fixed value of $\delta = 10^{-5}$ which defines a sufficiently small $B_h(x_0)$, (ii) chooses the value of $h$ and $\gamma$ on-the-fly to resolve erroneous situations, and (iii) chooses $k$ in nondecreasing manner, and (iv) sets a fixed value of $10^{-7}$ for $\sigma_c, \sigma_i, \sigma_p, \sigma_a, \mu_c$, and $\mu_h$. We also set $10^{-7}$ as the minimum value for $h$ and $\gamma$.

The bounded $\epsilon$-reach set for two different values of $\epsilon$ com-
computed by the implementation are shown in Fig. 3 and Fig. 4. For the case of $\epsilon = 0.1$, the algorithm terminates at the computational step $k = 2613$ at which the time $t = 2.2153$ sec. and $\text{jump} = 1$ at the location LEFT. The reason for this early termination is that the maximum sampling period $h$, which is determined by the value of $\epsilon$ and the accumulated numerical calculation errors $\rho$, is not large enough to separate $D_t(B_{\delta}(x_0))$ and $D_{t+h}(B_{\delta}(x_0))$ at the time of discrete transition. Hence the algorithm fails to determine the discrete transition from the location LEFT to the location DOWN. On the contrary, the algorithm successfully returns a bounded $\epsilon$-reach set of $A$ for the case with $\epsilon = 0.5$ as shown in Fig. 4. In this case, the algorithm terminates at the computational step $k = 1364$ at which the time $t = 5.8496$ sec. and $\text{jump} = 5$ at the location LEFT.

### 7. CONCLUSIONS

In this paper, we have extended the theoretical results presented in [7] to compute bounded $\epsilon$-reach sets of Deterministic and Transversal Linear Hybrid Automata with subroutines that only provide finite precision elementary computations. We have also proposed an architecture which separates the elementary subroutines from the policies that adapt the various parameters. This makes the overall algorithm flexible and amenable to different optimizations. An example of a bounded $\epsilon$-reach set computation using a prototype implementation of the proposed algorithm has also been shown in the last section.

### 8. REFERENCES