Verification of Polyhedral-Invariant Hybrid Automata Using Polygonal Flow Pipe Approximations

Alongkrit Chutinan and Bruce H. Krogh

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-8390
ac4c@andrew.cmu.edu/krogh@ece.cmu.edu

Abstract. This paper presents a computational technique for verifying properties of hybrid systems with arbitrary continuous dynamics. The approach is based on the computation of approximating automata, which are finite-state approximations to the (possibly infinite-state) discrete-trace transition system for the hybrid system. The fundamental computation in the generation of approximating automata is the mapping of sets of continuous states to the boundaries of the location invariants. This mapping is computed by intersecting flow pipes, the sets of reachable states for continuous systems, with the invariant boundaries. Flow pipes are approximated by sequences of overlapping convex polygons. The paper presents an application of the computational procedure to a benchmark hybrid system, a batch evaporator.

1 Introduction

Hybrid system behaviors can be described by an infinite-state transition system [7]. A standard approach to verifying properties of a hybrid system is to find an equivalent transition system called a *bisimulation* with a finite number of states [10, 11]. There are two principle difficulties with this approach. First, a finite-state bisimulation exists only for certain classes of hybrid systems [6, 10, 11]. Since a finite-state bisimulation may not exist in general, one cannot guarantee a procedure for computing a bisimulation will terminate. The second problem is that any procedure for computing a bisimulation for a hybrid system requires the computation of *flow pipes*, that is, the collection of continuous-time trajectories emanating from a set of initial states [12]. Although flow pipes can be computed exactly in certain cases [1, 3, 10, 11], sets of continuous trajectories can only be approximated numerically for most continuous systems. Moreover, the errors in these approximations typically grow with simulation time, making them too conservative to obtain meaningful results.

This paper concerns an approach to verification that addresses both of these problems. The first problem is dealt with by verifying finite-state transition systems that are conservative approximations (i.e. simulations), rather than bisimulations, of the infinite-state transition system. Previous papers have presented algorithms for computing finite-state transition systems for hybrid systems, called approximating automata [2–4]. In this paper we observe that the algorithm for constructing and refining approximating automata can be viewed as a modification of the iterative procedure for computing a bisimulation. Although it cannot be guaranteed the verification question can be resolved using approximating automata, there are cases where verification can be accomplished with this approach even when a finite-state bisimulation does not exist [2–4].

The second problem is addressed by computing polygonal approximations to flow pipes for continuous dynamic systems. The procedure for computing flow pipe approximations from [4] is summarized briefly in this paper. An attractive feature of this procedure is that the approximation error does not grow with the simulation time. We show how the flow pipe approximations are used in the algorithm for computing approximating automata. The complete procedure is illustrated for the benchmark hybrid system problem from [8].

2 Transition Systems and Bisimulations

This section introduces the formal definitions and procedures used for verifying properties of hybrid systems. The fundamental abstract structure for representing the system dynamics is the transition system defined as follows.

Definition 1. A *transition system* T is a 3-tuple $T = (Q, \rightarrow, Q_0)$ where Q is the set *states*, $\rightarrow \subseteq Q \times Q$ is the set of *transitions*, and $Q_0 \subseteq Q$ is the set of *initial states*.

Given q and $q' \in Q$, the notation $q \rightarrow q'$ indicates that $(q, q') \in \rightarrow$. Sets of valid trajectories for a transition system are defined in the obvious way for sequences of states. Trajectories can be of finite length or infinite length.

Definition 2. (Pre/Postcondition Sets) Given a transition system $T = (Q, \rightarrow, Q_0)$, and a set $P \subseteq Q$, the *precondition* of P, denoted $Pre(P)$, is defined as $Pre(P) = \{q \in Q \mid \exists p \in P, q \rightarrow p\}$; the *postcondition* of P, denoted $Post(P)$, is defined as $Post(P) = \{q \in Q \mid \exists p \in P, p \rightarrow q\}$.

Definition 3. (Simulation) Let $T_1 = (Q_1, \rightarrow_1, Q_{01})$ and $T_2 = (Q_2, \rightarrow_2, Q_{02})$ be transition systems. A *simulation relation* of T_1 by T_2 is a binary relation $\preceq \subseteq Q_1 \times Q_2$ such that:

i. If $q_1 \preceq q_2$ and $q_1 \rightarrow_1 q_1'$, then there exists q_2' such that $q_2 \rightarrow_2 q_2'$ and $q_1' \preceq q_2'$;
ii. For each $q_1 \in Q_{01}$, there exists $q_2 \in Q_{02}$ such that $q_1 \preceq q_2$.

We say T_2 simulates T_1, denoted $T_1 \preceq T_2$, if there exists a simulation relation of T_1 by T_2. T_2 is also called a simulation of T_1.

Definition 4. (Bisimulation) Let $T_1 = (Q_1, \rightarrow_1, Q_{01})$ and $T_2 = (Q_2, \rightarrow_2, Q_{02})$ be transition systems. A *bisimulation relation* between T_1 and T_2 is a binary relation $\equiv \subseteq Q_1 \times Q_2$ such that $\equiv$ is a simulation relation of T_1 by T_2 and $\equiv^{-1} \subseteq Q_2 \times Q_1$ is a simulation relation of T_2 by T_1.

We say the transition systems T_1 and T_2 bisimulate each other, denoted $T_1 \equiv T_2$, if there exists a bisimulation relation between T_1 and T_2. An approach to finding a bisimulation of a transition system uses *quotient transition systems*, defined as follows.

Definition 5. (Quotient Transition System) [6, 10, 11] Given a transition system $T = (Q, \rightarrow, Q_0)$ and a partition $\mathcal{P}$ of Q, the *quotient transition system* of T is defined as $T/\mathcal{P} = (\mathcal{P}, \rightarrow_{\mathcal{P}}, Q_0/\mathcal{P})$, where for all $P, P' \in \mathcal{P}$, $P \rightarrow_{\mathcal{P}} P'$ iff there exist $q \in P$ and $q' \in P'$ such that $q \rightarrow q'$, or equivalently $Post(P) \cap P' \neq \emptyset$, and $Q_0/\mathcal{P} = \{P \in \mathcal{P} \mid P \cap Q_0 \neq \emptyset\}$.

It is easy to see that the relation $\preceq = \{(q, P) \in Q \times \mathcal{P} \mid q \in P\}$ is a simulation relation of T by $T/\mathcal{P}$. Therefore, $T \preceq T/\mathcal{P}$. The quotient system $T/\mathcal{P}$ is a bisimulation of T with the relation $\equiv = \{(q, P) \in Q \times \mathcal{P} \mid q \in P\}$ if and only if the partition $\mathcal{P}$ satisfies

$$\forall P, P' \in \mathcal{P}, \text{ either } P \cap Pre(P') = \emptyset \text{ or } P \cap Pre(P') = P. \tag{1}$$

In words, (1) states that all states in each $P \in \mathcal{P}$ behaves uniformly; given another $P' \in \mathcal{P}$, either all states or no state in P can reach some state in P' in one transition. Condition (1) leads to the following general procedure for computing a bisimulation of a transition system using quotient systems.

Bisimulation Procedure (BP) [6, 10, 11]:
set $\mathcal{P} = \mathcal{P}_0$
% check termination condition
while $\exists P, P' \in \mathcal{P}$ such that $\emptyset \neq P \cap Pre(P') \neq P$ {
    % partition refinement
    split P into $P_1 = P \cap Pre(P')$ and $P_2 = P \setminus Pre(P')$
    set $\mathcal{P} = (\mathcal{P} \setminus \{P\}) \cup \{P_1, P_2\}$
}

Note that (1) is precisely the termination condition of BP. In each iteration of BP, the partition refinement scheme uses the information obtained from $Pre(P')$ to split P into the part that can reach P' and the part that cannot. Also note that the $Pre(\cdot)$ operation in BP is with respect to the transition relation $\rightarrow$ of T, not the transition relation $\rightarrow_{\mathcal{P}}$ of the quotient system $T/\mathcal{P}$.

The termination condition serves to guarantee that the quotient system is a bisimulation of T when BP terminates. However, when transition systems are used to represent hybrid systems there is no guarantee the procedure will terminate. In the sequel we show that it may be possible to perform verification successfully using the quotient transition system obtained from a partition before the termination condition is satisfied. One can also use an approximation to the quotient transition system, rather than the exact quotient system, provided it is based on a conservative approximation to the *Post* operator. Moreover, we note that the partition refinement can be performed using heuristics if it is not convenient to compute the *Pre* operator as required in BP. This may affect the rate at which a useful quotient transition system is obtained, but it may be adequate since it is not necessary to converge to a bisimulation. These concepts are developed and illustrated in this paper. Further theoretical analysis of the use of simulations and approximations for verification and control is given in [9].

3 Polyhedral-Invariant Hybrid Automata

We consider a class of hybrid system called *polyhedral-invariant hybrid automata* (PIHA) defined as follows using the formalism from [10, 11] (with some restrictions).

Definition 6. A *polyhedral-invariant hybrid automaton* is a tuple $H = (X, X_0, F, E, I, G, R)$ where

- $X = X_C \times X_D$ where $X_C \subseteq R^n$ is the continuous state space and X_D is a finite set of discrete locations.
- F : a function that assigns to each discrete location $u \in X_D$ a vector field $f_u(\cdot)$ on X_C.
- $I : X_D \to 2^{X_C}$ assigns to $u \in X_D$ an *invariant* set of the form $I(u) \subseteq X_C$ where $I(u)$ is a nondegenerate convex polyhedron.
- $E \subseteq X_D \times X_D$ is a set of discrete transitions.
- $G : E \to 2^{X_C}$ assigns to $e = (u, u') \in E$ a guard set that is a union of faces of $I(u)$.
- $X_0 \subseteq X$ is the set of initial states of the form $X_0 = \bigcup_i (P_i, u_i)$ where each $P_i \subseteq I(u_i)$ is a polytope and $u_i \in U$. Here, the notation (P, u) means the set $\{(x, u) \in X \mid x \in P\}$.
- I,G, and E must satisfy the following *coverage* assumption.

$$\forall e = (u, u') \in E, G(e) \subseteq I(u').$$

Definition 7. Given an initial hybrid system state (x_0, u), the continuous trajectory in location u, denoted by $\zeta_{(x_0,u)}(\cdot)$, evolves according to the following conditions:

i. $\zeta_{(x_0,u)}(0) = x_0$; and
ii. $\dot{\zeta}_{(x_0,u)}(t) = f_u(\zeta_{(x_0,u)}(t)), \forall t \geq 0$ (until a discrete transition occurs).

PIHA are equivalent to switched continuous dynamic systems in which there are no jumps in the continuous state trajectory. The set of states through which a location is entered is called an *entry set* which is defined as follows.

Definition 8. Given a PIHA H, the set of *entry states* for the locations is defined as

$$X_{entry} = \{\ (x, u') \in X \mid \text{for some } (x, u) \in X \text{ and } (u, u') \in E, x \in G((u, u'))\}$$

We assume that at each $(x, u) \in X_{entry}$, $f_u(x)$ points into the interior of $I(u)$ (denoted $int(I(u))$). This means the continuous state trajectory always initially enters the interior of the invariant when the system makes a transition to the new location.

Transition systems provide an effective formalism for defining the semantics of a hybrid automaton. Henzinger [7] described the complete untimed behavior of a hybrid automaton using the *time-abstract transition system*. In this paper, we are interested in the behaviors of the PIHA H only at the times when discrete transitions occur. To abstract away the continuous dynamics and obtain the projection of the hybrid system behaviors onto the instants of discrete transitions, we define the *discrete-trace transition system* for H as follows.

Definition 9. Given a PIHA H, its *discrete-trace transition system* is given by $T_H = \{Q_H, \rightarrow_H, X_0\}$ where $Q_H = X_0 \cup X_{entry} \bigcup_{u \in X_D} \{q_u^{\perp}\}$ and the transition relation $\rightarrow_H$ is defined as

i. **Discrete Transitions.** $(x, u) \rightarrow_H (x', u')$ iff $u' \neq u$ and there exist $e = (u, u') \in E$ and $t_1 > 0$ such that $\zeta_{(x,u)}(t_1) = x'$, $x' \in G(e)$, and $\zeta_{(x,u)}(t) \in int(I(u))$ for all $t \in (0, t_1)$.
ii. **Null Transitions.** $(x, u) \rightarrow_H q_u^{\perp}$ iff $\zeta_{(x,u)}(t) \in I(u)$ for all $t \geq 0$.

The discrete transition comprises all the continuous-state trajectories in the hybrid system between location transitions. The null transition comprises all the continuous-state trajectories that remain in a location indefinitely. Although null transitions can occur in general, we will assume that all continuous-state trajectories in a PIHA eventually lead to a location transition. This is similar to the liveness assumption in [10].

4 Verification Using Approximations

In this section we introduce the verification problem and describe how it can be solved using an alternative to BP.

4.1 CTL Specifications

Temporal logic is a well-established formulation for specification of finite-state transition systems [5]. The specification problem is formulated by attaching a finite set of *atomic propositions* to a transition system whose individual values are either true or false for each state in the transition system. The transition graph of the transition system is unfolded into an infinite *computation tree*. A temporal logic called *computation tree logic* (CTL) can then be used to specify system evolutions in terms of the atomic propositions along some or all paths of the computation tree. It has been shown in [5] that a CTL formula for a transition system corresponds to a region in the state space of the transition system for which the CTL expression is true. Furthermore, the region corresponding to a CTL formula can be computed using fixed-point iterations. For a finite transition system, such fixed-point computations are guaranteed to terminate. The transition system satisfies the specification if all initial states are included in the region corresponding to the CTL specification.

The same concept for specification of finite-state transition systems can be extended to infinite-state transition systems (e.g. T_H). Atomic propositions can be assigned to each state of T_H and the computation tree for T_H (with an uncountable number of nodes) can be obtained by unfolding its transition graph. CTL specification can be interpreted as before in the case of finite-state transition systems. The problem here is that the fixed-point computation for the CTL formula will not terminate because the state space of the transition system is uncountable.

4.2 Verification Using Simulations (Approximating Automata)

In the bisimulation approach to hybrid system verification, BP is applied to find a partition of the state space that will give a finite-state bisimulation of the infinite-state transition system. Verification of properties for the bisimulation is then equivalent to verification of properties of the original transition system. The problem with this approach is that a finite bisimulation may not exist, meaning BP will not converge and the verification step cannot be performed. Even when a finite-state bisimulation exists, it may take a very long time for BP to converge to the solution.

We propose using finite-state simulations, called *approximating automata*, rather than bisimulations to verify properties of the original hybrid system. Since quotient systems are, in general, simulations of the underlying transition system, one could attempt to perform verification on the quotient system in any iteration of BP. If the property is verified, there would be no need to refine the quotient system further. If not, another refinement iteration can be executed and verification can be attempted on the new quotient transition system. Continuing this process, it is sometimes possible to conclude whether or not the hybrid system satisfies the desired property before a bisimulation is achieved [2–4]. The restriction here is that since simulations are conservative approximations, only *universal* properties can be verified. In the context of CTL, a universal property is a property that holds along all paths in the computational tree.

To slow down the state explosion resulting from the partition refinement, we refine only the states in the quotient system that are relevant to the CTL specification. In the process of a CTL verification, one obtains the set of initial states in the current quotient system that satisfy the CTL specification. Since the specification is universal, the verification result cannot be improved by refining these states and their descendants. Thus, one should only refine the initial states that do not satisfy the CTL specification and their descendants. Integrating this additional refinement criterion into the bisimulation procedure, we obtain the *Approximating Automata Procedure (AAP)* shown in figure 1. In AAP, TBR is the set of states "to be refined" in each iteration. It consists of the set $REFINE$, the states that should be refined from the bisimulation requirement, and the set $reach((\mathcal{P} \setminus SPEC) \cap X_0/\mathcal{P}_N)$, the states that should be refined from the CTL specification requirement. Choices of *bisimulation_termination_condition* and *refinement_method* are left unspecified since there are various alternatives for these steps, as discussed in section 2.

Approximating Automata Procedure (AAP):

```
initialize N = 0 and P_N = P_0
repeat for the partition P_N
    compute (or approximate) T_H/P_N (see section 5)
    compute SPEC = {P ∈ P_N | P satisfies CTL specification for T_H/P_N}
    if X_0/P_N ⊆ SPEC
        stop = 1 % specification is satisfied
    else
        compute REFINE = {P ∈ P_N |∃P' ∈ P_N
            violating bisimulation_termination_condition}
        if REFINE == ∅
            stop = 1 % bisimulation obtained and specification is false
        else
            compute TBR = reach((P_N \ SPEC) ∩ X_0/P_N) ∩ REFINE
            if TBR == ∅
                stop = 1 % no state worth refining and specification is false
            else
                stop = 0 % refine partition
                P_{N+1} = P_N
                for each P ∈ TBR
                    split P using refinement_method into P_1, P_2 such that
                        P_1 ∪ P_2 = P and P_1 ∩ P_2 = ∅
                    set P_{N+1} = (P_{N+1} \ {P}) ∪ {P_1, P_2}
                endfor
                N = N + 1
            endif
        endif
    endif
until (stop == 1)
```

Fig. 1. Verification procedure using simulations

5 Approximating the PIHA Quotient System

This section discusses a computational method for approximating a quotient system of the discrete-trace transition system T_H where H is a PIHA, the first step in AAP. Given a partition $\mathcal{P}$, we use the flow pipe approximation [4] to conservatively approximate postcondition sets which are essential in defining the transition relation $\to_{\mathcal{P}}$ of $T_H/\mathcal{P}$ (see definition 5). We assume that the partition of the state space for T_H is of the form

$$\mathcal{P} = \{(\pi, u) \in X_0 \cup X_{entry} \mid \pi \text{ is a convex polytope and } u \in X_D\}.$$

To approximate the postcondition set for each $(\pi, u) \in \mathcal{P}$, we start by computing an outer approximation to the set on the boundary of $I(u)$ that can be first reached from π under the flow equation $\dot{x} = f_u(x)$ starting from continuous states in π. We will refer to such set as the *forward mapping* of π for location u, denoted $FMAP_u(\pi)$. The procedure for computing $FMAP_u(\pi)$ is given in figure 2. The forward mapping procedure computes outer approximations to the flow pipe segments from π under the differential equation $\dot{x} = f_u(x)$, denoted $\hat{\mathcal{R}}^u_{[t,t+\Delta t]}(\pi)$, and takes the intersection of the flow pipe segment approximation with the boundary of $I(u)$ in each time step until $\hat{\mathcal{R}}^u_{[t,t+\Delta t]}(\pi)$ lies completely outside of $I(u)$. (Note that $\hat{\mathcal{R}}^u_{[t,t+\Delta t]}(\pi)$ is guaranteed to eventually leave the location under the assumption there are no null events for PIHA.)

Forward Mapping Procedure:

```
initialize t = 0, FMAP_u(π) = ∅, and stop = 0
repeat
    compute flow pipe segment R̂^u_[t,t+Δt](π)
    if (R̂^u_[t,t+Δt](π) ∩ I(u) == ∅)
        stop = 1
    else
        FMAP_u(π) = FMAP_u(π) ∪ (R̂^u_[t,t+Δt](π) ∩ δI(u))
        t = t + Δt
    endif
until (stop == 1)
```

Fig. 2. Procedure for computing forward mapping set given (π, u)

The approximation to the flow pipe segment for each time step $[t, t + \Delta t]$ is computed as follows. First, a set of *enclosing* normal vectors, c_i, is chosen. By enclosing, we mean that the half-space intersection $\bigcap_i\{x|c_i^T x \leq d_i\}$ forms a closed polyhedron for some constants d_i. The flow pipe segment is computed by solving, for each normal vector c_i, the optimization problem

$$\max_{x_0,t} \; c_i^T \zeta_{(x_0,u)}(t)$$
$$\text{s.t.} \quad x_0 \in \pi, t \in [t, t + \Delta t] \tag{2}$$

Let d_i^* be the solution to (2) for c_i. The flow pipe segment is given by $\hat{\mathcal{R}}^u_{[t,t+\Delta t]}(\pi) = \bigcap_i\{x|c_i^T x \leq d_i^*\}$.

To find the set of normal vectors for the optimization problems, we use the heuristics depicted in figure 3a. The trajectories starting from the vertices of π are simulated using an ODE solver to the two time points t and $t + \Delta t$. The normal vectors are then taken from the normal vectors on the faces of the convex hull of these points. After obtaining the normal vectors, we solve (2) for each c_i to obtain the flow pipe segment as depicted in figure 3b. For more details on the flow pipe approximation, see [4].

Using the forward mapping procedure, the overall procedure for approximating the quotient system $T_H/\mathcal{P}$ is given in figure 4. The procedure constructs

(a) Initial directions from convex hull

(b) Solving optimization problems for flow pipe segment

Fig. 3.

the transition relation $\rightarrow_{\mathcal{P}}$ directly from the forward mapping sets. Alternatively, the postcondition sets can be used to define the quotient system as in definition 5. The postcondition sets can be obtained from the mapping sets by $Post((\pi, u)) = \bigcup_{u'\, s.t.\, (u,u') \in E}(FMAP_u(\pi) \cap I(u'), u')$.

Quotient System Approximation Procedure:
% compute forward mapping sets
for each $(\pi, u) \in \mathcal{P}$
  compute $FMAP_u(\pi)$
endfor
% define transition relation
for each $(\pi, u) \in \mathcal{P}$
  for each $(\pi', u') \in \mathcal{P}, (\pi', u') \neq (\pi, u)$
    define $(\pi, u) \rightarrow_{\mathcal{P}} (\pi', u')$ if $FMAP_u(\pi) \cap \pi' \neq \emptyset$ and $(u, u') \in E$
  endfor
endfor

Fig. 4. Procedure for approximating the quotient system $T_H/\mathcal{P}$

6 Application: Verification of a Batch Evaporator

The example hybrid system considered in this paper is a batch evaporator taken from Kowalewski and Stursberg [8]. The evaporation process diagram is shown in figure 5. The process follows the following production sequence. First, tank T_1 is filled with a solution which is evaporated until a desired concentration is reached. Tank T_1 is then drained as soon as tank T_2 is emptied from the previous batch. For safety reasons, the heating is shut off when the alarm temperature, T_{alarm}, is reached. When the temperature in tank T_1 falls below a certain temperature, T_{crys}, crystallization will occur and spoil the batch. Our objective is to verify that the alarm temperature is chosen appropriately such that from a given set of initial conditions the temperature in tank T_1 never falls below the crystallization temperature before T_1 is completely drained.

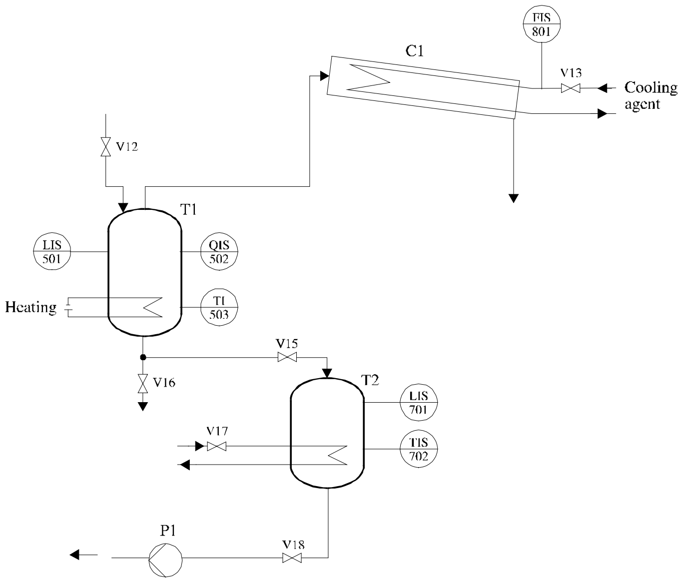


Fig. 5. Batch evaporation diagram

The control inputs to the system are the status of the heater (on/off) and the valve positions V_{15} and V_{18} (open/closed). Three configurations of the inputs are currently employed. We specify an input configuration by a discrete variable u. The three configurations u_1, u_2, and u_3, are tabulated below.

Configuration	**Heating**	$\mathbf{V_{15}}$	$\mathbf{V_{18}}$	**Description**
u_1	on	closed	open	heating T_1
u_2	off	closed	open	cooling T_1/drain T_2
u_3	off	open	closed	cooling/drain T_1

The continuous state variables are the liquid level in T_1, the liquid level in T_2, and the temperature in T_1, denoted by H_1, H_2, and T, respectively. The continuous dynamics depends on the input configuration. For configuration u_1, the state equations are

$$\begin{aligned}
\dot{H}_1 &= 0 \\
\dot{H}_2 &= -3.333 \cdot 10^{-4}\sqrt{19.62 H_2} \\
\dot{T} &= \frac{5000 - 24(T - 283)}{1.23 \cdot 10^5 H_1 - 1.327 \cdot 10^9 T^{-2} + 2.819 \cdot 10^6 T^{-1} + 6.433 \cdot 10^3 - 10.513 T}.
\end{aligned}$$

For u_2, the state equations are

$$\dot{H}_1 = 0$$
$$\dot{H}_2 = -3.333 \cdot 10^{-4}\sqrt{19.62H_2}$$
$$\dot{T} = \frac{-24(T-283)}{1.23 \cdot 10^5 H_1 + step(T-373)\left(-1.327 \cdot 10^9 T^{-2} + 2.819 \cdot 10^6 T^{-1} + 6.433 \cdot 10^3 - 10.513T\right)},$$

where $step(\cdot)$ denotes the standard step function. Finally, the state equations for u_3 are

$$\begin{aligned}\dot{H}_1 &= -6.667 \cdot 10^{-4}\sqrt{19.62H_1}\\ \dot{H}_2 &= 3.333 \cdot 10^{-4}\sqrt{19.62H_1}\\ \dot{T} &= \frac{-0.036(T-283)(0.03+0.628H_1)}{29.1H_1}.\end{aligned}$$

The production sequence described previously can be translated into the PIHA shown in figure 6. Locations u_1, u_2, and u_3 correspond directly to the input configurations u_1, u_2, u_3. Locations u_4 and u_5 are empty locations with trivial continuous dynamics. They are used to indicate the *failure* and *success* of the production sequence, respectively. Locations u_1, u_2, and u_3 have the same invariant given by the rectangular box,

$$\{H_{1min} \le H_1 \le H_{1max} \wedge H_{2min} \le H_2 \le H_{2max} \wedge T_{crys} \le T \le T_{alarm}\},$$

which represents the physical limits and thresholds on the continuous state variables.

The system starts with location u_1 and initial conditions $H_1 \in [0.2, 0.22]$ m, $H_2 \in [0.28, 0.3]$ m, and $T = 373K$. Since the liquid level in each tank in the ODE model can only reach zero asymptotically, we approximate the event that a tank is empty by a small threshold $H_{imin}, i = 1,2$. The numerical values for the limits on the rectangular invariant box for all locations are

$$\begin{aligned} &H_{imin} = 0.04 \text{ m}, \quad H_{imax} = 0.4 \text{ m}, i = 1,2,\\ &T_{crys} = 338 \text{ K}, \text{ and } T_{alarm} = 395 \text{ K}.\end{aligned}$$

Our verification problem can be restated as

> Verify that all trajectories from the initial continuous state set $X_0^C = \{0.2 \le H_1 \le 0.22, 0.28 \le H_2 \le 0.3, T = 373\}$ in location u_1 eventually reach location u_5.

which translates directly into the CTL expression $AF(u = u_5)$.

The verification was performed using AAP with the quotient system $T_H/\mathcal{P}$ approximated as described in section 5. The continuous part of the entry set for each location is the face(s) of the invariant box that is(are) the guard(s) on all its incoming transitions as shown in figure 6. The heuristics based on the vector field variation that is used to obtain the initial partition for each face is described briefly as follows.

Taking the whole invariant face as the starting point for the partition, we estimate the vector field variation on the face by taking the samples of the

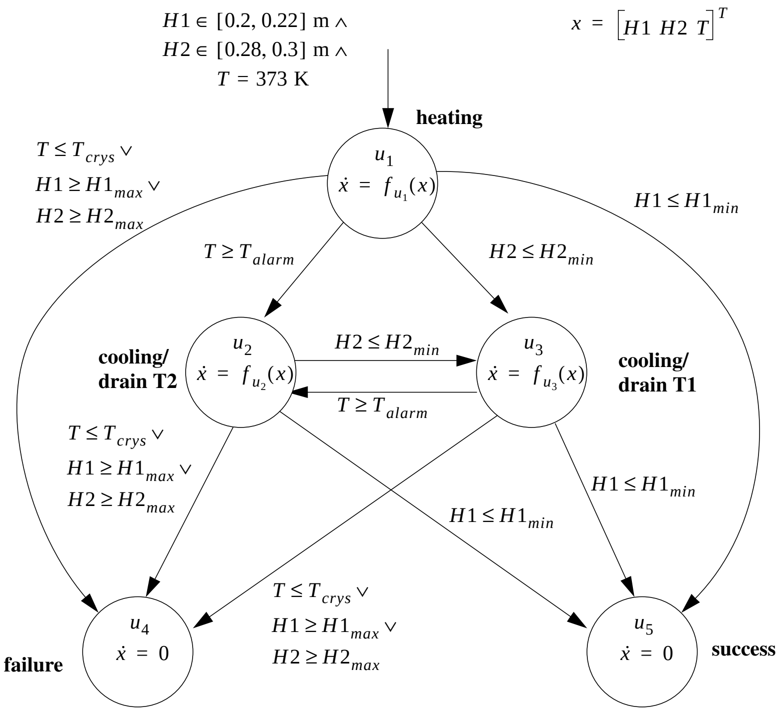


Fig. 6. PIHA for the batch evaporator

quantity $c^T f_u(x_i)$, where c is the normal vector to the face and u is the parent location, from points x_i on the face. We take the difference between the maximum and minimum samples as the variation estimate. If the variation is greater than the tolerance, the polytope is divided into two subsets by a hyperplane that pass roughly through the middle of the polytope. Each polytope is divided recursively until all polytopes in the partition satisfy the vector field variation tolerance.

The initial entry set partition $\mathcal{P}_0$ is shown in figure 8. Since each face of the invariant box can be identified with a unique location, the location associated with each face is not shown in the figure. Each continuous subset in the entry set partition will be referred to as a patch. The initial continuous state set X_0^C is not partitioned further because it is already a small set.

Using the forward mappings computed for the partition $\mathcal{P}_0$, we compute the transition relation for the quotient system $T_H/\mathcal{P}_0$ as described in section 5. The forward mapping is illustrated in Fig. 7.

The transition relation for the initial quotient system $T_H/\mathcal{P}_0$ is shown in figure 8. By our convention, the initial state is always the last state in the transition table. For example, patch 21 is the initial state representing (X_0^C, u_1) in figure 8. Some patches are identified as "indeterminate." These are patches containing singularity points for the vector field and the mapping cannot be computed for them. However, as we shall see, these patches finally become unreachable from

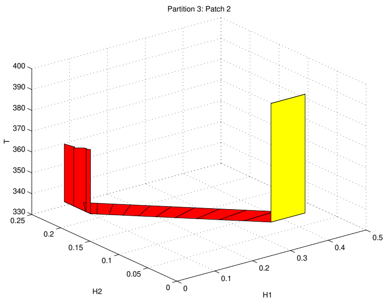


Fig. 7. Forward mapping approximation: the light-shaded patch is the initial set and its forward mapping to the boundary of the invariant box is composed of dark-shaded patches.

the initial set as the refinement proceeds with each iteration of the verification procedure. Thus, they do not affect our verification result.

The TBR set for the first iteration of the verification procedure is also shown in figure 8. Although patch 21 should be in the TBR set, we choose not to refine the initial state at all in this example to keep the presentation simple as there will only one initial state in each quotient system. Patches in the TBR set are simply split in half to form the next partition $\mathcal{P}_1$. We do not use the refinement method in BP because it is costly and difficult to implement with flow pipe approximations and convex polyhedral representation of sets. The quotient system $T_H/\mathcal{P}_1$ is computed with the new partition and the process continues.

After three iterations, we find that the TBR set is empty. This is because the initial state (patch 40 in figure 9) already satisfies the specification. Since $T_H/\mathcal{P}_3$ is a simulation of T_H, every trajectory of T_H is contained in $T_H/\mathcal{P}_3$ and we conclude that T_H for the batch evaporator also satisfies the specification.

7 Discussion

This paper presents a computational method for verifying properties of polyhedral-invariant hybrid automata (PIHA) with arbitrary continuous dynamics in the locations. The verification procedure is based on approximating automata computed as the quotient transition systems from partitions of the infinite state space for the discrete-trace transition system. The continuous-state flow pipes are approximated using sequences of convex polyhedra. Important research issues to be explored include: extending the flow pipe approximation technique to handle systems with uncertain dynamics, such as differential inclusions; numerical methods to guarantee outer approximations using floating-point arithmetic; and incorporating methods for identifying null transitions, that is, identifying when there are trajectories that do not leave invariants for a location. The computational procedures described in this paper are currently being incorporated into a Matlab/Simulink tool for specifying and analyzing PIHA.

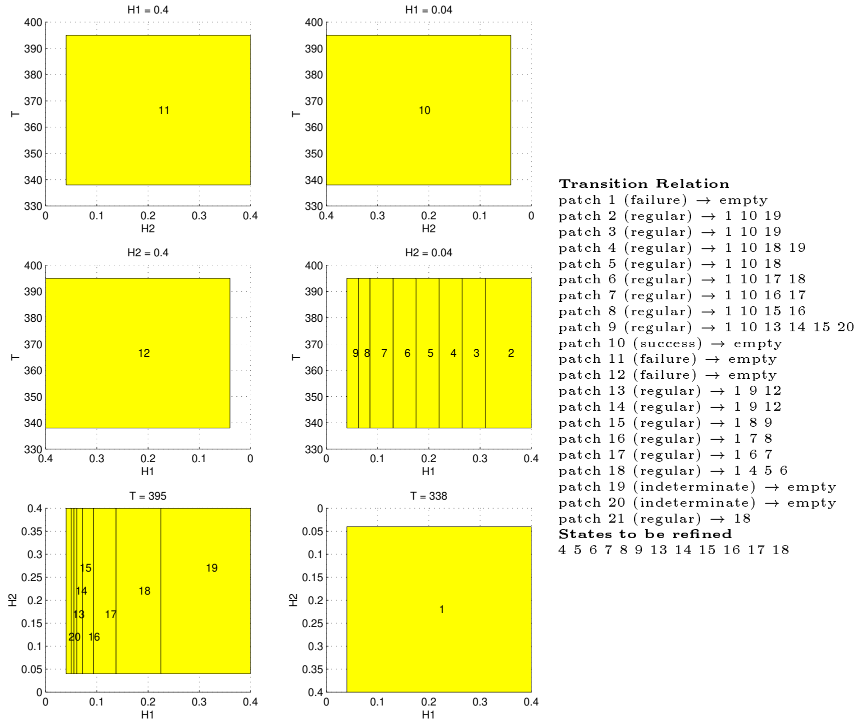
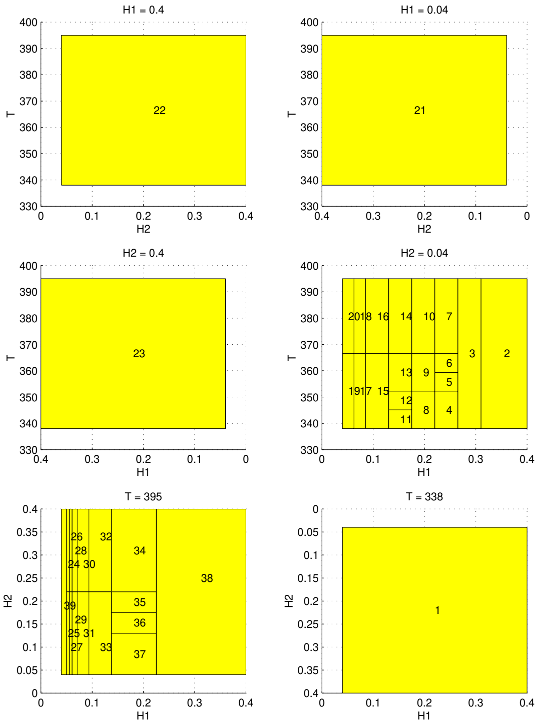


Fig. 8. Quotient System $T/\mathcal{P}_0$

References

1. R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. on Software Engineering*, 22(3):181–201, Mar 1996.
2. A. Chutinan and B.H. Krogh. Computing approximating automata for a class of hybrid systems. *submitted to Mathematical Modeling of Systems: Special Issue on Discrete Event Models of Continuous Systems*, 1997.
3. A. Chutinan and B.H. Krogh. Computing approximating automata for a class of linear hybrid systems. In *Hybrid Systems V*, Lecture Notes in Computer Science. Springer-Verlag, 1998.
4. A. Chutinan and B.H. Krogh. Computing polyhedral approximations to dynamic flow pipes. *submitted to* 37^{th} *IEEE Conference on Decision and Control: Invited Session on Synthesis and Verification of Controllers for Hybrid Systems*, 1998.
5. E. Clarke, O. Grumberg, and D. Long. Verification tools for finite-state concurrent systems. In *Proceedings of A Decade of Concurrency: Reflections and Perspectives*, pages 124–75, REX School/Symposium, Noordwijkerhout, The Netherlands, 1-4 June 1993. Springer-Verlag, Berlin, Germany, 1994.
6. T.A. Henzinger. Hybrid automata with finite bimulations. In Z. Fülöp and F. Gécseg, editors, *ICALP 95: Automata, Languages, and Programming*, pages 324–335. Springer-Verlag, 1995.



Transition Relation
patch 1 (failure) → empty
patch 2 (regular) → 1 21 38
patch 3 (regular) → 1 21 38
patch 4 (regular) → 1 21
patch 5 (regular) → 1 21
patch 6 (regular) → 21
patch 7 (regular) → 21
patch 8 (regular) → 1 21
patch 9 (regular) → 21
patch 10 (regular) → 21
patch 11 (regular) → 1
patch 12 (regular) → 1 21
patch 13 (regular) → 21
patch 14 (regular) → 21
patch 15 (regular) → 1 21
patch 16 (regular) → 21 31 33
patch 17 (regular) → 1 21
patch 18 (regular) → 21 29 31
patch 19 (regular) → 1 21
patch 20 (regular) → 21
patch 21 (success) → empty
patch 22 (failure) → empty
patch 23 (failure) → empty
patch 24 (regular) → 1 23
patch 25 (regular) → 1 19 20
patch 26 (regular) → 1 23
patch 27 (regular) → 1 19 20
patch 28 (regular) → 1
patch 29 (regular) → 1 17 18 19 20
patch 30 (regular) → 1
patch 31 (regular) → 1 15 16 17 18
patch 32 (regular) → 1 11 12 15
patch 33 (regular) → 1 12 13 14 15 16
patch 34 (regular) → 1 4 5 6 8 9 11 12 13
patch 35 (regular) → 6 7 9 10 12 13
patch 36 (regular) → 7 9 10 13 14
patch 37 (regular) → 7 10 13 14
patch 38 (indeterminate) → empty
patch 39 (indeterminate) → empty
patch 40 (regular) → 36 37
States to be refined:
none
States that eventually reach "success" patch (21):
6 7 9 10 13 14 20 21 36 37 40

Fig. 9. Quotient System $T/\mathcal{P}_3$

7. T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996. Invited tutorial.
8. S. Kowalewski and O. Stursberg. The batch evaporator: A benchmark example for safety analysis of processing systems under logic control. *submitted to: 4th Int. Workshop on Discrete Event Systems (WODES '98), Cagliari (Italy)*, August 1998.
9. B.H. Krogh and A. Chutinan. Hybrid systems: Modeling and control. In P.M. Frank, editor, *Advances in Control*. Springer-Verlag, 1999. To appear.
10. G. Lafferriere, G. J. Pappas, and S. Sastry. Hybrid systems with finite bisimulations. Technical Report UCB/ERL M98/15, University of California at Berkeley, April 1998.
11. G. Lafferriere, G. J. Pappas, and S. Yovine. Decidable hybrid systems. Technical Report UCB/ERL M98/39, University of California at Berkeley, June 1998.
12. Feng Zhao. *Automatic Analysis and Synthesis of Controllers for Dynamical Systems Based on Phase-Space Knowledge*. PhD thesis, MIT Artificial Intelligence Laboratory, 1992.