Deciding Concurrent Planar Monotonic Linear Hybrid Systems

Pavithra Prabhakar[1], Nima Roohi[2](✉), and Mahesh Viswanathan[2]

[1] IMDEA Software Institute, Madrid, Spain
pavithra.prabhakar@imdea.org

[2] Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, USA
{roohi2,vmahesh}@illinois.edu

Abstract. Decidability results for hybrid automata often exploit subtle properties about dimensionality (number of continuous variables), and interaction between discrete transitions and continuous trajectories of variables. Thus, the decidability results often do not carry over to parallel compositions of hybrid automata, even when there is no communication other than the implicit synchronization of time. In this paper, we show that the reachability problem for concurrently running planar, monotonic hybrid automata is decidable. Planar, monotonic hybrid automata are a special class of linear hybrid automata with only two variables, whose flows in all states are simultaneously monotonic along some direction in the plane. The reachability problem is known to be decidable for this class. Our concurrently running automata synchronize with respect to a global clock, and through labeled discrete transitions. The proof of decidability hinges on a new observation that the timed trace language of a planar monotonic automaton can be recognized by a timed automaton, and the decidability result follows from the decidability of composition of timed automata. Our results identify a new decidable subclass of multi-rate hybrid automata.

1 Introduction

Hybrid automata [6] model the interaction between a discrete controller and a physical environment. Such automata have finitely many control locations, modeling the state of the discrete controller, and real-valued variables that evolve continuously with time, modeling the state of the physical environment. They exhibit hybrid behavior where control location changes influence the values of the real-valued variables and the physical laws governing their evolution. The safety verification of such systems can often be reduced to the reachability problem, wherein one asks if a certain state/set of states can be reached during an execution that starts from some initial state.

The reachability question for hybrid automata is in general undecidable [7], and special classes of decidable hybrid automata have been identified [2–4,7,8,11,13]. Small variations to the decidable classes are known to make the reachability problem undecidable [1,3,7,10,13]. The reason for this is because the decidability results exploit subtle properties about dimensionality (number of continuous variables in the hybrid automaton), and the interaction between the discrete transitions and the continuous dynamics of the variables. One consequence of this is that the reachability problem is often undecidable for parallel compositions of decidable automata. This is true even when the concurrently running automata do not communicate other than the implicit synchronization of time [5].

In this paper, we show that the reachability problem is decidable for parallel compositions of *planar monotonic linear hybrid systems*. Planar monotonic linear hybrid systems are hybrid systems with the following restrictions — the automaton has only two real-valued variables; the flows (continuous dynamics with time) of the variables are given by linear functions, and all the constraints describing the invariants, guards, and resets are linear functions; the flow in every control location is monotonic in some direction on the plane and the variables are reset on discrete transitions that either change this monotonic direction, or are labeled through actions that are used for communication between automata. The reachability problem for the subclass of the automata with no resets is known to be decidable [11].

Our proof proceeds as follows. For a planar monotonic linear hybrid system $\mathcal{H}$, let us consider $\texttt{TReach}_{\mathcal{H}}$ to be the set of triples (x, t, y) such that there is an execution starting at x that reaches y at time t, and the variables are never reset along this execution. Our first observation is that $\texttt{TReach}_{\mathcal{H}}$ is expressible in the first order theory of linear arithmetic. This requires us to adapt the decidability proof for reachability presented in [11] to consider timed reachability and representation in the theory of reals; the challenge in proving such an expressivity result is that executions of such automata can have an unbounded number of discrete transitions. One consequence of this is that the set of times at which one can reach a polyhedron P_2 when starting from a polyhedron P_1 can also be expressed in the theory of linear arithmetic. Since this theory is o-minimal [12], this means that this set of times can be expressed as a finite union of intervals. We use this conclusion to argue that the timed trace language of $\mathcal{H}$ (the sequence of visible synchronizable actions along with the time when they happen) can be generated by a timed automaton. Hence the parallel composition of planar monotonic linear hybrid automata is timed trace language equivalent to the parallel composition of timed automata, which can be effectively constructed from the component planar monotonic linear hybrid automata. Since the reachability question can be reduced to the emptiness of the timed trace language, our decidability result follows from the fact that the emptiness of the timed trace language of timed automata can be decided [2].

Our proof is inspired by ideas presented in [9] where a new decidability proof for initialized rectangular hybrid automata is obtained by viewing such an automaton as the parallel composition of 1-dimensional systems. Our proof and the results in [9] suggest a method to lift decidability results for low dimensional systems to higher dimensional systems. For example, one consequence of our results is the identification of new decidable subclass of multi-rate automata that we call *phased multi-rate automata*; reachability for the general class of multi-rate automata is known to be undecidable [7]. Our result is proved by demonstrating that phased multi-rate automata can be seen as the parallel composition of planar monotonic linear hybrid systems.

2 Preliminaries

Notations. $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{R}_{\geq 0}$ are, respectively, the set of *rational*, *real*, and *non-negative real* numbers. Given $\mathbf{u} \in \mathbb{R}^n$, we use $\mathbf{u}_i$ to denote the i-th component of $\mathbf{u}$. Given $\mathbf{u} \in \mathbb{R}^n$ and $\mathbf{v} \in \mathbb{R}^m$, we use $\mathbf{u} \circ \mathbf{v}$ to denote the *concatenation* of $\mathbf{u}$ with $\mathbf{v}$ in $\mathbb{R}^{n+m}$. Given two sets $U \subseteq \mathbb{R}^n$ and $V \subseteq \mathbb{R}^m$, $U \circ V = \{\mathbf{u} \circ \mathbf{v} \mid \mathbf{u} \in U, \mathbf{v} \in V\}$. For any two functions $f_1 \in [A_1 \to \mathbb{R}^n]$ and $f_2 \in [A_2 \to \mathbb{R}^m]$, their Cartesian product $f_1 \times f_2$ maps (a_1, a_2) to $f_1(a_1) \circ f_2(a_2)$. For any two functions $f_1 \in [A_1 \to 2^{\mathbb{R}^n}]$ and $f_2 \in [A_2 \to 2^{\mathbb{R}^m}]$, $f_1 \otimes f_2$ maps (a_1, a_2) to $f_1(a_1) \circ f_2(a_2)$. Given a function $f \in [A \to B]$ and $A' \subseteq A$, we denote the restriction of f to A' by $f[A']$ and the image of A' under f, that is, $\{f(a) \mid a \in A'\}$, by $f(A')$. Finally, id_A denotes the identity function on A.

Polyhedral Sets. Let *Poly*$(\mathbb{R}^n)$, *OpenPoly*$(\mathbb{R}^n)$ and *BOpenPoly*$(\mathbb{R}^n)$, respectively, denote the set of all polyhedral sets, open polyhedral sets and bounded open polyhedral sets in $\mathbb{R}^n$. We refer to $\mathbb{R}^2$ as the *plane*, and any polyhedral set in $\mathbb{R}^2$ is called a *planar* polyhedral set. Recall that a polyhedral set P is the set of all points $\mathbf{x}$ satisfying a finite set of linear constraints $\{\mathbf{u}_1 \cdot \mathbf{x} + b_1 \lhd_1 0, \ldots, \mathbf{u}_k \cdot \mathbf{x} + b_k \lhd_k 0\}$. where $\lhd_i \in \{<, \leq\}$, $\mathbf{u}_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$. When the polyhedral set P is planar, we denote the lines corresponding to the constraints $\mathbf{u}_i \cdot \mathbf{x} + b_i = 0$ as *Lines*(P).

An n-dimensional *rectangular set* is a polyhedral set which can be expressed as the cross product of intervals $I_1 \times \ldots \times I_n$. The set of all rectangular sets and open rectangular sets of dimension n are denoted as *Rect*$(\mathbb{R}^n)$ and *OpenRect*$(\mathbb{R}^n)$, respectively. In addition, *PBOpenRect*$(\mathbb{R}^n)$ is the set of all partially bounded rectangular sets, that is, rectangular sets $I_1 \times \ldots \times I_n$, where for every $i \in \{1, \ldots n\}$, either $I_i = (-\infty, \infty)$ or I_i is bounded.

3 Linear Hybrid Systems

Hybrid automata [6] are a popular formalism for modeling systems with mixed discrete continuous behaviors. The discrete behavior is captured by a finite state automaton, and the continuous behavior is captured by a finite set of continuously evolving variables.

Definition 1. *A* linear hybrid system *or a* linear hybrid automaton $\mathcal{H}$*, is a tuple* (*Loc*, *dim*, *Act*, *Inv*, *Flow*, *Edge*, *Init*, *Final*)*, where*
- *Loc is a finite non-empty set of (discrete)* locations*.*
- *dim is the* dimension *of the hybrid system* $\mathcal{H}$ *and represents the number of continuous variables of the system.* $\mathbb{R}^{dim}$ *is referred to as the continuous state-space.*

- *Act is a finite non-empty set of* actions *which does not contain a special symbol* τ *or the elements of* $\mathbb{R}_{\geq 0}$.
- $\mathit{Inv} \in \big[\mathit{Loc} \to \mathit{Poly}\big(\mathbb{R}^{dim}\big)\big]$ *maps each location* q *to a polyhedral set as the* invariant *of* q.
- $\mathit{Flow} \in \big[\mathit{Loc} \to \mathbb{Q}^{dim}\big]$ *maps each location* q *to a vector as the* flow *in* q.
- $\mathit{Edge} \subseteq \mathit{Loc} \times \mathit{Loc} \times (\mathit{Act} \cup \{\tau\}) \times \mathit{Poly}\big(\mathbb{R}^{2dim}\big)$ *is a finite set of* edges. *Each* $e \in \mathit{Edge}$ *is a tuple* (q_1, q_2, l, r) *where*
  - $q_1 \in \mathit{Loc}$ *is the* source *of* e.
  - $q_2 \in \mathit{Loc}$ *is the* destination *of* e.
  - $l \in \mathit{Act} \cup \{\tau\}$ *is the* label *of* e.
  - $r \in \mathit{Poly}\big(\mathbb{R}^{2dim}\big)$ *is the* reset *of* e *and captures a binary relation on the continuous state-space.*

  An edge labelled by an element of Act is called a visible *edge, and that labelled by* τ *is called an* invisible *or a* silent *edge. We denote the different elements of* e *by* $\mathit{Src}(e)$, $\mathit{Dest}(e)$, $\mathit{Lab}(e)$, *and* $\mathit{Reset}(e)$, *respectively.*
- $\mathit{Init} \in \big[\mathit{Loc} \to \mathit{Poly}\big(\mathbb{R}^{dim}\big)\big]$ *maps each location* q *to a polyhedral set representing the* initial continuous states *in* q.
- $\mathit{Final} \in \big[\mathit{Loc} \to \mathit{Poly}\big(\mathbb{R}^{dim}\big)\big]$ *maps each location* q *to a polyhedral set representing the* final continuous states *in* q.

We require that for all $q \in \mathit{Loc}$, $\mathit{Init}(q) \subseteq \mathit{Inv}(q)$ *and* $\mathit{Final}(q) \subseteq \mathit{Inv}(q)$.

We denote the different elements of $\mathcal{H}_\eta$ respectively by Loc_η, dim_η, Act_η, Inv_η, Flow_η, Edge_η, Init_η and Final_η. From now on, a hybrid system will refer to a linear hybrid system. Finally, we require that all constants used in $\mathcal{H}$ are rationals.

3.1 Semantics

An execution of a hybrid system $\mathcal{H}$ starts in an initial state $(p, \mathbf{u})$, where $\mathbf{u} \in \mathit{Init}(p)$, and evolves through a sequence of continuous and discrete transitions. In any state $(q, \mathbf{v})$, a continuous transition corresponds to a continuous evolution of $\mathbf{v}$ using the rate $\mathit{Flow}(q)$ while remaining within the invariant $\mathit{Inv}(q)$; the location q does not change. On the other hand, a discrete transition from a state $(q, \mathbf{v})$ to a state $(q', \mathbf{v}')$ labelled by l is allowed if there is an edge (q, q', l, r) such that $(\mathbf{v}, \mathbf{v}') \in r$. Formally, the semantics of a linear hybrid system $\mathcal{H}$ is defined by the transition system $[\![\mathcal{H}]\!] = (S, S^{in}, S^{fin}, \rightarrow)$, where $S = \mathit{Loc} \times \mathbb{R}^n$ is the set of states, $S^{in} = \{(q, \mathbf{v}) \,|\, q \in \mathit{Loc}, \mathbf{v} \in \mathit{Init}(q)\}$ is the set of initial states, and $S^{fin} = \{(q, \mathbf{v}) \,|\, q \in \mathit{Loc}, \mathbf{v} \in \mathit{Final}(q)\}$ is the set of final states (note that $S^{in}, S^{fin} \subseteq S$). Finally, $\rightarrow \subseteq S \times (\mathit{Act} \cup \{\tau\} \cup \mathbb{R}_{\geq 0}) \times S$ is the union of discrete and continuous transitions that are defined as follows (we use $s \xrightarrow{l} s'$ to denote $(s, l, s') \in \rightarrow$):

- Discrete: $(q, \mathbf{v}) \xrightarrow{l} (q', \mathbf{v}')$ if $\mathbf{v} \in \mathit{Inv}(q)$, $\mathbf{v}' \in \mathit{Inv}(q')$, and there exists $e \in \mathit{Edge}$ such that $q = \mathit{Src}(e)$, $q' = \mathit{Dest}(e)$, $l = \mathit{Lab}(e)$, and $(\mathbf{v}, \mathbf{v}') \in \mathit{Reset}(e)$.
- Continuous: $(q, \mathbf{v}) \xrightarrow{t} (q', \mathbf{v}')$ if $q = q'$, $t \in \mathbb{R}_{\geq 0}$, $\mathbf{v}' = \mathbf{v} + t \cdot \mathit{Flow}(q)$ and $\forall t' \in [0, t] \bullet \mathbf{v} + t' \cdot \mathit{Flow}(q) \in \mathit{Inv}(q)$.

Path, Execution and Execution Fragment. A *path* of the hybrid system $\mathcal{H}$ is a sequence $\pi = e_1, e_2, \ldots, e_k$ such that for every j, $Dest(e_j) = Src(e_{j+1})$. An *execution fragment* of $\mathcal{H}$ is a sequence $\sigma = (q_0, \mathbf{v}_0)a_0(q_1, \mathbf{v}_1)a_1 \ldots (q_{m-1}, \mathbf{v}_{m-1})$ $a_{m-1}(q_m, \mathbf{v}_m)$, such that $(q_i, \mathbf{v}_i) \xrightarrow{a_i} (q_{i+1}, \mathbf{v}_{i+1})$, for $0 \leq i < m$. We call m the length of σ, and is denoted $|\sigma|$. Let $\sigma^s(i)$ denote the i-th state, namely, $(q_i, \mathbf{v}_i)$, and $\sigma^a(i)$ denote the i-th label, namely, a_i. Let $\sigma[i, j]$ denote the sequence from $(q_i, \mathbf{v}_i)$ to $(q_j, \mathbf{v}_j)$, namely, $(q_i, \mathbf{v}_i)a_i(q_{i+1}, \mathbf{v}_{i+1}) \ldots (q_{j-1}, \mathbf{v}_{j-1})a_{j-1}(q_j, \mathbf{v}_j)$. The *duration* of σ, denoted $Duration(\sigma)$, is given by $\sum_{0 \leq i < |\sigma|, \sigma^a(i) \in \mathbb{R}_{\geq 0}} \sigma^a(i)$. It taken to be 0 when the summation is over an empty set. We call the execution fragment σ an *execution* of $\mathcal{H}$ if $\sigma^s(0)$ is an initial state and $\sigma^s(|\sigma|)$ is a final state. Let $Exec(\mathcal{H})$ denote the set of all executions of $\mathcal{H}$.

Reachability Problem. The reachability problem asks, given a linear hybrid system $\mathcal{H}$, is $Exec(\mathcal{H})$ non-empty, i.e., is there an execution fragment of $\mathcal{H}$ from an initial state to a final state.

Timed Trace. A timed trace corresponding to an execution fragment is an alternating sequence of times (between consecutive visible transitions) and visible actions. First, we define a splitting of an execution. A *splitting* of an execution σ is a finite sequence of execution fragments $\sigma_0 \circ \sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_k$ such that there exists a sequence of indices $0 \leq i_0 \leq i_1 \leq i_2 \leq \ldots \leq i_k = |\sigma|$, where $\sigma_0 = \sigma[0, i_0]$, and for all $1 \leq j \leq k$, $\sigma_j = \sigma[i_{j-1}, i_j]$. A *visible splitting* of σ is a splitting $\sigma = \sigma_0 \circ \sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_k$ such that for all $0 \leq i < |\sigma_0|$, $\sigma_0^a(i) \notin Act$ and for all $1 \leq j \leq k$, $\sigma_j^a(i) \in Act$ if and only if $i = 0$. A *timed trace* of σ, denoted $TimedTrace(\sigma)$, is the sequence $t_0 a_1 t_1 \ldots a_k t_k$ in $(\mathbb{R}_{\geq 0} \cdot Act)^* \cdot \mathbb{R}_{\geq 0}$, such that $\sigma = \sigma_0 \circ \sigma_1 \circ \ldots \circ \sigma_k$ is a visible splitting of σ, for $0 \leq i \leq k$, $t_i = Duration(\sigma_i)$, and for $1 \leq i \leq k$, $a_i = \sigma_i^a(0)$ is the unique visible action in σ_i. We define the timed language of $\mathcal{H}$, denoted $TimedTrace(\mathcal{H})$, to be $\{TimedTrace(\sigma) \mid \sigma \in Exec(\mathcal{H})\}$. Furthermore, we call two hybrid systems $\mathcal{A}$ and $\mathcal{B}$ *timed language equivalent* (denoted by $\mathcal{A} \sim_{\text{tt}} \mathcal{B}$) if $TimedTrace(\mathcal{A}) = TimedTrace(\mathcal{B})$.

3.2 Special Subclasses of Linear Hybrid Automata

In this section, we present two subclasses of linear hybrid automata, namely, timed automata and planar monotonic linear hybrid automata. The decidability of the composition of the automata from the latter class is investigated in the paper, and the decidability proof reduces the reachability problem to that of timed automata.

Timed Automata. A timed automaton is a special type of linear hybrid automaton in which all the variables evolve at a constant rate of 1 in every location. The variables are referred to as clocks. The initial values of the clocks are 0. The resets are given by a pair of guard and zero sets. The guard is a rectangular set which specifies an enabling condition for the edge, and the zero set specifies a subset of the clocks that are reset to 0 with the remaining clock values unaltered during the discrete transition.

Definition 2. *A* timed automaton $\mathcal{H}$ *with* n *clocks is a linear hybrid system of dimension* n *with the following conditions:*

- *Flow maps each location to* 1^n*.*
- *Init maps each location to either the empty set or* $\{0^n\}$*.*
- *for every edge* $e \in Edge$*, there exists a* guard $Guard(e) \in Rect(\mathbb{R}^n)$ *and a* zero set $Zero(e) \subseteq \{1, \ldots, n\}$ *such that* $Reset(e) = \{(\boldsymbol{u}, \boldsymbol{v}) | \boldsymbol{u} \in Guard(e) \wedge \forall i \in \{1, \ldots, n\} \bullet [(i \in Zero(e) \Rightarrow \boldsymbol{v}_i = 0) \wedge (i \notin Zero(e) \Rightarrow \boldsymbol{v}_i = \boldsymbol{u}_i)]\}$*.*
- $Inv, Final \in [Loc \to Rect(\mathbb{R}^n)]$*.*

Planar Monotonic Linear Hybrid System. The main result of the paper is to show that the reachability problem of parallel compositions of a class of two dimensional linear hybrid systems called planar monotonic linear hybrid systems is decidable. *Planar Monotonic Linear Hybrid Systems* were introduced in [11] and their reachability problem was shown to be decidable. A planar monotonic linear hybrid system is a two dimensional system such that all the flows are "monotonic", that is, all the flow vectors have a positive projection on some direction vector. There are no jumps in the system, that is, upon a discrete transition, the values of the continuous states remain the same. We present below a definition of planar monotonic linear hybrid systems which is slightly more general than the original version. In particular, we allow strong resets on edges, wherein the values of the continuous variables are non-deterministically reset to a polyhedral set. Strong resets essentially disengage the continuous states before and after the discrete transition. Also, we allow the flow vectors to be monotonic with respect to different direction vectors; however, we require strong resets on the edges whose source and target have different direction vectors. Further, we require that the edges with visible actions are also strongly reset.

Definition 3. *A planar monotonic linear hybrid system (PMHS for short)* $\mathcal{H}$ *is a linear hybrid system with the following constraints:*

- $\mathcal{H}$ *is* 2 *dimensional.*
- $Inv \in \big[Loc \to OpenPoly\big(\mathbb{R}^2\big)\big]$*.*
- $Init, Final \in \big[Loc \to BOpenPoly\big(\mathbb{R}^2\big)\big]$*.*
- $\mathcal{H}$ *is* monotonic*, that is, there exists a* monotonicity function $Mon \in \big[Loc \to \mathbb{Q}^2\big]$ *that maps each location* q *to a direction* f *such that* $f \cdot Flow(q) > 0$*.*
- *For every* $e \in Edge$*, one of the following is true:*
  1. *There exists* $Guard(e) \in OpenPoly(\mathbb{R}^n)$ *such that* $Reset(e) = \{(x, x) | x \in Guard(e)\}$*. This means that in order to traverse an edge, the current values of the variables must satisfy a* guard $Guard(e)$*, and the values of variables before and after traversing* e *are the same. We call* e *an* identity reset *edge.*
  2. *There exist* $Guard(e), Target(e) \in BOpenPoly(\mathbb{R}^n)$ *such that* $Reset(e) = Guard(e) \times Target(e)$*. This means that in order to traverse an edge, the current values of variables must satisfy a* guard $Guard(e)$ *(same as the previous case), but the values of variables after traversing* e *are reset to some values satisfying the* reset $Target(e)$*. We call* e *a* strong reset *edge.*

– *For every $e \in Edge$, if $Lab(e) \neq \tau$ (a visible edge) or $Mon(Src(e)) \neq Mon(Dest(e))$ (monotonicity function changes), then e must be a strong reset edge.*

We define the polyhedral sets associated with a *PMHS* $\mathcal{H}$, denoted $\mathcal{P}(\mathcal{H})$, to consist of polyhedral sets $Inv(q)$, $Init(q)$ and $Final(q)$ for every location q, and the sets $Guard(e)$ and $Target(e)$ appearing in the reset $Reset(e)$ for every edge e. The set of lines associated with $\mathcal{H}$, denoted $L_{\mathcal{H}}$, is $Lines(\mathcal{P}(\mathcal{H}))$. We call a planar linear hybrid system $\mathcal{H}$ *simple* if no three distinct lines in $L_{\mathcal{H}}$ intersect at a common point. We will assume that the planar linear hybrid systems are simple.

Example 1. Consider the example of a thermostat shown in Figure 1. There are two locations: ON and OFF. It has one variable x that keeps track of temperature. When thermostat is off, temperature decreases with constant rate 2, and when it is on, temperature increases with constant rate 3. If thermostat is off and temperature is less than 19, we can turn it on by moving to location ON. Similarly, if the thermostat is on and temperature is above 21, we can turn it off by moving back to location OFF. When thermostat is off, temperature must always be above 18, and when it is on, temperature must always be below 22. This thermostat is an example of a *PMHS*. If we add an additional variable that behaves like a clock, then the automaton is monotonic.

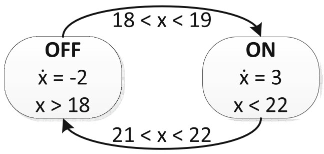

Fig. 1. Thermostat Example

3.3 Parallel Composition of Linear Hybrid Automata

The parallel composition of two linear hybrid automata corresponds to executing the two automata simultaneously with the restriction that they synchronize on common labels, that is, a transition labelled by a common label occurs only if both the automata execute a discrete transition labelled by the common label.

Definition 4. *For two linear hybrid systems $\mathcal{H}_{\mathcal{A}}$ and $\mathcal{H}_{\mathcal{B}}$, their parallel composition $\mathcal{H}_{\mathcal{A}} \parallel \mathcal{H}_{\mathcal{B}}$ is a linear hybrid system $\mathcal{H}_{\mathcal{C}}$ which is defined as follows:*

– $Loc_{\mathcal{C}} = Loc_{\mathcal{A}} \times Loc_{\mathcal{B}}$
– $dim_{\mathcal{C}} = dim_{\mathcal{A}} + dim_{\mathcal{B}}$
– $Act_{\mathcal{C}} = Act_{\mathcal{A}} \cup Act_{\mathcal{B}}$
– $Inv_{\mathcal{C}} = Inv_{\mathcal{A}} \otimes Inv_{\mathcal{B}}$
– $Flow_{\mathcal{C}} = Flow_{\mathcal{A}} \times Flow_{\mathcal{B}}$
– $Init_{\mathcal{C}} = Init_{\mathcal{A}} \otimes Init_{\mathcal{B}}$
– $Final_{\mathcal{C}} = Final_{\mathcal{A}} \otimes Final_{\mathcal{B}}$
– *$Edge_{\mathcal{C}}$ is the set of edges $((p_1, p_2), (q_1, q_2), l, r)$ which satisfy the following:*
  P_1 : *If $l \in Act_{\mathcal{A}} \cap Act_{\mathcal{B}}$, then $\exists r_1, r_2 \bullet (p_1, q_1, l, r_1) \in Edge_{\mathcal{A}} \wedge (p_2, q_2, l, r_2) \in Edge_{\mathcal{B}} \wedge r = r_1 \times r_2$*
  P_2 : *If $l \notin Act_{\mathcal{B}}$, then $p_2 = q_2 \wedge \exists r_1 \bullet r = r_1 \times id_{\mathbb{R}^{dim_{\mathcal{B}}}} \wedge (p_1, q_1, l, r_1) \in Edge_{\mathcal{A}}$*
  P_3 : *If $l \notin Act_{\mathcal{A}}$, then $p_1 = q_1 \wedge \exists r_2 \bullet r = id_{\mathbb{R}^{dim_{\mathcal{A}}}} \times r_2 \wedge (p_2, q_2, l, r_2) \in Edge_{\mathcal{B}}$*

  P_1 represents edges in both $\mathcal{H}_{\mathcal{A}}$ and $\mathcal{H}_{\mathcal{B}}$ such that their labels are in the common alphabet of $\mathcal{H}_{\mathcal{A}}$ and $\mathcal{H}_{\mathcal{B}}$. P_2 (P_3) represents edges in $\mathcal{H}_{\mathcal{A}}$ ($\mathcal{H}_{\mathcal{B}}$) such

that their labels are not in the alphabet of $\mathcal{H}_\mathcal{B}$ ($\mathcal{H}_\mathcal{A}$). Note that when $l = \tau$, the premises of both P_2 and P_3 hold.

Lemma 1 (Miller [9]). *For any linear hybrid automata $\mathcal{H}_1$, $\mathcal{H}_1'$, $\mathcal{H}_2$, and $\mathcal{H}_2'$ if $\mathcal{H}_1 \sim_{\mathrm{tt}} \mathcal{H}_1'$ and $\mathcal{H}_2 \sim_{\mathrm{tt}} \mathcal{H}_2'$ then $\mathcal{H}_1 \parallel \mathcal{H}_2 \sim_{\mathrm{tt}} \mathcal{H}_1' \parallel \mathcal{H}_2'$.*

Lemma 2 (Alur *et al.* [2]). *For any finite set of timed automata $\mathcal{T}_1, \ldots, \mathcal{T}_n$, reachability problem for $\mathcal{T}_1 \parallel \ldots \parallel \mathcal{T}_n$ is decidable in* PSPACE*.*

4 Timed Language Equivalence of PMHS and Timed Automata

Before presenting our decidability result for reachability in Section 5, we present the key idea that enables this decidability result to go through, namely, that the timed language of any *PMHS* is equivalent to that of a timed automaton that can be effectively constructed.

Theorem 1. *The timed language of a PMHS $\mathcal{H}$ is equivalent to the timed language of a timed automaton $TA(\mathcal{H})$ computable from $\mathcal{H}$ in EXPSPACE.*

We will now sketch the ideas behind the proof of Theorem 1. Consider any execution σ of *PMHS* $\mathcal{H}$. The timed automaton $TA(\mathcal{H})$ will "simulate" this execution σ of $\mathcal{H}$ by an execution σ' such that the strong reset transitions taken in σ and σ' are the same, in the same order, and at the same times. Since every visible transition of $\mathcal{H}$ is a strong reset transition, this ensures that $\mathit{TimedTrace}(\sigma)$ is the same as $\mathit{TimedTrace}(\sigma')$. Now if $TA(\mathcal{H})$ simulates (in this manner) all executions of $\mathcal{H}$, and if $TA(\mathcal{H})$ only has such executions, then the timed languages of $\mathcal{H}$ and $TA(\mathcal{H})$ are the same. The executions of the timed automaton $TA(\mathcal{H})$ will only consist of a sequence of strong reset transitions of $\mathcal{H}$. If a strong reset edge e_2 is taken immediately after a strong reset edge e_1, then $TA(\mathcal{H})$ will ensure that the time elapsed between taking e_1 and e_2 is the same as the duration of some reset-free execution fragment of $\mathcal{H}$ that starts in some state in $\mathit{Target}(e_1)$ and ends in some state in $\mathit{Guard}(e_2)$. Notice, that this will immediately guarantee that the executions of $TA(\mathcal{H})$ "simulate" (in the sense outlined above) executions of $\mathcal{H}$. The automaton $TA(\mathcal{H})$ will maintain such constraints by having a clock variable that measures the time between successive transitions, and having locations that remember the last strong reset edge taken.

Before giving a formal definition, we introduce some concepts that will help us describe $TA(\mathcal{H})$ precisely. Given a *PMHS* $\mathcal{H}$, let us denote by $\mathit{REdge}(\mathcal{H})$, the strong reset edges of $\mathcal{H}$. We say that $\mathcal{H}$ is a *reset-free PMHS*, if $\mathit{REdge}(\mathcal{H}) = \emptyset$. For any *PMHS* $\mathcal{H} = (\mathit{Loc}, \mathit{dim}, \mathit{Act}, \mathit{Inv}, \mathit{Flow}, \mathit{Edge}, \mathit{Init}, \mathit{Final})$ and $\mathit{Init}', \mathit{Final}' \in \left[\mathit{Loc} \rightarrow \mathit{BOpenPoly}(\mathbb{R}^2)\right]$, we define $\mathit{ResetFree}(\mathcal{H}, \mathit{Init}', \mathit{Final}')$ to be the *PMHS* $(\mathit{Loc}, \mathit{dim}, \mathit{Act}, \mathit{Inv}, \mathit{Flow}, \mathit{Edge}', \mathit{Init}', \mathit{Final}')$, where $\mathit{Edge}' = \mathit{Edge} \setminus \mathit{REdge}(\mathcal{H})$. Finally, for a *PMHS* $\mathcal{H}$, $\mathtt{DReach}(\mathcal{H})$ will denote the durations of all the executions of $\mathcal{H}$, i.e., $\mathtt{DReach}(\mathcal{H}) = \{t \mid \exists \sigma \in \mathit{Exec}(\mathcal{H}).\ t = \mathit{Duration}(\sigma)\}$.

Definition 5. *For a PMHS $\mathcal{H} = (Loc, dim, Act, Inv, Flow, Edge, Init, Final)$, define the timed automaton $TA(\mathcal{H})$ to be $(Loc', dim' = 1, Act, Inv', Flow', Edge', Init', Final')$ where*

- $Loc' = REdge(\mathcal{H}) \cup \{q_{Init}, q_{Final}\}$,
- *For every* $q \in Loc'$, $Inv'(q) = \mathbb{R}$, $Flow'(q) = 1$, $Init'(q) = \{0\}$ *if* $q = q_{Init}$ *and* $\emptyset$ *otherwise, and* $Final'(q) = \mathbb{R}$ *if* $q = q_{Final}$ *and* $\emptyset$ *otherwise,*
- $Edge'$ *is the set of all edges* e' *such that* $Zero(e') = \{1\}$ *(clock is always reset),* $(Src(e'), Dest(e')) \in (Loc' \setminus \{q_{Final}\}) \times (Loc' \setminus \{q_{Init}\})$ *and the following conditions hold:*
  1. *If* $Dest(e') \in REdge(\mathcal{H})$ *then* $Lab'(e') = Lab(Dest(e'))$ *and if* $Dest(e') = q_{Final}$ *then* $Lab'(e') = \tau$.
  2. $Guard(e') = \mathtt{DReach}(ResetFree(\mathcal{H}, Init', Final'))$, *where*
     - *If* $Src(e') = q_{Init}$ *then* $Init' = Init$. *If* $Src(e') \in REdge(\mathcal{H})$ *then* $Init'(p) = Target(Src(e'))$, *if* $p = Dest(Src(e'))$, $Init'(p) = \emptyset$ *otherwise.*
     - *If* $Dest(e') = q_{Final}$ *then* $Final' = final$. *If* $Dest(e') \in REdge(\mathcal{H})$ *then* $Final'(p) = Guard(Dest(e'))$ *if* $p = Src(Dest(e'))$, *and* $Final'(p) = \emptyset$ *otherwise.*

Definition 5 formalizes the intuition outlined at the beginning of this section, and so, its timed language is equivalent to the timed language of $\mathcal{H}$. However, to finish the proof of Theorem 1, we still need to establish two facts. First, the automaton outlined above is a timed automaton only if the guards in the above definition are "nice" sets; in particular they need to be finite unions of intervals [1]. Second, to argue that $TA(\mathcal{H})$ can be effectively constructed from $\mathcal{H}$, we need to show that the transition guards can be computed. These two requirements do indeed turn out to be true, and is established in Lemma 3.

Lemma 3. *Given a reset-free PMHS $\mathcal{H}$, the set* $\mathtt{DReach}(\mathcal{H})$ *is computable and is a finite union of intervals.*

The proof of Lemma 3 relies on the following key lemma. Define a timed reachability predicate for an automaton $\mathcal{H}$ as $\mathtt{TReach}_{\mathcal{H}}^{q_1,q_2} = \{(\mathbf{u}, t, \mathbf{v}) \mid \exists \sigma \in Exec(\mathcal{H}), \sigma^s(0) = (q_1, \mathbf{u}), \sigma^s(|\sigma|) = (q_2, \mathbf{v}), Duration(\sigma) = t\}$.

Lemma 4. *For a reset-free PMHS $\mathcal{H}$ and $q_1, q_2 \in Loc$, there is a first order logic formula $\varphi_{\mathcal{H}}^{q_1,q_2}(\boldsymbol{x}, \tau, \boldsymbol{y})$ over $(\mathbb{R}, +, <)$ such that $(\boldsymbol{v}, t, \boldsymbol{v}') \in \mathtt{TReach}_{\mathcal{H}}^{q_1,q_2}$ iff $\varphi_{\mathcal{H}}^{q_1,q_2}(\boldsymbol{v}/\boldsymbol{x}, t/\tau, \boldsymbol{v}'/\boldsymbol{y})$ holds. Further, $\varphi_{\mathcal{H}}^{q_1,q_2}$ is only existentially quantified and its length is bounded by an exponential in the size of $\mathcal{H}$.*

Proof. (Sketch.) The proof builds upon the results in [11], where it is shown that the problem of point-to-point and region-to-region reachability is decidable for the class of reset-free *PMHS*. Below we present briefly an overview of the proof in [11], and highlight the changes in extending it to prove the current lemma.

[1] Typically, the guards in a timed automaton are intervals. But transitions with finite unions of intervals as guards can be thought of as a set of finitely many nondeterministic transitions on intervals.

The first step in [11] is to divide the state-space of $\mathcal{H}$ into regions where each region is such that if it intersects with a guard or an invariant of $\mathcal{H}$, then it is contained in it. The maximal set of such regions can be uniquely determined for $\mathcal{H}$ and effectively computable. A tree is constructed which has as nodes subsets of edges of the regions. The children together capture the set of all states reached from the states in the parent by a "region-execution" — executions which remain within a single region. The main technical challenge lies in computing the children, since the number of mode switches in a region-execution between two edges of a region is not bounded. The challenging case is when the hybrid system restricted to the region has cycles. Hence, the problem is reduced to computing the reachable set of a hybrid automaton restricted to a region whose underlying graph is strongly connected. Here, it is shown that the following property P holds, that is, the reach set can be characterized by states reached by certain "executions" with bounded number of mode switches which can potentially violate the invariants and guards. The reach set can then be computed by a finite number of state-space exploration steps. More importantly, the monotonicity property of the flows ensures that the height of the tree is bounded, when the initial and final regions are bounded. And a final region is reachable if and only if one of the trees with root corresponding to an edge of the initial region, contains a node which has an edge of the final region.

In this paper, we extend the proof to compute the predicate $\varphi_{\mathcal{H}}^{q_1,q_2}(\mathbf{x},\tau,\mathbf{y})$. Note that if node N_2 is a descendant of N_1, then there is an execution from every state in N_1 to some state in N_2. Our main idea is to extend the information along an edge in the tree to capture a ternary relation consisting of tuples $(\mathbf{u}, t, \mathbf{v})$ such that there is a region execution from $\mathbf{u}$ on node N_1 to node $\mathbf{v}$ on its child N_2 of duration t. Our main observation is that the boundedness property in P holds even when we require the executions to be of equal duration. More precisely, $\mathbf{v}$ can be reached from $\mathbf{u}$ by a region-execution of duration t if and only if there exists a certain "execution" with bounded number of mode switches from $\mathbf{u}$ to $\mathbf{v}$ of duration t potentially violating the invariants and guards. Further, we show that this predicate can be captured as an existentially quantified first-order logic formula over $(\mathbb{R}, +, <)$ with only conjunctions. The predicate $\varphi_{\mathcal{H}}^{q_1,q_2}(\mathbf{x},\tau,\mathbf{y})$ is then constructed by composing the predicates corresponding to the edges.

Next, we provide an upper bound on the length of $\varphi_{\mathcal{H}}^{q_1,q_2}(\mathbf{x},\tau,\mathbf{y})$. Let n be the size of the input representation of $\mathcal{H}$. Note that the number of constraints used in the description of the invariants and guards is at most n. Hence, the number of regions associated with it is at most 2^n, and the number of nodes in the tree is linear in the number of regions and is bounded by $O(2^n)$ [11]. If L is a bound on the length of the predicate corresponding to an edge, then the length of the predicate $\varphi_{\mathcal{H}}^{q_1,q_2}(\mathbf{x},\tau,\mathbf{y})$ is bounded by $O(L2^n)$. L is bounded by a polynomial in the size of the automaton. Hence, $\varphi_{\mathcal{H}}^{q_1,q_2}(\mathbf{x},\tau,\mathbf{y})$ is bounded by $2^{O(n)}$.

We now complete the proof of Lemma 3 (and therefore, Theorem 1). Recall that the first order theory of $(\mathbb{R}, +, <)$ is *o-minimal* [12]. Therefore, any set defined by a first order formula with one free variable in this structure is a finite union of intervals [12]. Finally, the theory of $(\mathbb{R}, +, <)$ has a *PSPACE* quantifier elimination procedure for existentially quantified formulas (formulas with no quantifier alternation), and hence, the intervals in the finite union are computable. Now since $\texttt{DReach}(\mathcal{H}) = \exists \mathbf{x}, \mathbf{y}.\ \vee_{q_1,q_2} \varphi_{\mathcal{H}}^{q_1,q_2}(\mathbf{x}, \tau, \mathbf{y})$, we have established Lemma 3. Further, since, the length of the formula $\varphi_{\mathcal{H}}^{q_1,q_2}$ is at most exponential in the size of $\mathcal{H}$, $\texttt{DReach}(\mathcal{H})$ can be computed in *EXPSPACE*, and hence $TA(\mathcal{H})$ can be computed in *EXPSPACE*.

5 Main Result

The timed trace equivalence of *PMHS* with timed automata (Theorem 1) allows us to prove the following main result of this paper.

Theorem 2. *The following problem is decidable in EXPSPACE: Given hybrid automata $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k$ such that each $\mathcal{H}_i$ is either a PMHS or an initialized rectangular hybrid automaton, is Exec$(\mathcal{H}_1 \parallel \mathcal{H}_2 \parallel \ldots \parallel \mathcal{H}_k)$ empty?*

Proof. (Proof Sketch). Theorem 1 shows that any *PMHS* $\mathcal{H}$ is equivalent to the timed language of a timed automaton $TA(\mathcal{H})$ which can be constructed in *EXPSPACE*. Similarly, Miller [9] showed that any initialized rectangular hybrid automaton is also timed language equivalent to a timed automaton, constructible in *PSPACE*. Hence, for each $\mathcal{H}_i$, we can construct a timed automaton $TA(\mathcal{H}_i)$ such that $\mathcal{H}_i \sim_{\texttt{tt}} TA(\mathcal{H}_i)$. From Lemma 1, $\mathcal{H}_1 \parallel \mathcal{H}_2 \parallel \ldots \parallel \mathcal{H}_k \sim_{\texttt{tt}} TA(\mathcal{H}_1) \parallel TA(\mathcal{H}_2) \parallel \ldots \parallel TA(\mathcal{H}_k)$. Note that the latter is a composition of k timed automata. Therefore, from Lemma 2, the reachability problem, namely, if $Exec(TA(\mathcal{H}_1) \parallel TA(\mathcal{H}_2) \parallel \ldots \parallel TA(\mathcal{H}_k))$, is empty, is decidable in *PSPACE*. Equivalently, the emptiness of $Exec(\mathcal{H}_1 \parallel \mathcal{H}_2 \parallel \ldots \parallel \mathcal{H}_k)$, is decidable in *EXPSPACE*.

The above theorem, in particular, implies that the control state reachability problem is decidable for the composition of *PMHS*.

6 A Decidable Class of Multi-rate Automata

A multi-rate automaton is a generalization of timed automaton, where the continuous variables need not flow at rate 1. The reachability problem for general multi-rate automata is known to be undecidable [1,7]. In this section we identify a new subclass of multi-rate automata with a decidable reachability problem. The decidability result will be a consequence of our main result (Theorem 2). We begin by recalling what a multi-rate automaton is.

Definition 6. *A* multi-rate automaton *of dimension n is a linear hybrid system $\mathcal{H}$ with the restriction that*

– *$Inv \in [Loc \rightarrow OpenRect(\mathbb{R}^n)]$,*
– *Init maps each location to either the empty set or $\{0^n\}$,*
– *$Final \in [Loc \rightarrow PBOpenRect(\mathbb{R}^n)]$, and*

– *for each edge $e \in$ Edge, there exists a* guard $\mathit{Guard}(e) \in \mathit{PBOpenRect}(\mathbb{R}^n)$ *and a* zero set $\mathit{Zero}(e) \subseteq \{1, \ldots, n\}$ *such that* $\mathit{Reset}(e) = \{(\boldsymbol{u}, \boldsymbol{v}) | \boldsymbol{u} \in \mathit{Guard}(e) \wedge \forall i \in \{1, \ldots, n\} \bullet [(i \in \mathit{Zero}(e) \Rightarrow \boldsymbol{v}_i = 0) \wedge (i \notin \mathit{Zero}(e) \Rightarrow \boldsymbol{v}_i = \boldsymbol{u}_i)]\}$.

In the above definition, we assume that variables reset on an edge are reset to 0. However, this condition can be relaxed to one where the variables are reset to any value in a bounded interval without affecting the decidability results. We make the simplifying assumption to make the presentation and notation simpler.

We identify a special subclass of multi-rate automata that we call phased multi-rate automata. Phased multi-rate automata are such that every execution of the machine can be divided into "phases". Each phase begins with a discrete transition that resets some set of variables, and every other discrete transition in the phase, leaves the continuous variables unchanged. In addition, during a phase, after the first transition, the flow of at most one variable can change. We will show that the reachability problem for such automata is decidable. Before defining this class, we introduce some definitions and notations that we will need.

Affected and Used Variables. Consider a multi-rate automaton $\mathcal{H}$ of dimension n and an edge e of $\mathcal{H}$. We will say that a variable $i \in \{1, \ldots n\}$ is *affected* by edge e, if either (a) i is reset, i.e., $i \in \mathit{Zero}(e)$, or (b) i's flow changes after taking the edge, i.e., $\mathit{Flow}(\mathit{Src}(e))_i \neq \mathit{Flow}(\mathit{Dest}(e))_i$. The set of variables affected by e will be denoted by $\mathit{affect}(e)$. A variable i is *used* by edge e if either (a) i is affected by e, i.e., $i \in \mathit{affect}(e)$, or (b) i appears in either $\mathit{Inv}(\mathit{Src}(e))$ or $\mathit{Inv}(\mathit{Dest}(e))$, i.e., I_i or I'_i not equal to $(-\infty, \infty)$, where $\mathit{Inv}(\mathit{Src}(e)) = I_1 \times \ldots \times I_n$ and $\mathit{Inv}(\mathit{Dest}(e)) = I'_1 \times \ldots \times I'_n$, or (c) variable i appears in $\mathit{Guard}(e)$, i.e., $I_i \neq (-\infty, \infty)$, where $\mathit{Guard}(e) = I_1 \times \ldots \times I_n$. The set of variables used by e is denoted by $\mathit{use}(e)$.

Phase Consistency. Consider a path $\pi = e_1, e_2, \ldots e_k$ of $\mathcal{H}$. A *phase* of π is a pair $(i, j) \in \{0, 1, \ldots k+1\}^2$ with $i < j$ such that (a) if $i > 0$ then $\mathit{Zero}(e_i) \neq \emptyset$, (b) if $j < k+1$ then $\mathit{Zero}(e_j) \neq \emptyset$, and (c) for all ℓ, $i < \ell < j$, $\mathit{Zero}(e_\ell) = \emptyset$. In other words, (i, j) is a phase if e_i and e_j are successive reset edges in π. In the definitions that follow, we will find it convenient to take $\mathit{Zero}(e_0) = \mathit{Zero}(e_{k+1}) = \{1, \ldots n\}$. The path π is *phase consistent* for phase (i, j) if there is a variable $x_{ij} \in \{1, \ldots n\}$ such that (a) $x_{ij} \in \mathit{Zero}(e_i) \cap \mathit{Zero}(e_j)$, (b) for all $i < \ell < k$, $\mathit{affect}(e_\ell) \subseteq \{x_{ij}\}$ and $\mathit{use}(e_\ell) \subseteq \mathit{Zero}(e_i)$, and (c) $\mathit{use}(e_j) \backslash \mathit{Zero}(e_j) \subseteq \mathit{Zero}(e_i)$. When this happens, x_{ij} is said to be the *phase variable* of π in (i, j). We will say π is phase consistent if it is phase consistent for every phase (i, j).

Definition 7. *A* phased multi-rate automaton (PMA) *of dimension n is multi-rate automaton $\mathcal{H}$ with the following restrictions.*

– *For every edge e of $\mathcal{H}$, if $|\mathit{affect}(e)| > 1$ then $\mathit{affect}(e) \subseteq \mathit{Zero}(e)$. In other words, if more than one variable is affected, then all affected variables are reset.*
– *Every path π of $\mathcal{H}$ is phase consistent with respect to every phase.*

Theorem 3. *Reachability problem for phased multi-rate automata is decidable.*

Phased multi-rate automata are incomparable to the class of initialized multi-rate automata. Recall that in an initialized multi-rate automaton, a variable must be reset if its flow changes. This is not required in a phased multi-rate automaton as the phase variable can change its flow repeatedly without being reset. On the other hand, in phased multi-rate automata, the phase variable must be reset at the start and end of a phase; there is no analogous restriction in initialized multi-rate automata.

Example 2. Figure 2 shows an example of a water tank system. Here, we have two tanks and one hose. The hose could be on or off, and it could point to tank 1 or tank 2. Water is added at a constant rate to a tank when hose is on and pointing to that tank. Also, both tanks are leaking at a constant rate. One can turn the hose on and off at any time. However, to move the hose from tank 1 to tank 2, the tank 1 must have a sufficiently high level of water, and tank 2 must have a low level; similar constraints are required to be satisfied when moving the hose from tank 2 to tank 1. The level in a tank is considered "sufficiently high" if the level is between almost full (AF) and full (F), and it is low if it is between almost low (AL) and low (L). Observe that such a system is not initialized as the rate of change of level in a tank can change due to turning the hose on and off. However, it is a PMA as such changes to flow without resets are allowed. This automaton has the slightly more general form of resets where a variable can be reset to any value in a bounded set.

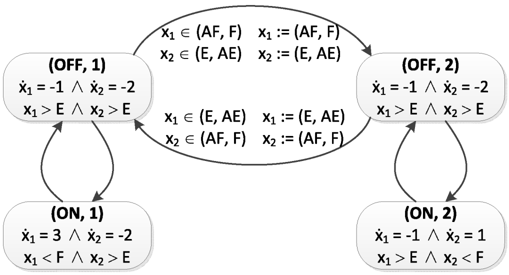

Fig. 2. Water Tank Example

7 Conclusion

Our main result proved that the reachability problem is decidable for the parallel composition of a collection of planar monotonic linear hybrid systems. Our proof extends the observations in [11] to first showing that timed reachability (and not just reachability) is decidable by reducing it to the theory of linear arithmetic. This result allows us to conclude that the planar monotonic linear hybrid automata are timed trace equivalent to timed automata. Finally, our decidability result for concurrent planar monotonic linear hybrid systems follows from the decidability of the emptiness problem for timed trace language of concurrent timed automata. One consequence of our results is that it identifies a new decidable subclass of multi-rate automata, namely, phased multi-rate automata. Our results present a general technique of lifting decidability results for low dimensional systems to hybrid automata with many continuous variables.

A future direction of research would be to see if this idea can be applied to other decidable planar hybrid automata, i.e., automata with 2 variables.

References

1. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. TCS **138**(1), 3–34 (1995)
2. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science **126**, 183–235 (1994)
3. Asarin, E., Maler, O., Pnueli, A.: Reachability analysis of dynamical systems having piecewise-constant derivatives. Theoretical Computer Science **138**(1), 35–65 (1995)
4. Asarin, E., Schneider, G., Yovine, S.: Algorithmic analysis of polygonal hybrid systems. Part I: Reachability. TCS **379**(1–2), 231–265 (2007)
5. Casagrande, A., Corvaja, P., Piazza, C., Mishra, B.: Decidable compositions of o-minimal automata. In: Cha, S.S., Choi, J.-Y., Kim, M., Lee, I., Viswanathan, M. (eds.) ATVA 2008. LNCS, vol. 5311, pp. 274–288. Springer, Heidelberg (2008)
6. Henzinger, T.A.: The theory of hybrid automata. In: Proceeding of IEEE Symposium on Logic in Computer Science, pp. 278–292 (1996)
7. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? Journal of Computer and System Sciences **57**(1), 373–382 (1995)
8. Lafferriere, G., Pappas, G., Sastry, S.: o-minimal hybrid systems. MCSS **13**, 1–21 (2000)
9. Miller, J.S.: Decidability and complexity results for timed automata and semi-linear hybrid automata. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 296–309. Springer, Heidelberg (2000)
10. Mysore, V., Pnueli, A.: Refining the undecidability frontier of hybrid automata. In: Sarukkai, S., Sen, S. (eds.) FSTTCS 2005. LNCS, vol. 3821, pp. 261–272. Springer, Heidelberg (2005)
11. Prabhakar, P., Vladimerou, V., Viswanathan, M., Dullerud, G.: A decidable class of planar linear hybrid systems. Theoretical Computer Science **574**, 1–17 (2015)
12. L. van den Dries: Tame Topology and O-minimal Structures. Cambridge UnivesityPress (1998)
13. Vladimerou, V., Prabhakar, P., Viswanathan, M., Dullerud, G.E.: STORMED Hybrid Systems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 136–147. Springer, Heidelberg (2008)