Decidability and Complexity Results for Timed Automata and Semi-linear Hybrid Automata

Joseph S. Miller *

Department of Mathematics
Cornell University
Ithaca, NY 14853
jmiller@math.cornell.edu

Abstract. We define a new class of hybrid automata for which reachability is decidable—a proper superclass of the *initialized rectangular hybrid automata*—by taking *parallel compositions* of simple components. Attempting to generalize, we encounter timed automata with algebraic constants. We show that reachability is undecidable for these *algebraic timed automata* by simulating two-counter Minsky machines. Modifying the construction to apply to *parametric timed automata*, we reprove the undecidability of the emptiness problem, and then distinguish the dense and discrete-time cases with a new result. The algorithmic complexity—both classical and parametric—of one-clock parametric timed automata is also examined. We finish with a table of computability-theoretic complexity results, including that the existence of a Zeno run is Σ^1_1-complete for semi-linear hybrid automata; it is too complex to be expressed in first-order arithmetic.

1 Introduction

Though the bulk of this paper will be given over to undecidability results, our initial motivation is the extension, even by a small amount, of the class of hybrid automata for which reachability is known to be decidable. It has been suggested that it is the coupling of continuous variables which leads to undecidability [7]. Parallel composition couples only the discrete dynamics of its components. Thus, arguing informally, if we consider parallel compositions of hybrid automata which obey a sufficient decoupling between discrete and continuous dynamics, then we should be able to circumvent undecidability. We will bring this simple idea to a simple fruition in Sect. 2, but first we must dispose of the preliminaries.

1.1 Hybrid Automata

A hybrid system is a physical system which combines discrete and continuous dynamics. *Hybrid automata* are intended as formal mathematical models of such

* Research supported by the ARO under the MURI program "Integrated Approach to Intelligent Systems", grant no. DAA H04-96-1-0341.

systems. The following definition is provided to fix notation for the duration of this paper. Though no standard definition exists, this one is not unusual. Note that the continuous dynamical behavior is expressed by a (non-deterministic) semi-flow, not by vector fields as is more common.

Definition 1. A *hybrid automaton* $\mathcal{A}$ is a tuple $(\mathcal{Q}, \mathcal{E}, \mathcal{X}, \mathcal{I}, \mathcal{S}, \mathfrak{s}, \mathfrak{d}, \mathcal{R}, \Phi)$ such that:

- [*discrete states*] $\mathcal{Q}$ is a finite set
- [*edges*] $\mathcal{E}$ is a finite set
- [*plant states*] $\mathcal{X}$ is any set (usually taken to be a manifold)
- [*invariant set*] $\mathcal{I} \subseteq \mathcal{Q} \times \mathcal{X}$
- [*initial set*] $\mathcal{S} \subseteq \mathcal{Q} \times \mathcal{X}$
- [*source map*] $\mathfrak{s} : \mathcal{E} \to \mathcal{Q}$
- [*destination map*] $\mathfrak{d} : \mathcal{E} \to \mathcal{Q}$
- [*reset relation*] $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{E} \times \mathcal{X}$
- [*semi-flow*] $\Phi : \mathcal{Q} \times \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathcal{P}(\mathcal{X})$ such that for all $(q, x) \in \mathcal{Q} \times \mathcal{X}$:
  1. $\Phi(q, x, 0) = \{x\}$
  2. $\forall t_1, t_2 \in \mathbb{R}_{\geq 0}\ \ \Phi(q, x, t_1 + t_2) = \bigcup_{y \in \Phi(q,x,t_1)} \Phi(q, y, t_2)$.

The components of a hybrid automaton $\mathcal{A}$ are written with $\mathcal{A}$ as a superscript, as in $\mathcal{Q}^{\mathcal{A}}$, $\mathfrak{s}^{\mathcal{A}}$ and $\Phi^{\mathcal{A}}$. The superscript may be omitted when the automaton is clear from context. $\mathcal{I}_q$ denotes the invariant set in discrete state q and is taken to be a subset of $\mathcal{X}$. Similarly, $\mathcal{S}_q$, $\mathcal{R}_e$ and Φ_q are given their expected interpretations as subsets of $\mathcal{X}$, $\mathcal{X}^2$ and $\mathcal{X} \times \mathbb{R}_{\geq 0} \times \mathcal{X}$, respectively. Finally, by the *guard* of and edge $e \in \mathcal{E}$ we refer to the support of the reset relation $\mathcal{R}_e$.

Definition 2. A *run* of a hybrid automaton $\mathcal{A}$ is a sequence $(q_0, x_0, f_0, t_0, y_0, e_0, q_1, x_1, f_1, t_1, y_1, e_1, \ldots, e_{n-1}, q_n, x_n, f_n, t_n, y_n)$ such that for all $0 \leq i \leq n$:

- $q_i \in \mathcal{Q}$
- $x_i, y_i \in \mathcal{X}$
- $(q_0, x_0) \in \mathcal{S}$
- $t_i \in \mathbb{R}_{\geq 0}$
- $f_i : [0, t_i] \to \mathcal{X}$
- $f_i(0) = x_i$ and $f_i(t_i) = y_i$
- $\forall t \in [0, t_i)\ f_i(t) \in \mathcal{I}$
- $\forall s, t \in [0, t_i]\ s < t \longrightarrow f_i(t) \in \Phi(q_i, f_i(s), t - s)$

and for all $0 \leq i < n$:

- $e_i \in \mathcal{E}$
- $\mathfrak{s}(e_i) = q_i$ and $\mathfrak{d}(e_i) = q_{i+1}$
- $(y_i, e_i, x_{i+1}) \in \mathcal{R}$.

In Sect. 5 we will generalize the notion of run both by allowing the final time interval to be infinite and by allowing infinite sequences of transitions. Until then, finite runs will be more convenient.

Definition 3. The *semi-linear* (resp. *semi-algebraic*) subsets of $\mathbb{R}^n$ are formed by taking boolean combinations of sets defined by linear (resp. algebraic) equalities and inequalities with rational coefficients.

Definition 4. By *semi-linear hybrid automata* (SLHA) we mean that elusive class of automata which has been variously known as polyhedral and—to the consternation of control theorists—as linear. $\mathcal{A}$ is an *n-dimensional* SLHA if:

- $\mathcal{X}^{\mathcal{A}} = \mathbb{R}^n$ for some n
- for every $q \in \mathcal{Q}^{\mathcal{A}}$ and $e \in \mathcal{E}^{\mathcal{A}}$, the projected components $\mathcal{I}_q^{\mathcal{A}}$, $\mathcal{S}_q^{\mathcal{A}}$, $\mathcal{R}_e^{\mathcal{A}}$ and $\Phi_q^{\mathcal{A}}$ are semi-linear subsets of $\mathbb{R}^n$, $\mathbb{R}^n$, $\mathbb{R}^{2n}$ and $\mathbb{R}^{2n+1}$, respectively.

Semi-algebraic hybrid automata are defined analogously.

1.2 Annotated Hybrid Automata

It will be convenient to add a layer of abstraction to our hybrid automata. An *annotation* associates to each edge an *event* and to each discrete state a nonempty set of possible *conditions*. These annotations do not affect the behavior of the automaton but will be used when we define the *timed language* of an automaton and when we define the operation of *parallel composition*.

Definition 5. An *annotated hybrid automaton* $\mathcal{A}$ is a hybrid automaton with four additional components $(\Sigma, \Gamma, \mathfrak{e}, \mathfrak{c})$:

- [*events*] Σ is a finite set
- [*conditions*] Γ is a finite set
- [*event assignment*] $\mathfrak{e} : \mathcal{E} \to \Sigma$
- [*condition assignment*] $\mathfrak{c} : \mathcal{Q} \to \mathcal{P}(\Gamma)$ such that $\forall q \in \mathcal{Q}\ \mathfrak{c}(q) \neq \emptyset$.

Definition 6. To each run $(q_0, x_0, f_0, t_0, y_0, e_0, q_1, x_1, f_1, t_1, y_1, e_1, \ldots, e_{n-1}, q_n, x_n, f_n, t_n, y_n)$ of an annotated hybrid automaton $\mathcal{A}$, we associate an *annotated run* $(c_0, t_0, v_0, c_1, t_1, v_1, \ldots, c_n, t_n)$ such that:

- for all $0 \leq i \leq n$, $c_i \in \mathfrak{c}(q_i)$
- for all $0 \leq i < n$, $v_i = \mathfrak{e}(e_i)$.

The *timed language* $\mathcal{L}(\mathcal{A})$ of an annotated hybrid automaton $\mathcal{A}$ is set of all annotated runs of $\mathcal{A}$.

The following equivalence relation will be important.

Definition 7. We say that the annotated hybrid automata $\mathcal{A}$ and $\mathcal{B}$ are *language equivalent* iff:

- $\Sigma^{\mathcal{A}} = \Sigma^{\mathcal{B}}$
- $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$.

We denote language equivalence by $\mathcal{A} \sim_{\mathfrak{le}} \mathcal{B}$.

Remark 1. Invoking symmetry, one might expect the requirement that $\Gamma^{\mathcal{A}} = \Gamma^{\mathcal{B}}$ in the definition of language equivalence. We disclude this requirement because it is unnecessary, though it would not falsify the results that follow. The interested reader should note in Sect. 1.3 that the set Γ does not play a very important role in parallel composition, while Σ is crucial.

By the *reachability problem* for an annotated hybrid automaton $\mathcal{A}$, we mean the problem of determining which conditions $c \in \Gamma^{\mathcal{A}}$ occur on *some* annotated run. This ensures that language equivalent hybrid automata have equivalent

reachability problems. Of course, the reachability of a discrete state can be detected with a suitable annotation and we may suppress explicit mention of annotations when discussing reachability. We say that the reachability problem is decidable for a class $\mathcal{K}$ if there is an algorithm which uniformly solves the reachability problem for every member of $\mathcal{K}$.

1.3 Parallel Composition

Given two annotated hybrid automata we define a product automaton called the *parallel composition*. Conceptually, a run of the parallel composition is comprised of simultaneous runs of the component automata which are independent except that:

- They must synchronize on shared events.
- The only product states that are permitted are those for which the restrictions on conditions are jointly satisfiable.

Definition 8. We define the *parallel composition* $\mathcal{A} \parallel \mathcal{B}$ of the annotated hybrid automata $\mathcal{A}$ and $\mathcal{B}$ in two stages. First, we define a synchronized product automaton $\mathcal{A} \otimes \mathcal{B}$ such that:

- $\mathcal{Q} = \mathcal{Q}^{\mathcal{A}} \times \mathcal{Q}^{\mathcal{B}}$
- $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$ where:
  $\mathcal{E}_1 = \{(e_1, q_2) \in \mathcal{E}^{\mathcal{A}} \times \mathcal{Q}^{\mathcal{B}} \mid \mathfrak{e}^{\mathcal{A}}(e_1) \notin \Sigma^{\mathcal{B}}\}$
  $\mathcal{E}_2 = \{(q_1, e_2) \in \mathcal{Q}^{\mathcal{A}} \times \mathcal{E}^{\mathcal{B}} \mid \mathfrak{e}^{\mathcal{B}}(e_2) \notin \Sigma^{\mathcal{A}}\}$
  $\mathcal{E}_3 = \{(e_1, e_2) \in \mathcal{E}^{\mathcal{A}} \times \mathcal{E}^{\mathcal{B}} \mid \mathfrak{e}^{\mathcal{A}}(e_1) = \mathfrak{e}^{\mathcal{B}}(e_2)$
- $\mathcal{X} = \mathcal{X}^{\mathcal{A}} \times \mathcal{X}^{\mathcal{B}}$
- $\mathcal{I} = \{((q_1, q_2), (x_1, x_2)) \in \mathcal{Q} \times \mathcal{X} \mid (q_1, x_1) \in \mathcal{I}^{\mathcal{A}} \wedge (q_2, x_2) \in \mathcal{I}^{\mathcal{B}}\}$
- $\mathcal{S} = \{((q_1, q_2), (x_1, x_2)) \in \mathcal{Q} \times \mathcal{X} \mid (q_1, x_1) \in \mathcal{S}^{\mathcal{A}} \wedge (q_2, x_2) \in \mathcal{S}^{\mathcal{B}}\}$
- $\mathfrak{s}(c_1, c_2) = \begin{cases} (\mathfrak{s}^{\mathcal{A}}(c_1), c_2) & \text{if } c_2 \in \mathcal{Q}^{\mathcal{B}} \\ (c_1, \mathfrak{s}^{\mathcal{B}}(c_2)) & \text{if } c_1 \in \mathcal{Q}^{\mathcal{A}} \\ (\mathfrak{s}^{\mathcal{A}}(c_1), \mathfrak{s}^{\mathcal{B}}(c_2)) & \text{otherwise} \end{cases}$
- $\mathfrak{d}(c_1, c_2) = \begin{cases} (\mathfrak{d}^{\mathcal{A}}(c_1), c_2) & \text{if } c_2 \in \mathcal{Q}^{\mathcal{B}} \\ (c_1, \mathfrak{d}^{\mathcal{B}}(c_2)) & \text{if } c_1 \in \mathcal{Q}^{\mathcal{A}} \\ (\mathfrak{d}^{\mathcal{A}}(c_1), \mathfrak{d}^{\mathcal{B}}(c_2)) & \text{otherwise} \end{cases}$
- $\mathcal{R} = \{((x_1, x_2), (c_1, c_2), (y_1, y_2)) \in \mathcal{X} \times \mathcal{E} \times \mathcal{X} \mid$
  $((c_1 \in \mathcal{Q}^{\mathcal{A}} \wedge x_1 = y_1)$ or $(c_1 \in \mathcal{E}^{\mathcal{A}} \wedge (x_1, c_1, y_1) \in \mathcal{R}^{\mathcal{A}}))$ and
  $((c_2 \in \mathcal{Q}^{\mathcal{B}} \wedge x_2 = y_2)$ or $(c_2 \in \mathcal{E}^{\mathcal{B}} \wedge (x_2, c_2, y_2) \in \mathcal{R}^{\mathcal{B}}))\}$
- $\Phi((q_1, q_2), (x_1, x_2), r) = \Phi^{\mathcal{A}}(q_1, x_1, r) \times \Phi^{\mathcal{B}}(q_2, x_2, r)$.

$\mathcal{A} \otimes \mathcal{B}$ is annotated as follows:

- $\Sigma = \Sigma^{\mathcal{A}} \cup \Sigma^{\mathcal{B}}$
- $\Gamma = \Gamma^{\mathcal{A}} \cap \Gamma^{\mathcal{B}}$
- $\mathfrak{e}(c_1, c_2) = \begin{cases} \mathfrak{e}^{\mathcal{A}}(c_1) & \text{if } c_1 \in \mathcal{E}^{\mathcal{A}} \\ \mathfrak{e}^{\mathcal{B}}(c_2) & \text{otherwise} \end{cases}$
- $\mathfrak{c}(q_1, q_2) = \mathfrak{c}^{\mathcal{A}}(q_1) \cap \mathfrak{c}^{\mathcal{B}}(q_2)$.

The second stage in the formation of $\mathcal{A} \parallel \mathcal{B}$ is to discard all discrete states $q \in \mathcal{Q}^{\mathcal{A}\otimes\mathcal{B}}$ such that $\mathfrak{c}^{\mathcal{A}\otimes\mathcal{B}}(q) = \emptyset$. This ensures that $\mathcal{A} \parallel \mathcal{B}$ is an annotated hybrid automaton and completes the construction.

Remark 2. Parallel composition is commutative and associative (up to isomorphism). Therefore we can, and will, refer to the parallel composition of several annotated hybrid automata without fear of ambiguity.

The concept of parallel composition defined here is nowise new. Conditions are just an alternative to the *propositional constraints* that commonly arise in the temporal logic literature. The novelty is not in our definition, but in the use we will make of parallel composition—to define a new class of hybrid automata for which the reachability problem is decidable. The following simple relationship between language equivalence and parallel composition will be a key ingredient; it will allow us to do reductions component-wise.

Lemma 1. *If* $\mathcal{A} \sim_{\mathfrak{le}} \mathcal{A}'$ *and* $\mathcal{B} \sim_{\mathfrak{le}} \mathcal{B}'$ *then* $\mathcal{A} \parallel \mathcal{B} \sim_{\mathfrak{le}} \mathcal{A}' \parallel \mathcal{B}'$.

2 A New Decidable Class

Definition 9. If $\mathcal{K}$ is a class of hybrid automata, then the *parallel closure* $\mathcal{K}^{\parallel}$ is the class of all parallel compositions of all annotations of the elements from $\mathcal{K}$.

Definition 10.
Clock Components:
Let $\mathcal{C}$ be the class of 1-dimensional SLHA such that $\Phi(q, t, x) = x + t$, the plant state is zero in all initial states, and each edge satisfies either:
(a) zero reset
or (b) identity reset

Rectangular Components:
Let $\mathcal{R}$ be the class of 1-dimensional SLHA such that $\Phi(q, t, x) = x + tI_q$, where I_q is an interval for each q, and such that each edge satisfies either:
(a) constant set-valued reset map
or (b) identity reset and
source and destination have the same flow

Deterministic Components:
Let $\mathcal{D}$ be the class of SLHA with deterministic flows and finite initial set such that each edge satisfies either:
(a) constant (single-valued) reset map
or (b) identity reset and
source and destination have the same flow

Nondeterministic Components:
Let $\mathcal{N}$ be the class of SLHA such that each edge satisfies either:
(a) constant set-valued reset map
or (b) identity reset and
trivial guard and
source and destination have the same flow and invariant set

The reader is probably already familiar with $\mathcal{C}^{\|}$ and $\mathcal{R}^{\|}$, though our presentation is somewhat unusual. They are, respectively, *timed automata* [1] and *initialized rectangular hybrid automata* [12, 7]. Both of these classes are known to have decidable reachability problems.

Lemma 2.

1. *If $\mathcal{A} \in \mathcal{R}$ then every annotation of $\mathcal{A}$ is language equivalent to a two clock timed automaton.*
2. *If $\mathcal{A} \in \mathcal{D} \cup \mathcal{N}$ then every annotation of $\mathcal{A}$ is language equivalent to an annotation of a clock component.*

Part (1) is contained in [12] while Part (2) offers no real difficulty. Combining Lemma 1 with Lemma 2 and the decidability of reachability for timed automata, the following theorem is immediate.

Theorem 1. *Reachability is decidable for $(\mathcal{R} \cup \mathcal{D} \cup \mathcal{N})^{\|}$.*

Note that $(\mathcal{R} \cup \mathcal{D} \cup \mathcal{N})^{\|}$ is a proper superclass of the initialized rectangular hybrid automata, and that the possibility of further extension remains open. New building blocks may be added easily; they will slip right into place, as long as they are language equivalent to timed automata. Admittedly, this is a severe restriction.

3 Irrational Timed Automata

The semi-algebraic sets share many of the nice properties of the semi-linear sets [14]; in particular, they are closed under projection [13] and the boolean operations. So it is natural to ask if the results of the preceding section remain true in this more general context.

Definition 11. *We use $\mathcal{C}_{SA}$, $\mathcal{R}_{SA}$, $\mathcal{D}_{SA}$ and $\mathcal{N}_{SA}$ for the generalizations of $\mathcal{C}$, $\mathcal{R}$, $\mathcal{D}$ and $\mathcal{N}$ to semi-algebraic hybrid automata.*

As before, we can prove that every automaton $\mathcal{A} \in (\mathcal{R}_{SA} \cup \mathcal{D}_{SA} \cup \mathcal{N}_{SA})^{\|}$ is language equivalent to an automaton $\mathcal{A}' \in \mathcal{C}_{SA}^{\|}$. But note that $\mathcal{A}'$ is *not* necessarily a timed automaton; its constants are arbitrary algebraic numbers and may be irrational. So we are led to ask if reachability remains decidable for *algebraic timed automata*. Unfortunately, it does not.

Theorem 2. *Reachability is undecidable for $\mathcal{C}_{SA}^{\|}$.*

Before preceding with a proof of this theorem, there is further motivation. Reachability is decidable for several classes of hybrid systems, for example [8] and [9]; we focus on two. We have already mentioned the initialized rectangular hybrid automata, and even offered a modest generalization. The second class contains the semi-algebraically defined hybrid automata with constant (set-valued)

reset maps, which are proven to have computable finite bisimulations in [10]. To what extent can these classes be combined while preserving the decidability of reachability? Algebraic timed automata represent, in our opinion, a simple midpoint between these two classes, and in this light, the undecidability of the reachability problem presents an obstacle to a natural unification.

3.1 Minsky Machines and Undecidability

We prove our main theorem in more generality to illustrate that undecidability does not arise from some subtle property of the algebraics. Rather, it is a consequence of irrationality. This generality will also be useful in Sect. 4.

Definition 12. Given $\mathcal{S} \subseteq \mathbb{R}$, the class $\mathcal{T}_\mathcal{S}$ of *irrational timed automata over* $\mathcal{S}$ is the generalization of timed automata in which the guards and state invariants are allowed to have constants from $\mathbb{Q} \cup \mathcal{S}$.

In particular, $\mathcal{C}^{\|}_{SA} = \mathcal{T}_\mathbb{A}$ is the class of *algebraic timed automata*, where $\mathbb{A}$ is the set of all algebraic numbers, i.e. real roots of polynomial equations with rational coefficients.

Theorem 3. *Let $\tau \in (1,2)$ be irrational. Let $\mathcal{S} = \{0, 1, \tau, 3-\tau\}$. Then the reachability problem for the class $\mathcal{T}_\mathcal{S}$ is undecidable.*

Our proof of undecidability closely follows the technique in [7], where the undecidability of several slight generalizations of timed automata is proved. In particular, we proceed by reducing the halting problem for two-counter Minsky machines to the reachability problem for the class $\mathcal{T}_\mathcal{S}$. Before presenting this reduction, we give a definition of two-counter machines. It is well known that the halting problem for two-counter machines is undecidable [11].

Definition 13. A *two-counter Minsky machine* is finite state machine with two natural number counters c_1 and c_2. Each machine state has an associated command which is executed when the machine is in that state. Possible commands are:

- increment c_i and go to n
- decrement c_i and go to n; if $c_i = 0$ then it is unchanged
- if c_i is zero go to n, otherwise go to m
- halt

where $i \in \{1, 2\}$ and n, m are machine states. There is a distinguished start state and the machine begins its execution with both counters set to zero.

Proof (Theorem 3).

Let $\mathcal{A}$ be a two-counter machine. To simplify the encoding we can assume that it never decrements a counter containing zero. Of course, any two-counter machine can easily be modified to meet this restriction.

Let $\langle x \rangle$ denote the non-integer part of $x \in \mathbb{R}$. In particular, for any x, $0 \leq \langle x \rangle < 1$. We will encode the values of the counters c_1 and c_2 in continuous

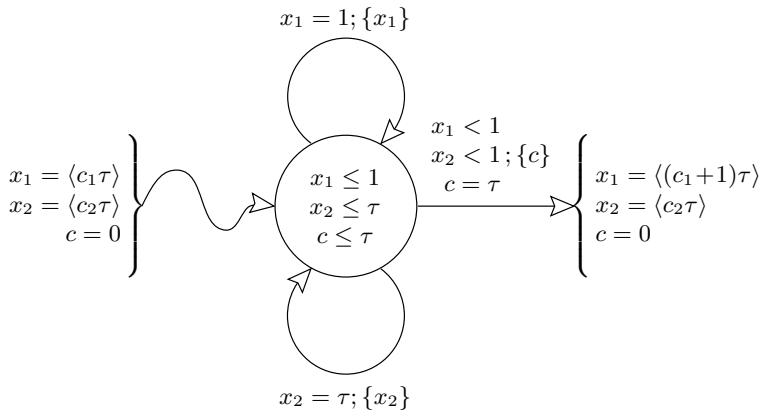

Fig. 1. Increment c_1

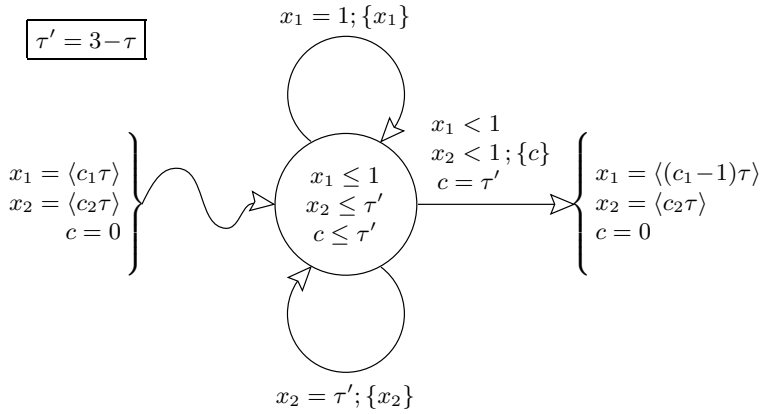

Fig. 2. Decrement c_1

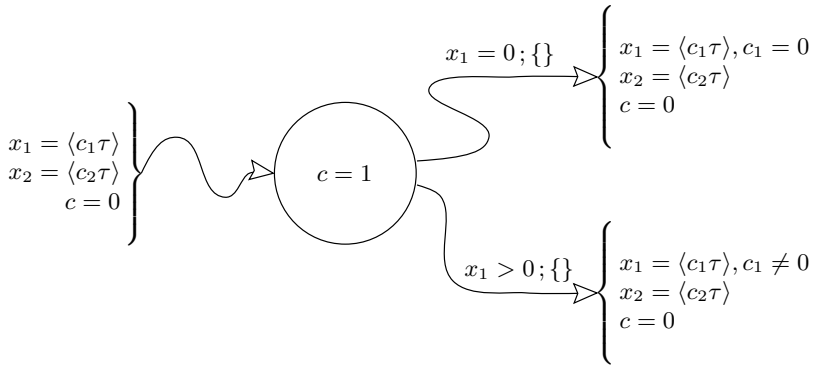

Fig. 3. Test if $c_1 = 0$

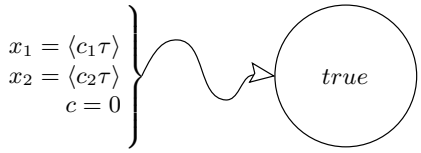

Fig. 4. Halt

variables x_1 and x_2 by representing the natural number n by the real number $\langle n\tau \rangle$. Because τ is irrational, $\langle n\tau \rangle = \langle m\tau \rangle$ if and only if $n = m$.

We now construct a timed automaton $\mathcal{A}^* \in \mathcal{T}_\mathcal{S}$. It will have three clocks components. We represent the continuous state of these components by x_1, x_2 and c. As indicated, x_1 an x_2 store the counter values.

In the construction of $\mathcal{A}^*$, each state of $\mathcal{A}$ is replaced with one of the four gadgets illustrated in Figs. 1–4, depending on its associated command. For example, a state with command "Increment c_2" would be replaced by the gadget in Fig. 1, but with the roles of x_1 and x_2 reversed. In the figures, a state q is represented by a node labeled with the state invariant $\mathcal{I}_q$. An edge e is represented by an arrow from the node for $\mathfrak{s}(e)$ to the node for $\mathfrak{d}(e)$ labeled by both the guard for e and by the set of clocks reset to zero by the transition.

We define the destination of edges leaving a gadget to correspond to the transitions in the two-state machine $\mathcal{A}$. Finally, let $\mathcal{S} = (q_0, \mathbf{0})$, where q_0 is the discrete state in the gadget corresponding to the initial state of $\mathcal{A}$. This completely specifies a timed automaton $\mathcal{A}^*$.

The reader is encouraged to carefully examine Figs. 1–4 to understand why the gadgets that they depict have the asserted effects. It is worth noting that Fig. 2 is the same as Fig. 1 except that τ is replaced by $\tau' = 3-\tau \in (1, 2)$. It is also worth noting that the each gadget is defined to guarantee that $c = 0$ when the next gadget is entered.

By construction, the two-counter machine $\mathcal{A}$ halts if and only if there is a reachable state of $\mathcal{A}^*$ which corresponds to a halting state of $\mathcal{A}$. As mentioned above, the halting problem for two-counter machines is undecidable. This proves that reachability is undecidable for the class $\mathcal{T}_\mathcal{S}$. □

Theorem 2 is proved by letting $\tau = \sqrt{2}$ in Theorem 3 and noting that $\mathcal{T}_{\{0,1,\sqrt{2},3-\sqrt{2}\}} \subseteq \mathcal{C}^{\|}_{SA}$.

3.2 Further Results

Both $\mathcal{R}_{SA}$ and $\mathcal{D}_{SA}$ are extensions of $\mathcal{C}_{SA}$. Therefore, the undecidability of reachability for $\mathcal{R}^{\|}_{SA}$ and $\mathcal{D}^{\|}_{SA}$ follows from Theorem 2. On the other hand, $\mathcal{N}^{\|}_{SA}$ requires a different proof. The following results are each proved by refining of the construction for Theorem 3. The gadgets become rather complicated to circumvent the additional restrictions, but no other change is necessary.

Theorem 4.

1. *Reachability is undecidable for* $\mathcal{N}_{SA}^{\|}$
2. *Let* $\tau > 0$ *be irrational. Then reachability is undecidable for* $\mathcal{T}_{\{\tau,1\}}$ *(with as few as three clocks).*

Definition 14. Given $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathbb{R}$, let $\mathcal{T}_{\mathcal{S}_1,\mathcal{S}_2}$ be the class of irrational timed automata with the first clock constrained by constants from $\mathcal{S}_1$ and the remaining clocks constrained by constants from $\mathcal{S}_2$.

Theorem 5. *Let* $\tau > 0$ *be irrational. Reachability is undecidable for* $\mathcal{T}_{\{\tau\},\{1\}}$ *(with as few as four clocks).*

Before moving on, one simple decidability result should be mentioned.

Theorem 6. *Reachability for one-clock timed automata over* $\mathbb{R}$ *depends only on the order of the constants (including zero), and is decidable given that order.*

4 Parametric Timed Automata

Without significant modification, the undecidability results of the previous section carry over to the context of *parametric timed automata*. These automata—introduced in "Parametric Real-time Reasoning" [2]—allow us to express a more sophisticated range of synthesis and verification questions, but their most basic properties turn out to be undecidable [2]. After discussing the connection between parametric timed automata and timed automata with irrational constraints, we state a new undecidability result and then examine the complexity of the one-clock case, for which reachability *is* decidable.

Definition 15.

(a) *Parametric timed automata* are a generalization of timed automata in which the guards and state invariants are allowed to have constants from $\mathbb{Q} \cup \Psi$, where Ψ is a set of parameter variables.
(b) Let $\mathcal{A}$ be a parametric timed automaton with parameters from Ψ and let $\lambda : \Psi \to \mathbb{Q}$. Then $\mathcal{A}_\lambda$ is the timed automaton that results from using λ to substitute for the parameters in $\mathcal{A}$.
(c) If q is a state of $\mathcal{A}$, then $\Gamma_q(\mathcal{A})$ is the subset of parameter space for which $\mathcal{A}$ has a run reaching q. In other words:

$$\Gamma_q(\mathcal{A}) = \{\lambda : \Psi \to \mathbb{Q} \mid q \text{ is a reachable state of } \mathcal{A}_\lambda\}.$$

Now consider what would happen were τ a rational number in the proof of Theorem 3. In particular, let $\tau = a/b \in (1, 2)$, where a and b are relatively prime. As long as our virtual Minsky machine keeps its counter values below b, nothing can go wrong. But a counter value of b is indistinguishable from zero; we have an overflow error. Such an error is easy to detect if we always test for zero after incrementing a counter. Thus, we can correctly simulate the Minsky machine as

long as the counter values remain small and suspend the simulation when an overflow error is detected.

At the risk of stating the obvious, note that if a Minsky machine halts then its counters remain bounded. Also note that the rational numbers in the interval $(1, 2)$ have arbitrarily large denominators (in reduced form). Therefore, a Minsky machine halts if and only if that fact is detected by the simulation for *some* rational $\tau \in (1, 2)$. With a few simple details swept under the rug, this is all it takes to translate the theorems of the last section into theorems about parametric timed automata. Under this translation, Theorem 4.2 becomes:

Theorem 7. *The emptiness of $\Gamma_q(\mathcal{A})$ is undecidable for the class of parametric timed automata with three clocks and one parameter.*

This is not an essentially new result. That the emptiness of $\Gamma_q(\mathcal{A})$ for parametric timed automata is undecidable was proved in [2]. The proof given there uses three clocks and six parameters but has the advantage of working for both the dense-time and discrete-time cases. The translation of Theorem 5 is more interesting.

Theorem 8. *The emptiness of $\Gamma_q(\mathcal{A})$ is undecidable for the class of parametric timed automata with only one clock constrained by parameters.*

The corresponding problem is decidable for discrete-time [2], so we have exposed a divergence in the expressive power of dense-time and discrete-time parametric timed automata.

Before leaving the subject of parametric timed automata, let us turn our attention to the one-clock case. Let $\mathcal{A}$ be a one-clock parametric timed automaton. As was noted before, the time-abstract runs of a one-clock timed automaton depend only on the order of the constants. So, to calculate $\Gamma_q(\mathcal{A})$, we simply determine the reachability of q in $\mathcal{A}_\lambda$ for sample assignments λ corresponding to every possible ordering of the constants (including zero) and parameters. Unfortunately, the number of such orderings[1] grows exponentially in the number of parameters.

[1] Let π_m^n be the number of (non-strict) orderings that can be formed from n parameters with respect to m distinct constants. The following formulae generalizations of those in [6], where the sequence $\{\pi_0^n\}_{n\in\mathbb{N}}$ of *preferential arrangements* is studied.

$$\pi_m^n = \frac{1}{2}\sum_{k=0}^{\infty}\binom{k}{m}k^n 2^{-k}$$

$$\pi_m^n = 2m\pi_m^{n-1} + \sum_{k=0}^{n-1}\binom{n}{k}\pi_m^k$$

Finally, writing $f \sim_n g$ to denote $\lim_{n\to\infty} f/g = 1$,

$$(\forall m) \quad \pi_m^n \sim_n \frac{(n+m)!}{2m!\ln^{n+m+1}2}$$

$$(\forall n) \quad \pi_m^n \sim_m (2m)^n.$$

We conclude with a number of observations about the complexity of the problem of determining emptiness for one-clock parametric timed automata. Both standard algorithmic complexity and parametric complexity [5] results are considered.

Theorem 9. *Consider the problem of determining the non-emptiness of* $\Gamma_q(\mathcal{A})$ *for the class of one-clock parametric timed automata.*

1. *The problem is NP-complete.*
2. *For any fixed k bounding the number of parameters, the problem can be solved in polynomial time.*
3. *Parameterized by the number of parameters, the problem is W[SAT]-hard. Note that it is strongly suspected that W[SAT] is a proper superset of the fixed parameter tractable (FPT) problems* [5].
4. *Parameterized by the number of both constants and parameters, the problem is FPT.*

5 The Complexity of Questions About SLHA

This last section deviates from the course of the paper thus far. The only connection it bears to the earlier sections is the reliance on Minsky machine simulations; they play a central role in proving the hardness directions of all of the completeness results that follow.

Definition 16. A *maximal* run is any run that can not be extended. It either has an infinite number of transitions, ends with an infinite time interval spent in the same discrete state, or reaches a state from which it can neither flow nor jump. A maximal run is said to be *jump-infinite* if it makes an infinite number of discrete transitions; otherwise *jump-finite*. It is *time-infinite* if its time intervals sum to infinity; otherwise *time-finite*. A maximal run is *blocking* if it is both time-finite and jump-finite; otherwise we call it *infinite*. Finally, a maximal run is *Zeno* if it is time-finite but jump-infinite.

A hybrid automaton is said to have *arbitrarily long runs* if for each $n \in \mathbb{N}$ there is either a run that makes at least n transitions or a run with a duration of at least n.

Definition 17. An SLHA is *compact* if all of its defining regions are compact. An SLHA with at most one initial state and at most one possible evolution from each state is called *deterministic*, and an SLHA with at least one initial state and at least one evolution from each state is called *non-blocking*.

Table 1 gives, for different classes of SLHA, the complexity of determining whether certain types of runs exist. It is only a sampling of complexity results. Further questions might prove interesting; for example, "Is there an time-infinite run for every initial state?"

Table 1. The complexity of detecting various types of runs

<table>
<tr><th>type
of run</th><th>Semi-linear
Hybrid
Automata
(SLHA)</th><th>Compact
SLHA</th><th>Deterministic
SLHA</th><th>Deterministic
and
Non-blocking
SLHA</th><th>Non-blocking
SLHA</th></tr>
<tr><td>Zeno</td><td rowspan="6">Σ_1^1-complete</td><td colspan="2" rowspan="2">Σ_2-complete</td><td rowspan="6">Σ_1^1-complete</td></tr>
<tr><td>time-finite</td></tr>
<tr><td>non-Zeno</td><td colspan="2" rowspan="3">Π_2-complete</td></tr>
<tr><td>time-infinite,
jump-infinite</td></tr>
<tr><td>time-infinite</td></tr>
<tr><td>jump-infinite</td><td colspan="3" rowspan="2">Π_1-complete</td></tr>
<tr><td>infinite</td></tr>
<tr><td>arbitrarily
long</td><td colspan="3">always</td></tr>
<tr><td>arbitrarily
long blocking</td><td>Π_2-complete</td><td colspan="3" rowspan="2">never</td></tr>
<tr><td>blocking</td><td colspan="3" rowspan="3">Σ_1-complete</td></tr>
<tr><td>time-infinite,
jump-finite</td></tr>
<tr><td>jump-finite</td></tr>
</table>

Table 2. Supplement to Table 1

	Semi-linear Hybrid Automata (SLHA)	Weakly Deterministic SLHA	Deterministic SLHA
Zeno run	Σ_1^1-complete		Σ_2-complete
time-infinite run for every initial state	Π_2^1-complete	Π_1^1-complete	Π_2-complete

Also, our definition of determinism is very restrictive. A more reasonable property is that there is at most one evolution from each state, with no restriction made on the initial set. We call this property *weak determinism*. Table 2 shows that questions may be much harder for weakly deterministic SLHA than for deterministic SLHA. The complexity of many questions matches that of general SLHA, but this is not always the case.

As a closing note, the Zeno phenomenon is exploited in [3] and [4] to show that the reachability problem for dynamical systems with piecewise constant derivatives is arithmetic and hyper-arithmetic, respectively.

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoret. Comput. Sci.*, 126(2):183–235, 1994.
2. Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 592–601, San Diego, California, 16–18 May 1993.
3. Eugene Asarin and Oded Maler. Achilles and the tortoise climbing up the arithmetical hierarchy. *Journal of Computer and System Sciences*, 57(3):389–398, December 1998.
4. Olivier Bournez. Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science*, 210(1):21–71, January 1999.
5. R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer-Verlag, New York, 1999.
6. O. A. Gross. Preferential arrangements. *Amer. Math. Monthly*, 69:4–8, 1962.
7. Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *J. Comput. System Sci.*, 57(1):94–124, 1998. 27th Annual ACM Symposium on the Theory of Computing (STOC'95) (Las Vegas, NV).
8. Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: a class of decidable hybrid systems. In *Hybrid systems*, pages 179–208. Springer, Berlin, 1993.
9. M. Kourjanski and P. Varaiya. A class of rectangular hybrid systems with computable reach set. *Lecture Notes in Computer Science*, 1273:228–234, 1997.
10. G. Lafferriere, G. J. Pappas, and S. Sastry. O-minimal hybrid systems. Technical Report UCB/ERL M98/29, Department of Electrical Engineering and Computer Science, University of California at Berkeley, May 1998.
11. Marvin L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *Ann. of Math. (2)*, 74:437–455, 1961.
12. Anuj Puri and Pravin Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In *Computer aided verification (Stanford, CA, 1994)*, pages 95–104. Springer, Berlin, 1994.
13. Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. RAND Corporation, Santa Monica, Calif., 1948.
14. Lou van den Dries. *Tame topology and o-minimal structures*. Cambridge University Press, Cambridge, 1998.