Decidability of Emptiness Checking in various Alternating Timed Automata and Connections to Timed Logics

B. Tech. Project Stage II Report

Submitted in partial fulfillment of the requirements
for the degree of

Bachelor of Technology

by

Garvit Juniwal
Roll No: 08005008

under the guidance of

Prof. Krishna S
Prof. Paritosh Pandya

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai
## Contents

1 Introduction 1

2 Hybrid Automata 1
   2.1 Timed Automata 2
   2.2 Rectangular Hybrid Automata 3

3 Extensions on Rectangular Hybrid Automata 4
   3.1 One Variable 4
   3.2 Two Variables 5
   3.3 Three Variables 5

4 Alternating Timed Automata 7
   4.1 Decidability of Language Emptiness 8

5 Alternation and Stopwatches 10
   5.1 Language Emptiness 11
   5.2 Language Emptiness Decidability of One Variable Initialized Stopwatch ATA 11
   5.3 Language Emptiness Undecidability of One Variable Stopwatch ATA 11

6 Snoopy ATAs and Connections to TPTL 14
   6.1 Decidability of One Variable Snoopy ATA 15
   6.2 Snoopiness adds expressive power 16
   6.3 TPTL 16
      6.3.1 One Variable TPTL 17
      6.3.2 Multiple Variable $TPTL_F^+$ 18

7 Summary 20

8 Future Work 20
Abstract

In the initial parts of this thesis, we start by approaching the problem of decidability of emptiness of one variable alternating stopwatch automata. We start off by presenting an overview of important results in the field of hybrid systems and useful subclasses and their proofs. Then we go on to alternating timed automata and redo the previously known result that one clock ATA has decidable language emptiness.

In the later and more interesting part of the thesis, we then show that language emptiness for one stopwatch ATA is undecidable. We also investigate some connections with timed logics and show that one variable TPTL can be embedded into one clock ATA. We then define a new class of automata called snoopy ATA and show that it is strictly more expressive than one clock ATA and still has decidable language emptiness. We also claim that multiple variable $TPTL^+_F$ can be embedded into one clock snoopy ATA. But a rigorous proof is not yet developed.
1 Introduction

Timed Automata, introduced by Alur and Dill are an accepted formalism in modelling timed systems. The emptiness problem for timed automata is decidable - this property has paved the way for building tools such as UPPAAL and KRONOS to model check timed systems. However, timed automata are not closed under complementation; moreover their universality and inclusion problems are undecidable. This makes it difficult to use timed automata as specification mechanisms. A way to get around this issue is to use timed automata with alternation. Alternating Timed Automata (ATA) are a powerful extension of Alur-Dill timed automata, and are closed under all boolean operations. ATA have played a key role in providing verification algorithms for prominent specification mechanisms such as Metric Temporal Logic (MTL). The decidability of the emptiness of 1-clock ATA over finite words has been used to prove the decidability of pointwise MTL over finite words.

In this thesis we show that language emptiness problem for ATA with a single stopwatch variable is undecidable. Hence eliminating the possibility of the use of these automaton to prove decidability of the logic MTL with durations. Since ATAs with a stopwatch turns out to be too powerful, we look for less powerful versions of ATA which can capture logics with more expressive power than pointwise MTL over finite words. We define a new notion of snoopy ATAs with one clock variable which have the added capability of across-state clock valu comparisons and show that a fragment of pointwise TPTL over finite words can be embedded into these automata. We also have an auxiliary result that one variable pointwise TPTL over finite words can be embedded into one clock ATA.

2 Hybrid Automata

In this section we define a general class of automata which we will later restrict in various different ways to present different results.

Definition A general hybrid automata is a tuple \((L, l_0, C, \Sigma, \Phi(C), U(C), \delta, F, Act)\), where

- \(L\) is a set of locations.
- \(l_0\) is the initial location.
- \(C\) is a set of real valued variables.
- \(\Sigma\) is the vocabulary
- \(\Phi(C)\) is the set of constraints over \(C\), which can take various forms depending on the type of automaton considered.
- \(U(C)\) is the set of updates over \(C\), which again change in various kinds of automata.
- \(\delta\) is the transition relation. \(\delta \subseteq L \times \Sigma \times \Phi(C) \times L \times U(C)\)
- \(F\) is the set of final states.
- \(Act\) is the activity function. For \(l \in L\) and \(x \in C\), \(Act[l][x]\) denotes the derivative of the variable \(x\) in location \(l\). The most general form that this derivative takes in all the automatons we consider is \([u_1, u_2]\), which means the derivative lies in the compact bounded interval \([u_1, u_2]\) of \(\mathbb{Z}\).
From any given location-valuation pair, there are two kinds of transitions possible:

- A jump \((l, v) \xrightarrow{a} (l', v')\) if \(\exists (l, \gamma, a, \alpha, l') \in \delta \) s.t. \(\gamma(v) = \text{true}\) and \(\alpha(v) = v'\).
- A time elapse \((l, v) \xrightarrow{t} (l, v')\) if \(\exists d \in \text{Act}[l] \) s.t. \(v' = v + d \cdot t\).

This naturally defines a transition system \(\mathcal{T}\) with states as the set of all location valuation pairs i.e. \(L \times \mathbb{R}^C\) and transition relation given by \(\to\) above.

A timed word over \(\Sigma\) is a sequence
\[
(w_1, t_1), (w_2, t_2), \ldots, (w_k, t_k)
\]
of pairs from \(L \times \mathbb{R}_+\). A timed word \(w\) is in the language \(L(A)\) of an automaton \(A\) iff there exists a sequence of transitions
\[
(l_0, v_0) \xrightarrow{a_1} (l_1, v_1) \xrightarrow{a_2} (l_2, v_2) \ldots \xrightarrow{a_k} (l_k, v_k)
\]
where \(v_0 = 0\), usually and \(l_k \in F\), the set of final states. We start our discussion with the most basic form of hybrid automata and then move on to more complex forms towards our problem.

2.1 Timed Automata

In the traditional timed automata model, first introduced by Alur-Dill, the constraint set \(\Phi(C)\) is a set of constraints \(\sigma\) defined by
\[
\sigma := x \sim c | \sigma_1 \land \sigma_2 | \neg \sigma, \sim \in \{\leq, <, \geq, >, =\}, c \in \mathbb{Z}_+, x \in C
\]
The updates are just resets to zero like \(x := 0\) and the derivative of each variable (often denoted by \(\dot{x}\)) in every location is 1 or \(\text{Act}[l][x] = 1\) \(\forall l \in Q, x \in C\).

**Theorem 2.1** Emptiness checking for A-D timed automata is decidable. [AD94]

**Proof** We present the technique of this proof to introduce the concept of a bisimulation which is used profusely to prove many other results.

**Definition** For a transition system \(\mathcal{T} = (Q, \to)\) of any automaton, a equivalence relation \(\approx\) of \(Q\) is a bisimulation if for all \(q_1, q_2 \in Q\), \(q_1 \approx q_2\),

- whenever \(q_1 \xrightarrow{a} q_1'\) for some \(a \in \Sigma\), there exists \(q_2' \in Q\) such that \(q_2 \xrightarrow{a} q_2'\) and \(q_2 \approx q_2'\)
- whenever \(q_1 \xrightarrow{t} q_1'\) for some \(t \in \mathbb{R}_+\), there exists \(q_2' \in Q\) such that \(q_2 \xrightarrow{t} q_2'\) and \(q_2 \approx q_2'\)

A bisimulation is finite if it has finite number of equivalence classes. Given some \(R \subseteq Q\), notion of successor of defined as
\[
\text{Succ}(R) = \{q \in Q | \exists q' \in R, q' \rightarrow q\}
\]
A property of a bisimulation is that for every equivalence class \(R\) of \(\approx\), the successor \(\text{Succ}(R)\) of \(R\) is another unique equivalence class of \(\approx\).

For timed automata, finite bisimulations can be constructed by looking at appropriate partition of the the space of all clock valuations. Consider a partitioning relation \(\simeq\) for the space \(\mathbb{R}^C\). Define an equivalence relation on the location-clock valuation pairs in the manner that
\[(l, v) \simeq (l', v')\] iff
\begin{itemize}
  \item $l = l'$
  \item $v \simeq v'$
\end{itemize}

Let $\phi$ be any clock constraint from the set $\Phi(C)$ and $\sigma$ be an update from the set $\mathcal{U}(C)$. The above equivalence relation defined using $\simeq$ is a finite bisimulation iff

1. (Consistency with constraints) If $v_1 \simeq v_2$ then $\phi(v_1) \iff \phi(v_2)$
2. (Consistency with updates) If $v_1 \simeq v_2$ then $\sigma(v_1) \simeq \sigma(v_2)$
3. (Consistency with time-elapse) If $v_1 \simeq v_2$ and $(l, v_1) \xrightarrow{i} (l, v_1')$ then there exists a $l'$ such that $(l, v_2) \xrightarrow{i} (l, v_2')$ and $v_1' \simeq v_2'$
4. (Consistency with jump) If $v_1 \simeq v_2$ and $(l_1, v_1) \xrightarrow{a} (l_2, v_1')$ then there exists a $v_2'$ such that $(l_1, v_2) \xrightarrow{a} (l_2, v_2')$ and $v_1' \simeq v_2'$

[BDFP04] In this case the partition induced by $\simeq$ is called a set of regions.

Consider and timed automata $\mathcal{A}$ with $c_x$ denote the value of maximum integer constant to which the clock variable $x$ is compared to. $k$ is $|C|$. Then, $v \simeq_{AD} v'$ iff

- For all $1 \leq i \leq k$, either $int(v(x_i)) = int(v'(x_i))$ or both exceed $c_{x_i}$.
- For all $1 \leq i, j \leq k$ such that $v(x_i) \leq c_x$, and $v(x_j) \leq c_x$, $fr(v(x_i)) \leq fr(v(x_j))$ iff $fr(v'(x_i)) \leq fr(v'(x_j))$
- For all $1 \leq i \leq k$, such that $v(x_i) \leq c_x$, $fr(v(x_i)) = 0$ iff $fr(v'(x_i)) = 0$

It can be easily seen that $\simeq_{AD}$ induces a set of regions and hence a finite bisimulation of the automaton $\mathcal{A}$. A few other proofs in the report rely on this technique of region construction.

### 2.2 Rectangular Hybrid Automata

**Definition** A region $R \subset \mathbb{R}^n$ is called a rectangular region if it is a cartesian product of intervals(possibly unbounded) whose finite endpoints are integers(rational constants can be used and do not enhance expressiveness). An automaton $\mathcal{A}$ is called a rectangular hybrid automaton if $\Phi(C)$ and $\mathcal{U}(C)$ consist of constraints and updates(possibly non-deterministic) resp. which induce rectangular regions in the space $\mathbb{R}^C$. Also, for all $l \in L$ and $x \in C$, $Act[l][x]$ takes the form $[u_1, u_2]$, $u_1, u_2 \in \mathbb{Z}$. We will use the shorthand $\dot{x}$ from now on.

**Definition** A variable $x$ is called a singular variable if it is such that for every location $l$, $\dot{x} = k, k \in \mathbb{Z}$. An automaton where all the variables are singular variables are called singular automata.

**Definition** A variable $x$ is called a stopwatch if it is such that for every location $l$, $\dot{x}$ is either 0 or 1. An automaton where all the variables are stopwatches and all the updates are deterministic(i.e. of the form $x := c$ for some constant integer $c$) is called a stopwatch automata.

**Definition** A rectangular hybrid automata is said to be initialized if for every transition $(l, \gamma, a, \alpha, l') \in \delta$, if for a variable $x$ we have that $Act[l][x] \neq Act[l'][x]$, then that variable is updated by the update $\alpha$. 

3
**Theorem 2.2** Every initialized rectangular hybrid automata can be transformed into another timed automata. Hence the expressive powers of initialized rectangular hybrid automata is same as timed automata. As a direct corollary, the emptiness checking problem from initialized rectangular hybrid automata is decidable. [HKPV95]

**Proof** Proof of this theorem is not important to this report.

**Theorem 2.3** The emptiness checking problem for an uninitialized stopwatch automata with one variable acting as a stopwatch and 5 other variables acting as clocks is undecidable. [HKPV95]

**Proof** The proof of this is given by showing a reduction of the halting problem of two-counter machines to the emptiness checking problem of an uninitialized stopwatch automata with 5 clocks and 1 stopwatch.

Note: It can also be shown that deterministic updates of the form $x := c$ do not add any expressive power to stopwatch automata with only updates as $x := 0$. Hence the discussion ahead assumes that all updates in a stopwatch automata are just resets to 0.

3 Extensions on Rectangular Hybrid Automata

3.1 One Variable

**Lemma 3.1** The emptiness checking problem for a 1 variable uninitialized rectangular singular automata is decidable.

**Proof** We exhibit a partition of the integer line into regions and define a new region automata which is equivalent to the untimed of the given automata $A = \{L, l_0, x, \Sigma, \Phi(x), U(x), \delta, F, Act\}$.

**Definition** A critical point is a point on the integer line that appears somewhere in the constraint/update as an endpoint of a rectangular interval.

Let $C(A)$ be the set of such critical points. Order them and place them on the integer line. Let the ordered set be $C(A) = \{c_1, c_2, \ldots c_{max}\}$. Define the ordered set of regions $R(A)$ as

$\{(−\infty, c_1), (c_1, c_2), (c_2, \ldots (c_{max}, \infty))\}$

Two location, clock evaluation pairs $(l_1, v_1)$ and $(l_2, v_2)$ are equivalent iff $v_1$ and $v_2$ lie in the same region and $l_1 = l_2$.

It is easy to see that this equivalence relation respects the bisimulation criteria

1. (Consistency with constraints) Since any constraint in a region is clearly satisfied by all the points in the region.

2. (Consistency with updates) Since we have only one variable, this is trivial.

3. (Consistency with time-elapsed) If $v_1 \simeq v_2$ and $(l, v_1) \xrightarrow{t} (l, v'_1)$ i.e. $v'_1(x) = v_1(x) + k.t$ where $k$ is the slope in that location, then there exists a $t'$ such that $v'_1(x) = v_2(x) + k.t'$ by taking $t' = t + (v_1(x) - v_2(x))/k$.

4. (Consistency with jump) (1) and (2) imply this.
**Definition** Let $l$ be a location of $A$. We define $l$-time successor of a region $R$ as:

1. $R$

2. if $\dot{x} = 0$ in $l$, then only $R$

3. if $\dot{x} > 0$ in $l$ then all regions in $R(A)$ that occur to the right of $R$ on the integer line

4. if $\dot{x} < 0$ in $l$ then all regions in $R(A)$ that occur to the left of $R$ on the integer line

Now, we have a state transition system on the state space $R(A)$ with transitions defined as follows: There is a transition from $(l_1, R_1)$ to $(l_2, R_2)$ on a symbol $a$ iff there is a transition $<l_1, con, a, up, l_2 > \in \delta$ and there is a region $R'$ which is a $l_1$-time successor of $R_1$, $R'$ satisfies $con$ and $R_2 \in R'[up]$ (if $up$ is null then $R_2$ must be $R'$) Define initial and final states appropriately.

This new state transition system accepts empty language only iff $L(A)$ is empty.

### 3.2 Two Variables

**Lemma 3.2** The emptiness checking problem for a 2 variable stopwatch automata is decidable.

**Proof** $A = \{Q, \Sigma, \{x, y\}, \Phi(x, y), U(x, y), \delta, F\}$ We work with the regions $\text{Reg}_{AD}$ as described in the proof of theorem 2.1 for the 2 variable case. The notion of time successor of a region is changed to suit stopwatches.

**Definition** Let $l$ be a location of $A$. We define $l$-time successor of a region $R$ as:

1. $R$

2. if $\dot{x} = 0, \dot{y} = 0$ in $l$, then only $R$

3. if $\dot{x} = 1, \dot{y} = 0$ in $l$ then all regions in $R(A)$ that lie along the line $y = \alpha, x > \beta$, for some $(\beta, \alpha) \in R$

4. if $\dot{x} = 0, \dot{y} = 1$, then regions lying along the line $x = \beta, y > \alpha$ for some $(\beta, \alpha) \in R$.

5. if $\dot{x} = 1, \dot{y} = 1$, then the regions lying along the line $(y - \alpha) = (x - \beta), x > \beta, y > \alpha$ for some $(\beta, \alpha) \in R$.

Such a notion of time successors helps convince ourselves that the equivalence relation between states of $A$ induced by this set of regions indeed satisfies all the four requirements of being a bisimulation. It is obviously a finite bisimulation hence proving decidability.

### 3.3 Three Variables

A similar approach could be tried in the case where we have three variables $(x, y, z)$ and one of them is a stopwatch(say $x$).

If we look at the 3-d space partitioned as per A-D regions, and take a cross section at some $x = A; A$ being a constant. Let $a = \frac{A}{2}$. Then the 2-d space would look like this(ignore the dotted lines):
It is clear that for locations with $\dot{x} = 0$, this will not work, since some regions have two different immediate successors. So we draw out the dotted lines too. This amounts to refining the A-D partitions by introducing the following planes in each of the $1 \times 1 \times 1$ cubes.

\begin{align*}
  f(x) - f(y) - f(z) &= 0 \quad (5) \\
  f(z) - f(y) - f(x) &= 0 \quad (6) \\
  f(y) - f(z) + f(x) &= 1 \quad (7) \\
  f(z) - f(y) + f(x) &= 1 \quad (8)
\end{align*}

Where $f(x)$ denotes the fractional part of $x$.

But careful inspection shows that this is not a correct region construction for the case where all the three variables act as clocks. For eg. the point marked by a star in region $R_1$ would cross into region $R_2$ while the point marked by a dot would cross into region $R_3$.

So it looks like a finite region partitioning is not likely in case of three variables. Upon further digging into the literature, the following result was found which settles the question.

**Theorem 3.3** There exists an automaton with 2 clock and 1 stopwatch variables such that no bisimulation is finite. [BBR06]
Proof The proof relies on showing that for the automaton shown in the figure, each of the points in the following sequence necessarily form a distinct region in any bisimulation

\[(0, 1 - \frac{1}{2^n}, \frac{1}{2^n}), n = 1, 2, \ldots \] (9)

The complete proof is tedious.

4 Alternating Timed Automata

We start by defining alternating timed automata. Since we can prove decidability of such automata with only one clock variable, we consider just one variable. Let \( S \) be a finite set of locations and let \( x \) be a distinguished clock variable. The set \( \Phi^a(S) \) of formulas is generated by the grammar:

\[\phi ::= \bot | \top | \phi_1 \land \phi_2 | \phi_1 \lor \phi_2 | s | x \bowtie k | x.\phi\] (10)

where \( k \in \mathbb{N}, \bowtie \in \{<, \leq, \geq, >\}, s \in S \).

Definition A timed alternating automata is a tuple \( A = (\Sigma, S, s_0, F, \delta) \), where

- \( \Sigma \) is a finite alphabet
- \( S \) is a finite set of locations
- \( s_0 \in S \) is the initial location
- \( F \subseteq S \) is a set of accepting locations
- \( \delta : S \times \Sigma \rightarrow \Phi^a(S) \) is the transition function

A state of \( A \) is a pair \((s, v)\), where \( s \in S \) and \( v \in \mathbb{R}_{\geq 0} \) is a clock valuation. Write \( Q = S \times \mathbb{R}_{\geq 0} \) for the set of all possible states.

A set of states \( M \subseteq Q \) and a clock valuation \( v \in \mathbb{R}_{\geq 0} \) defines a Boolean valuation on \( \Phi^a(S) \) as follows:

- \( M \models_v s \) iff \( (s, v) \in M \)
- \( M \models_v x \bowtie k \) iff \( v \bowtie k \)
- \( M \models_v x.\phi \) iff \( M \models_0 \phi \)

(The boolean connectives are handled in the usual way)

Note that the satisfaction relation is monotone: \( N \models_v \phi \) implies \( M \models_v \phi \). We say that \( M \) is a minimal model of \( \phi \in \Phi^a(S) \) w.r.t \( v \) if \( M \models_v \phi \) and no proper subset of it models \( \phi \).

A configuration of states of \( A \) is a finite set of states; the set of configurations is denoted by \( 2^Q \). The initial configuration is \( \{s_0, 0\} \) and a configuration is accepting if all the locations in it are accepting. The empty configuration would be always accepting.

Definition Given a timed alternating automata \( A \), we define the labelled transition system \( T_A = (2^Q, \sim, \rightarrow) \) over the set of configurations. The \( R_{\geq 0} \) labelled transition relation \( \sim \) captures time evolutions or flow steps and is defined by

\[ P \sim_t P' \iff P' = \{(s, v + t) : (s, v) \in P\} \] (11)
The Σ-labelled transition relation \( \rightarrow \) captures instantaneous transitions between locations. Let \( P = \{(s_i, v_i)\}_{i \in I} \). We include a transition \( P \xrightarrow{a} P' \) iff one can choose, for each \( i \in I \), a minimal model \( M_i \) of \( \delta(s_i, a) \) w.r.t. \( v_i \), such that \( P' = \bigcup_{i \in I} M_i \).

Given a finite timed word \((\rho, \tau) = \{(\sigma_1, \tau_1), \ldots, (\sigma_n, \tau_n)\}\), with \( \tau_1 = 0 \), the runs of this ATA are defined from configuration to configuration. Write \( d_i = \tau_{i+1} - \tau_i \) for the time od delay between the \( i \)-th and \( i+1 \)-st event, \( 1 \leq i < n \). Define a run of \( A \) on \( \rho \) to be a finite alternating sequence of edge steps and flow steps in \( T_A \):

\[
P_0 \xrightarrow{\sigma_1} P_1 \xrightarrow{d_1} P_2 \ldots P_{2n-2} \xrightarrow{\sigma_n} P_{2n-1}
\]

\( P_0 \) is the initial configuration. The run is accepting if the last configuration \( P_{2n-1} \) is accepting, and the timed word is accepted by \( A \) if there exists an accepting run.

**Claim 4.1** One clock ATAs are closed under union and intersection.

**Proof** We just have to take disjunctions and conjunctions.

**Claim 4.2** One clock ATAs are closed under complementation.

**Proof** Define the dual formula \( \bar{\phi} \) of a formula \( \phi \in \Phi^a(S) \) as follows. The dual of a clock constraint is its negation, whereas the location is itself’s dual. For the propositional connectives we have de-morgan dualities. \( \phi_1 \lor \phi_2 \) is \( \bar{\phi}_1 \land \bar{\phi}_2 \). Clock resets distributes through duality operator. Define the complement automata \( A^c \) as \((\Sigma, S, s_0, S - F, \bar{\delta})\). Proof that this is in fact the complement can be found in [OW07].

### 4.1 Decidability of Language Emptiness

Any formula \( \phi \in \Phi^a(S) \) can be written in disjunctive normal form \( \phi = \bigvee_{j \in J} \bigwedge A_j \), where each \( A_j \) is a set of atoms of the form \( s, x.s, x \gg k \). For a set of atoms \( A \) and a clock valuation \( v \in \mathbb{R}_{\geq 0} \), let \( A[v] \subseteq Q \) be the set of states given by \( A[v] = \{(s,v): s \in A\} \cup \{(s,0): x.s \in A\} \). Then each minimal model \( M \) of \( \phi \) w.r.t \( v \) has the form \( M = A_j[v] \), for some \( j \in J \), where \( v \) satisfies all the clock constraints in \( A_j \).

For a configuration \( P = \{(s_i, u_i)\}_{i \in I} \), suppose a transition \( P \xrightarrow{a} P' \) exists in \( T_A \). Then, it is clear from the above consideration that \( P' = \bigcup_{i \in I} A_i[u_i] \), where for each \( i \in I \), the set of atoms \( A_i \) is a clause in the DNF of \( \delta(s_i, a) \).

Construct another transition system \( \mathcal{T} \) whose states are again configurations and the transitions are defined as: For two configurations \( P \) and \( P' \), \( P \xrightarrow{a,t} P' \) iff we can do a time lapse of \( t \) followed by an edge step on \( a \) in \( T_A \) taking \( P \) to \( P' \).

Accepting configuration are called bad states of \( \mathcal{T} \).

**Lemma 4.3** \( A \) accepts a timed word iff there is a path in \( \mathcal{T} \) labeled by the timed word from \( P_0 \) to a bad state in \( \mathcal{T} \).

Let \( \hat{\mathcal{T}} \) be a labeled transition system obtained from \( \mathcal{T} \) by erasing the time information from the transition labels. We still have

**Lemma 4.4** \( \mathcal{L}_A \) is non empty iff there is a path in \( \hat{\mathcal{T}} \) from \( P_0 \) to a bad state.
But the number of states in \(\hat{T}\) are potentially uncountable. We define a word \(H(P)\) to abstract our these states.

Let \(c_{\text{max}}\) be the largest constant occurring the in clock constraints. Define an equivalence relation \(\sim\) on \(\mathbb{R}_{\geq 0}\) by \(u \sim v\) if either \(u, v > c_{\text{max}}\), or floor and ceiling functions of \(u\) and \(v\) have the same value. The corresponding set of equivalence classes, or regions, is \(\text{REG} = \{r_0, r_0, 1, \ldots, r_{c_{\text{max}}, \infty}\}\), where \(r_x = \{x\}\), \(r_{x,y} = (x,y)\). Let \(\text{reg}(u)\) denote the equivalence class of \(u \geq 0\).

For \(u, v \in (\mathbb{R}_{\geq 0})^n\). Say that \(u \approx v\) iff \(u_i \sim v_i\), and \(\text{frac}(u_i) \leq \text{frac}(u_j)\) iff \(\text{frac}(v_i) \leq \text{frac}(v_j)\) for all \(i, j \in [n]\).

**Proposition 4.5** Let \(u, v \in \mathbb{R}_{\geq 0}^n\) and \(u \approx v\). Then for all \(t \geq 0\) there exists \(t' \geq 0\) such that \((u + t) \approx (v + t')\).

**Definition** An equivalence relation \(R\) on \(2^Q\) is a time-abstract bisimulation on \(T'\) if \(pRq\) implies:

- \((\forall a \in \Sigma)(\forall t \in \mathbb{R}_{\geq 0})(p \rightarrow_{a,t} p')\) implies \(\exists t' \exists q'(q \rightarrow_{a,t} q')\) and \(p'Rq'\)

Take \(\Lambda = \mathcal{P}(S \times \text{REG})\)

**Definition** Define \(H(P)\) from \(\Lambda^*\) as the one obtained by the following procedure:

- replace each \((s,u) \in P\) by a triple \((s, \text{reg}(u), \text{frac}(u))\)
- sort all these triples and quadruples w.r.t. \(\text{frac}(v)\)
- group together triples that have the same value of \(\text{frac}(v)\), ignoring multiple occurrences (this yields a finite sequence of finite sets of triples)
- forget about \(\text{frac}(v)\) (this yields a word in \(\Lambda^*\)).

**Definition** Let \(H\) be the transition system whose states are words \(H(P)\) for \(P\) a state of \(\hat{T}\); a transition \(W_1 \rightarrow_a W_2\) is in \(H\) if there is a transition \(P_1 \rightarrow_a P_2\) in \(\hat{T}\) with \(H(P_1) = W_1, H(P_2) = W_2\). The initial state in \(H\) is \(W_0 = H(P_0)\).

If \(P\) is bad and \(H(P) = H(P')\) then \(P'\) is bad as well. Therefore, it is correct to call a state \(W\) in \(H\) bad if \(W = H(P)\) and \(P\) is bad.

**Lemma 4.6** The equivalence relation induced by \(H\) is a bisimulation of \(T\).

**Proof** We prove this in two parts. Given \(P\) and \(Q\) with the property that \(H(P) = H(Q)\) and \(P = \{(s_i, u_i)\}_{i \in I}\) and \(Q = \{(t_i, v_i)\}_{i \in I}\) (index set is same because they are equivalent)

1. **Matching edge transitions.** Suppose we have a transition \(P \rightarrow_a P'\) (in \(\mathcal{T}_A\)), for some \(a\). By consideration of minimal models we know that \(P' = \bigcup_{i \in I} A_i[u_i]\), where for each \(i \in I\), the set of atoms \(A_i\) is a clause in the DNF of \(\delta(s_i, a)\). Setting \(Q' = \bigcup_{i \in I} A_i[v_i]\) (again, since \(H(P) = H(Q)\), the states in the configurations are the same), we have \(Q \rightarrow_a Q'\) (in \(\mathcal{T}_A\)).

Here we rely on the fact that since \(u_i \sim v_i\), they satisfy the same clock constraints. Also, \(H(Q') = H(P')\) is clear.

2. **Matching flow transitions:** Notice that \(H(P) = H(Q)\) implies that \(u \approx v\). In the light of proposition 4.5, for any \(t \in \mathbb{R}_{\geq 0}\), and \(P \rightarrow_t P'\) we can find a \(t'\) such that \(Q \rightarrow_{t'} Q'\) and \(H(P') = H(Q')\).
Lemma 4.7 \( \mathcal{L}(A) \) is non-empty iff a bad state is reachable in \( \mathcal{H} \) from \( W_0 \).

**Proof** By Lemma 4.4 we only need to show that a bad state is reachable in \( \mathcal{T} \) from \( P_0 \) iff a bad state is reachable in \( \mathcal{H} \).

Consider a transition system \( \mathcal{T}' \) obtained from \( \mathcal{T} \) by imposing one additional restriction on a transition that whenever \( u_1 \) and \( u_2 \) are in the same region, then the minimal model selected for \( (s,u_1) \) and \( (s,u_2) \) is the same. By \( \mathcal{T}' \) and \( \mathcal{H}' \) denote the transition systems obtained from \( \mathcal{T}' \). Clearly, this does not affect on acceptance. A bad state is reachable in \( \mathcal{H} \) iff a bad state is reachable in \( \mathcal{H}' \).

Now observe that the graph of \( \mathcal{H} \), i.e. the set of all pairs \( (P,H(P)) \), is a bisimulation between \( \mathcal{T}' \) and \( \mathcal{H}' \) (using Lemma 4.6) which proves the claim.

Lemma 4.8 For each state \( W \) in \( \mathcal{H} \), its set of successors\( \{W' \in \Lambda^* : W \rightarrow a W' \text{ for some } a\} \) is finite and effectively computable.

**Proof** The next time successor of a state \( W \) in \( \mathcal{H} \) can be obtained by simple cycil shifting of the letter in the word and appropriate changes to the region of the valuation. Refer [LW08]

The following result is major in what follows:

**Definition** Given a set \( Q \), a quasi-order on \( Q \) is a reflexive and transitive relation \( \preceq \subseteq Q \times Q \). An infinite sequence \( < q_1,q_2,.. > \) in \( Q \) is said to be saturating if there exist indices \( i < j \) such that \( q_i \preceq q_j \). A quasi-order \( \preceq \) is a well-quasi-order (wqo for short) on \( Q \) if every infinite sequence in \( Q \) is saturating.

Let \( \sqsubseteq \) be a quasi-order on \( \nu \). Define the induced monotone domination order \( \preceq \) on \( \nu^* \), the set of finite words over \( \nu \), as follows: \( a_1...a_m \preceq b_1...b_n \) if there exists a strictly increasing function \( f : \{1,...m\} \rightarrow \{1...n\} \) such that, for all \( 1 \leq i \leq m, a_i \sqsubseteq b_{f(i)} \).

Higman’s lemma:

**Lemma 4.9** If \( \sqsubseteq \) is a wqo on \( \nu \), then the induced monotone domination order \( \preceq \) is also a wqo on \( \nu^* \).

We can perform a search on \( \mathcal{H} \) effectively since for a given word \( W \), we can compute the set of successor words of \( W \) effectively. The termination of such a search is guaranteed by the Higman’s lemma (put \( \nu = \Lambda \) and \( \sqsubseteq = \subseteq \)). We leave apart the details of this because when we try to modify the proof for the stopwatch/snoopy case, these details will remain unchanged and hence are not important. Reader may refer to [OW07] or [LW08] for details.

5 Alternation and Stopwatches

In this case, we generalize the one variable ATA by allowing the clock variable to have different rates in different location. So the automaton is now a tuple \( A = (\Sigma, S, s_0, F, \delta, \text{Rate}) \), where \( \text{Rate} : S \rightarrow \{0,1\} \). For notation, we use \( \dot{s} \) to denote \( \text{Rate}(s) \).

The evolution of configurations of the ATA change in the following manner: The \( R_{\geq 0} \) labeled transition relation \( \sim_\tau \) captures time evolution or flow steps and is defined by

\[
P \sim_\tau P' \iff P' = \{(s,v + (s),t) : (s,v) \in P\}
\]

All the other semantics remain the same.
5.1 Language Emptiness

The approach outlined in the previous section does not work for this genera case because the function $H$ does not remain a bisimulation with respect to time elapses. This idea was outlined in the three variable hybrid automata case. Turns out, the language emptiness problem is in fact undecidable. We present that proof later. Before that we show decidability of a fragment.

5.2 Language Emptiness Decidability of One Variable Initialized Stopwatch ATA

Initialization condition essentially means that the clock variable must be reset whenever a state $s$ in a configuration introduces a new location $t$ in the configuration with a different rate of the variable. Syntactically, it means that for any location $s$ for every symbol $a$, the DNF of the $\delta(s,a)$ cannot have $t \in S$ as an atomic formula unless $\dot{s} = \dot{t}$.

Claim 5.1 Language emptiness problem for one variable initialized stopwatch ATA is decidable.

Proof The proof is exactly the same as that for one clock ATA with the same function $H(P)$ except for the first step. Define: $H^0(P)$

- Replace each $(s,u) \in P$ such that $\dot{s} = 1$, by a triple $(s,\text{reg}(u),\text{frac}(u))$. And replace each $(s,u) \in P$ such that $\dot{s} = 0$ by a triple $(s,r_{0},0)$.

Observe that in all states with rate 0, the clock values invariably remains zero because it gets reset in the beginning and whenever a transition occurs from a rate 1 state. And then it freezes to 0.

$H^0(P)$ induces a bisimulation over $T$. Matching edge steps goes through without change. Matching Time Steps: For a configuration $P$, $H^0(P)$ has some states with rate 0. The immediate time successor of $H^0(P)$ can be obtained by freezing the location, valuation pairs where the location has a rate 0 and cyclically shifting the rest as in the case where we constructed the timed successor of $H(P)$. The immediate successor is therefore unambiguous.

Successors of $W \in H$ can again be computed effectively. We just have to take care of making the right cyclic shifts as described earlier.

5.3 Language Emptiness Undecidability of One Variable Stopwatch ATA

Claim 5.2 Language emptiness problem for one variable stopwatch automata is undecidable.

We encode the halting problem of two counter machines using one stopwatch ATAs. The encoding relies heavily on the encoding of two counter machines presented in [HKPV95] in section 4.1 (case $k_2 = 0$ and $k_1 \neq 0$). We cite the theorem here in a slightly modified more strict form.

Theorem 5.3 The reachability problem is undecidable for a simple timed automata with one stopwatch and 5(might be 4) clocks.

The proof of this theorem encodes the instructions in a two counter machine using stopwatch automata. The values of the counters($u,v$) are kept track of by two distinguished clock
variables \((c = 2^{1-u}, d = 2^{1-v})\). Increment/decrement correspond to halving/doubling the clock variable. One of the gadgets is shown here.

These gadgets are then chained up according to the jump/next instructions of the two counter machine. The halting instruction is kept as the final state. The resulting automata has an empty language iff the machine never halts.

We now present a general approach to translate some special kinds of timed automata with stopwatches into ATAs with one stopwatch.

**Definition** A *strongly deterministic* stopwatch automata is a stopwatch automata where for each location and each symbol, we have a single transition and the variable constraint on that transition must only be a conjunction of atomic constraints i.e. generated by the grammar \(\Phi^d(C)\):

\[
\phi := \phi_1 \land \phi_2 \mid x \triangleright k
\]

where \(x \in C\), \(\triangleright \in \{<, \leq, \geq, >\}\) and \(k \in \mathbb{Z}^+\).

For \(\phi \in \Phi^d(C)\) and \(y \in C\), we also define \(\phi(y)\) as the part of the constraint referring only to \(y\). For example, \((x > 2 \land y \leq 3 \land y > 1)(y) = (y \leq 3 \land y > 1)\).

**Claim 5.4** Any strongly deterministic stopwatch automata can be translated into a one stopwatch ATA (preserving the language).

The translation is given below.

Consider a strongly deterministic stopwatch automaton \(A = (L^A, l_0^A, C, \Sigma^A, \delta^A_n, F^A, Act^A)\), where all the components in the tuple have the usual meaning and \(Act : L \times C \rightarrow \{0, 1\}\) is the activity function. Note that the transition function \(\delta^A_n \subseteq L \times \Sigma \times \Phi^d(C) \times L \times 2^C\) can be re-written as a partial function \(\delta^A : L \times \Sigma \rightarrow \Phi^d(C) \times L \times 2^C\). For every location and symbol, we have a unique transition. The third component \((2^C)\) would denote the set of variables that are reset on that transition.

We construct a one stopwatch ATA \(B = (\Sigma^B, L^B, l_0^B, \delta^B, F^B, Act^B)\)

- \(\Sigma^B = \Sigma^A \cup \{\$\}\)
- \(L^B = (L^A \times C) \cup \{q_0\}\)
- \(l_0^B = q_0\)
- \(\delta^B : L^B \times \Sigma^B \rightarrow \Phi^a(L^B)\) is the transition relation defined below:

\[
\delta^B(q_0, \$) = \bigwedge_{i \in C} (x = 0 \land (l_0^A, i))
\]

\(\forall i \in C, \forall l, l' \in L^A, a \in \Sigma^A,\) and \(\sigma \in \Phi^d(C)\) such that \(\delta^A(l, a) = (\sigma, l', R)\)

\[
\delta^B((l, i), a) = (\sigma(i)[i := x] \land x, (l', i)), \quad i \in R
\]

\[
\delta^B((l, i), a) = (\sigma(i)[i := x] \land (l', i)), \quad i \notin R
\]
[] is the substitution operator

- $F^B = F^A \times C$
- $A_{ct}^B : L^B \to \{0, 1\}$ is defined as below:

\[
A_{ct}^B((l, i)) = A_{ct}^A(l)_i \quad (18)
\]
\[
A_{ct}^B(q_0) = 1 \quad (19)
\]

$A_{ct}^A(l)_i$ is the i-th component of the vector $A_{ct}^A(l)$.

**Example** Consider the following SD TA over the alphabet $\Sigma = \{\alpha, \beta\}$. On transitions not marked with a symbol, assume the symbol is $\alpha$.

![Figure 2: A strongly deterministic timed automata](image1)

![Figure 3: Corresponding one clock ATA](image2)

**Claim 5.5** $L(B) = (\$, 0).L(A)$, . means concatenation.

**Proof** (I)($(\$, 0).L(A) \subseteq L(B)$)

Take a finite timed word

\[
w = (a_1, t_1), (a_2, t_2), ..., (a_n, t_n) \quad (20)
\]

in $L(A)$. The accepting run, $\rho$ looks like:

\[
(l_0, v_0) \xrightarrow{t_1} (l_0, v'_0) \xrightarrow{a_1} (l_1, v_1) \xrightarrow{t_2} (l_1, v'_1) \ldots (l_n, v_n) \quad (21)
\]

where $v_0 = 0$ and $l_n \in F^A$. 

13
The following sequence of configurations, with \( d_i = t_{i+1} - t_i \), show that \((\$, 0).w is in \( \mathcal{L}(\mathcal{B}) \).

\[
\{(q_0, 0)\} \rightarrow \$ P_1 \xrightarrow{d_1} P'_1 \xrightarrow{a_1} P_2 \ldots P'_n \xrightarrow{a_n} P_{n+1}
\]  \hspace{1cm} (22)

where \( P_i = \bigcup_{j \in \mathbb{C}} \{(l_{i-1}, j), (v_{i-1})_j\} \). Since \( l_n \in F^A \), \( P_{n+1} \) is accepting.

(II)(\$(, 0).\mathcal{L}(\mathcal{A}) \supseteq \mathcal{L}(\mathcal{B}))

First observe that \( \mathcal{L}(\mathcal{B}) \subseteq (\$, 0).T(\Sigma^A)^* \). Take any timed word

\[
(\$, 0), (a_1, t_1), \ldots, (a_n, t_n)
\]  \hspace{1cm} (23)

in \( \mathcal{L}(\mathcal{B}) \), with \( a_i \in \Sigma^A \). Consider any accepting sequence of configurations.

\[
\{(q_0, 0)\} \rightarrow \$ P_1 \xrightarrow{d_1} P'_1 \xrightarrow{a_1} P_2 \ldots P'_n \xrightarrow{a_n} P_{n+1}
\]  \hspace{1cm} (24)

Quite clearly, \( P_1 = \bigcup_{j \in \mathbb{C}} \{(l^A_j, j), 0\} \). Now using the strong determinism property, we can say that each \( \delta^\mathcal{B}((l, j), v) \) in any configuration state \((l, j, v)\) upon reading a symbol \( a \) would either have no model(clock constraints not being met or no transition present on the symbol) or a minimal model with a singleton state. Since the given sequence on configurations is accepting, at each stage, every state gives rise to exactly one new state in the next configuration upon reading a symbol. So configurations will be of the form \( P_i = \bigcup_{j \in \mathbb{C}} \{(l_{i-1}, j), v_{i-1, j}\} \) with \( l_n \in F^A \) since \( P_{n+1} \) is accepting. Take the following as the accepting run over \( \mathcal{A} \) for the word \((a_1, t_1), (a_2, t_2) \ldots (a_n, t_n)\):

\[
(l_0, \hat{0}), (l_1, \hat{v}_1) \ldots (l_n, \hat{v}_n)
\]  \hspace{1cm} (25)

For a clock \( c \in \mathcal{C} \), \( \hat{v}_i(c) = v_i.c \). Each \( \hat{v}_i \) satisfies the clock constraint in \( \mathcal{A} \) since the constraints were only conjunction of constraints over individual clock variables.

In the stopwatch automata obtained by coding the halting problem of a two counter machine, observe that the alphabet of the automata are of no significance in this construction. We can arbitrarily assign symbols to the transitions and the encoding would still be correct. Hence, the resulting stopwatch automata would be strongly deterministic. From the above claim, we can construct a one stopwatch ATA for that.

6 Snoopy ATAs and Connections to TPTL

We start by defining snoopy alternating timed automata with one variable. One clock snoopy ATA are strictly stronger than one clock simple ATA by definition. But, interestingly, it has decidable language emptiness. Let \( S \) be a finite set of locations and let \( x \) be a distinguished clock variable and \( m \in S \). The set \( \Phi_m^\Sigma(S) \) of formulas is generated by the grammar:

\[
\phi ::= \perp | T | \phi_1 \land \phi_2 | \phi_1 \lor \phi_2 | s | x \bowtie c | x.\phi | x(s') \bowtie c
\]  \hspace{1cm} (26)

where \( c \in \mathbb{N}, \bowtie \in \{<, \leq, \geq, >\}, s \in S \), and \( s' \in S - \{m\} \).

**Definition** A snoopy timed automata is a tuple \( \mathcal{A} = (\Sigma, S, s_0, F, \delta) \), where

- \( \Sigma \) is a finite alphabet
- \( S \) is a finite set of locations

14
• $s_0 \in S$ is the initial location

• $F \subseteq S$ is a set of accepting locations

• $\delta$ is a transition function. $\delta(m, a) \in \Phi_m^a(S)$

A state of $A$ is a pair $(s, v)$, where $s \in S$ and $v \in \mathbb{R}_{\geq 0}$ is a clock valuation. Write $Q = S \times \mathbb{R}_{\geq 0}$ for the set of all possible states.

A configuration of states of $A$ is a finite set of states; the set of configurations is denoted by $2^Q$. Given a configuration $P$, a state $(t, u) \in P$, and some set of states $M \subseteq Q$, the semantics of the transition function while evaluating models for $\delta(t, a) \in \Phi_t(S)$ for the state $(t, u) \in P$, are given by:

- $M |_P s$ iff $(s, u) \in M$
- $M |_P x \bowtie c$ iff $u \bowtie c$
- $M |_P x.\phi$ iff $M = (P - \{(t, u)\}) \cup \{(t, 0)\} \phi$
- $M |_P x(s') \bowtie c$ iff $\forall (s', v_{s'}) \in P$, we have $v_{s'} \bowtie c$.

(The boolean connectives are handled in the usual way)

**Definition** Given a timed alternating automata $A$, we define the labelled transition system $T_A = (2^Q, \rightsquigarrow, \rightarrow)$ over the set of configurations. The $\mathbb{R}_{\geq 0}$ labelled transition relation $\rightsquigarrow$ captures time evolutions or flow steps and is defined by

$$P \rightsquigarrow_t P' \iff P' = \{ (s, v + t) : (s, v) \in P \}$$

(27)

The $\Sigma$-labelled transition relation $\rightarrow$ captures instantaneous transitions between locations. Let $P = \{(s_i, v_i)\}_{i \in I}$. We include a transition $P \rightarrow_a P'$ iff one can choose, for each $i \in I$, a minimal model $M_i$ of $\delta(s_i, a)$ w.r.t. the configuration $P$, such that $P' = \bigcup_{i \in I} M_i$.

Given a finite timed word $(\rho, \tau) = \{(s_1, \tau_1), \ldots, (s_n, \tau_n)\}$, the runs of this ATA are defined from configuration to configuration in the usual sense.

**Remark** When we do not include the clock constraints of the kind $x(s') \bowtie c$ in the transitions, the definition is equivalent to the one-clock ATAs.

**Claim 6.1** Snoopy ATA are closed under union and intersection.

**Proof** We just have to take disjunctions and conjunctions respectively.

### 6.1 Decidability of One Variable Snoopy ATA

**Claim 6.2** The emptiness checking problem for snoopy ATA is decidable.

**Proof** We look back at the proof of language emptiness decidability for one clock ATA. The space of configurations here is exactly the same as in simple ATA. We use the same function $H$ to induce an equivalence relation.

First note that, given $(m, u)$, $m \in S$, and $u \in \mathbb{R}_{\geq 0}$ in a configuration $P$, any formula $\delta(m, a) \in \Phi_m^a(S)$ can be written in disjunctive normal form $\delta(m, a) = \vee_{j \in I} \land A_j$, where each $A_j$
is a set of atoms of the form \( s, x, s, x \preceq k, x(s') \preceq k, s' \in S - \{m\} \). For a set of atoms \( A \) from \( \Phi_m^0(S) \), and a state \((m, u)\) in a configuration \( P \), let \( A[P, m, u] \subseteq Q \) be the set of states given by \( A[P, m, u] = \{(s, u) : s \in A\} \cup \{(s, 0) : x.s \in A\} \). Then each minimal model \( M \) of \( \delta(m, a) \) w.r.t \( P \) has the form \( M = A_j[P, m, u] \), for some \( j \in J \), where \( u \) along with \( P \) satisfies all the clock constraints in \( A_j \).

For a configuration \( P = \{(s_i, u_i)\}_{i \in I} \), suppose a transition \( P \rightarrow_a P' \) exists in \( T_A \). Then, it is clear from the above consideration that \( P' = \bigcup_{i \in I} A_i[P, s_i, u_i] \), where for each \( i \in I \), the set of atoms \( A_i \) is a clause in the DNF of \( \delta(s_i, a) \).

Construct another transition system \( T \) whose states are again configurations and the transitions are defined as: For two configurations \( P \) and \( P' \), \( P \rightarrow_{a,t} P' \) iff we can do a time lapse of \( t \) followed by an edge step on \( a \) in \( T_A \) taking \( P \) to \( P' \).

\( H(P) \) gives rise to a bisimulation over \( T \). **Matching Time Steps** goes through without change. **Matching Edge Steps**: Suppose we have a transition \( P \rightarrow_a P'(in T_A) \), for some \( a \). By consideration of minimal models we know that \( P' = \bigcup_{i \in I} A_i[P, s_i, u_i] \), where for each \( i \in I \), the set of atoms \( A_i \) is a clause in the DNF of \( \delta(s_i, a) \). Setting \( Q' = \bigcup_{i \in I} A_i[Q, s_i, v_i] \) (again, since \( H(P) = H(Q) \), the states in the configurations are the same), we have \( Q \rightarrow_a Q'(in T_A) \). Here we rely on the fact that since \( \forall i \in I, u_i \sim v_i \), they satisfy the same clock constraints. Also, \( H(Q') = H(P') \) is clear.

Rest of the proof follows without change.

### 6.2 Snoopiness adds expressive power

**Example** We give example of a language which was proven to be outside the scope of one clock ATA in [LW08] but can be written as a one clock snoopy ATA.

**The candidate language**: Over the alphabet \( \Sigma = \{\alpha\} \). Every word has exactly two occurrences of \( \alpha \) between 0 and 1, say at \( t_1 \) and \( t_2 \) and precisely one occurrence between \( t_1 + 1 \) and \( t_2 + 1 \). Figure 4 gives a three variable timed automata for the language and figure 5 gives a one variable snoopy ATA for it.

![Timed automata for the language with three variables](image)

### 6.3 TPTL

**Definition TPTL** We use the negation normal form of TPTL defined by the syntax [AMSM+09]

\[
\psi ::= \bot \mid T \mid \psi_1 \land \psi_2 \mid \psi_1 \lor \psi_2 \mid p \mid \phi \psi \mid \phi \bar{\psi} \mid \circ \psi \mid \text{end} \mid y.\psi \mid y \preceq c
\]  

(28)

where \( p \in AP \) (set of atomic propositions), \( y \in C \) (set of clock variables).

**Pointwise Semantics** TPTL is interpreted over finite timed words over the vocabulary \( AP \). Models are timed words \( w \) and satisfiability is interpreted at a position \( i \in \mathbb{N} \) along the word. For a timed word \( w = (a_1, t_1), (a_2, t_2), \ldots, (a_n, t_n) \), we define the satisfiability relation,
Figure 5: Snoopy one clock ATA for the language

\[ w, i, v \models \phi \] saying that the the formula \( \phi \) is true at position \( i \) of the timed word \( w \) with valuation \( v \) of all the clock variables: [RS05]

- \( w, i, v \models p \) iff \( a_i = p \)
- \( w, i, v \models \psi U \phi \) iff there exists \( j \geq i \) such that \( w, j, v \models \phi \), and \( \forall i \leq k < j.w, k, v \models \psi \)
- \( w, i, v \models \psi U \phi \) iff \( \forall j \geq i, w, j, v \models \phi \), or \( \exists k.i \leq k < j.w, k, v \models \psi \)
- \( w, i, v \models \circ \phi \) iff \( i < n \), and \( w, i + 1, v \models \phi \)
- \textbf{end} \equiv \neg \circ \top \text{ is a temporal operator which is true only in the last position in the word.}

- \( w, i, v \models y.\phi \) iff \( w, i, v[y \rightarrow t_i] \models \phi \)
- \( w, i, v \models y \cdot c \) iff \( t_i - v(y) \cdot c \)

A word \( w \) is said to be accepted by a formula \( \phi \) iff \( w, 1, \emptyset \models \phi \).

**Remark** We do not have negation in the the syntax but from the semantics negation is definable. Using the following:

- \( \phi_1 U \phi_2 \equiv \neg (\neg \phi_1 U \neg \phi_2) \)
- \( \neg \circ \phi \equiv \circ \neg \phi \lor \textbf{end} \)

### 6.3.1 One Variable TPTL

**Claim 6.3** For every TPTL formula with only one clock variable, there exists a one clock ATA with exactly the same language.

**Translation:**
Given a TPTL formula \( \phi \) we construct one clock ATA \( A_\phi \) with the same language. Since timed alternating automata are closed under union and intersection, and since it is clear how to define \( A_\phi \) for atomic formulas we concentrate only on formulae who have outermost connectives...
of temporal operators. Define the closure of \( \phi \), denoted by \( cl(\phi) \), to consist of \( \phi \) itself all its subformulas with outermost connective as a temporal modality (including \( \text{end} \)), plus, for each subformula \( \circ \phi \), add a residual subformula \((\circ \phi)^r\). The locations of the ATA \( A_{\phi} \) are precisely \( cl(\phi) \). Starting location is \( \phi \). A location is accepting iff it corresponds to a subformula whose outermost connective is \( \bar{U} \) or \( \text{end} \). Transition function is defined as:

\[
\delta(a, a) = \top \quad (29)
\]

\[
\delta(a, b) = \bot, \quad b \neq a \quad (30)
\]

\[
\delta(\phi_1 \lor \phi_2, a) = \delta(\phi_1, a) \lor \delta(\phi_2, a) \quad (31)
\]

\[
\delta(\phi_1 \land \phi_2, a) = \delta(\phi_1, a) \land \delta(\phi_2, a) \quad (32)
\]

\[
\delta(\psi U \phi, a) = \delta(\phi, a) \lor \delta(\psi, a) \land (\psi U \phi)) \quad (33)
\]

\[
\delta(\psi U \phi, a) = \delta(\phi, a) \land (\delta(\psi, a) \lor (\psi U \phi)) \quad (34)
\]

\[
\delta(\circ \phi, a) = (\circ \phi)^r \quad (35)
\]

\[
\delta((\circ \phi)^r, a) = \delta(\phi, a) \quad (36)
\]

\[
\delta(\text{end}, a) = \bot \quad (37)
\]

\[
\delta(y \triangleright\triangleright c, a) = y \triangleright\triangleright c \quad (38)
\]

\[
\delta(y.\phi, a) = x.\delta(\phi, a) \quad (39)
\]

**Proof** We show that \( L(A_{\phi}) = L(\phi) \) in two parts.

(I) \( L(A_{\phi}) \subseteq L(\phi) \)

To this end, let \( w = (a, t) \), \(|w| = n \), be a timed word accepted by the automata. As usual, write \( d_i = t_{i+1} - t_i \) for \( 1 \leq i < n \). Suppose that the accepting run is:

\[
P_0 \rightarrow_\sigma P_1 \sim_\sigma_d P_2 \ldots P_{2n-1} \rightarrow_\sigma_n P_{2n-1} \quad (40)
\]

We claim that for each subformula \( \psi \) of \( \phi \) and each \( i \) such that \( 1 \leq i < n \), \((w, i, v) \models \psi \) whenever \( P_{2i-1} \models v, \delta(\psi, a_i) \). This is proved using structural induction.

### 6.3.2 Multiple Variable \( TPTL_+^F \)

\( TPTL_+^F \) is the negation free version of TPTL with \( F \) as the only temporal modality. The syntax is

\[
\psi ::= \bot \mid \top \mid \psi_1 \land \psi_2 \mid \psi_1 \lor \psi_2 \mid p \mid F\psi \mid y.\psi \mid y \triangleright\triangleright k \quad (41)
\]

where \( p \in AP, k \in \mathbb{N}, y \in C \), a set of clock variables, and \( F\psi \) has the same meaning as \( \top U \psi \).

**Claim 6.4** For any \( TPTL_+^F \) formula \( \phi \) over \( \Sigma \), we have a one clock snoopy ATA \( A_{\phi} \) over some other alphabet \( \Sigma' \) and a simple projection function \( h : \Sigma' \rightarrow \Sigma \), such that \( L(\phi) = h(L(A_{\phi})) \).

**Remark** By observing that both \( y \) and \( F \) distribute over disjunctions, we can re-write any TPTL formula to have the following form:

\[
\psi = \psi \lor \psi^\wedge \quad (42)
\]

\[
\psi^\wedge = \bot \mid \top \mid \psi_1^\wedge \land \psi_2^\wedge \mid p \mid F\psi^\wedge \mid y.\psi^\wedge \mid y \triangleright\triangleright k \quad (43)
\]

where \( \psi^\wedge \) denotes a disjunction free TPTL formula.
So, it suffices to construct one clock snoopy ATA for disjunction free formulae only. Henceforth, any formula we refer is assumed to be disjunction free.

**Translation**

Given a disjunction free TPTL formula $\phi$, we do a little preprocessing. We mark every $F$ operator in the formula by an identifying number so that we are able to distinguish between identical subformulae that occur at different points in $\phi$. Define the closure of $\phi$, referred to as $\text{cl}(\phi)$ to be the set of all sub formulae of $\phi$ which have $F$ as an outermost connective. The state space of the one clock snoopy ATA is $\text{cl}(\phi) \times C$.

Define preliminary transition formulae for these locations over the alphabet $\Sigma$ as follows. These would have to be modified later to get the correct automaton.

$$
\delta((a,i),b) = \begin{cases} 
\top & a = b \\
\bot & a \neq b 
\end{cases}
$$ (44)

$$
\delta((\phi_1 \land \phi_2,i),a) = \delta((\phi_1,i),b) \land \delta((\phi_2,i),a)
$$ (45)

$$
\delta((Fz \phi,i),a) = \delta((\phi,i),a) \lor (Fz \phi,i)
$$ (46)

$$
\delta((y.\phi,i),a) = \begin{cases} 
x.\delta((\phi,i),a) & i = y \\
\delta((\phi,i),a) & i \neq y 
\end{cases}
$$ (47)

$$
\delta((y \bowtie c,i),a) = \begin{cases} 
x \bowtie c & i = y \\
x((\Box, y)) \bowtie c & i \neq y 
\end{cases}
$$ (48)

$z$ is the identifying number of the operator $F$. $\Box$ is a placeholder which is supposed to contain a first component of a location of the automaton. After evaluation of the recursive definition of $\delta$, for each formula $\delta((Fz \psi,i),a)$ in the automata, we fill the place holders within that formula with $Fz \psi$.

The automata has $|C|$ portions one for each clock variable. The clock $x$ in portion $y$ keeps track of variable $y$ of $\phi$. So the above way of filling placeholder in $x((\Box, y)) \bowtie c$ essentially means that I want to look at the value of the clock in the corresponding location in the copy of the automaton that runs keeping track for value of the variable $y$ of $\phi$.

Also introduce a start location $q_0$ and a dummy alphabet $\$ with the transition:

$$
\delta(q_0, a) = \land_{y \in C} \delta((\phi,j),a)
$$ (49)

At this point, we observe that each portion of the automata runs fairly independently. Whenever a location $(Fz \psi,i)$ occurs in the configuration it means that the portion keeping track of variable $i$ is trying to assert $Fz \psi$ at the current point in the word. It might be possible that we need to check a constraint for another variable $y$ to go ahead. So the snoopy automata refers to the value of the clock in the location $(Fz \psi,y)$. So we need to ensure that the portion running for variable $y$ is also currently asserting the same subformula. To make sure that this is true, we synchronize various portions of the automata by introducing new alphabet. Write every transition formula $\delta(\psi,a)$ in DNF form. Construct $\delta'$ from $\delta$ as follows.

Given $\delta((\psi,y),a) = \lor_{j \in J} A_j$, introduce new alphabet $\{a_j : j \in J\}$ and say $\delta'(((\psi,y),a_j) = A_j$. Now since $A_j$ has no disjunctions, all the portions of the new automata (call it $A'$) are synchronized in the sense of the following claims.

**Claim 6.5** In any configuration $P$ of $A'$ arising from the starting configuration $\{(q_0,0)\}$ has the property that $\forall i,j \in C, ((\psi,i),u) \in P \iff \exists v$ such that $((\psi,j),v) \in P$.  

Claim 6.6 In any configuration $P$ of $A'$ arising from $\{(q_0, 0)\}$ for each $i \in C$ any subformula $F_2\psi$ occurs exactly once in $P$.

For this thesis, we do not prove the claims formally. But this is an interesting point for future work. Assuming the truth of these claims, it is easy to verify that $h(L(A')) = L(\phi)$, where $h(a_i) = a$, for every $a \in \Sigma$.

7 Summary

- Language emptiness for one stopwatch ATA is undecidable.
- One variable TPTL can be embedded into one clock ATA
- One variable Snoopy ATA is strictly more expressive than one clock ATA and still has decidable language emptiness.
- Multiple variable $TPTL^+_F$ can be embedded into one clock snoopy ATA and is hence its satisfiability is decidable.

8 Future Work

Following ideas are worth exploring:

- Are snoopy ATA closed under complementation? We have a candidate automata to show that the dual automaton would not work.
- Are one variable $TPTL$ and one stopwatch ATA equivalent?
- Are one stopwatch ATA contained in one variable $TPTL$ with past operators?
- Can we use snoopy ATA to embed richer $TPTL$ with multiple variables?

Acknowledgements

I thank Prof. Krishna S and Prof. Paritosh Pandya for their great guidance and support. I also thank Pritish Kamath for help with drawing automata in xfig.

References


