Adding Dense-Timed Stack to Integer Reset Timed Automata

Devendra Bhave[1(✉)] and Shibashis Guha[2]

[1] Indian Institute of Technology Bombay, Mumbai, India
devendra@cse.iitb.ac.in
[2] The Hebrew University of Jerusalem, Jerusalem, Israel
shibashis@cs.huji.ac.il

Abstract. Integer reset timed automata (IRTA) are known to be a determinizable subclass of timed automata, but it is not known whether they are input-determined, i.e., the clock values are completely determined by an input timed word. We first define a syntactic subclass of IRTA called strict IRTA and show that strict IRTA is equivalent to IRTA. We show that the class of strict IRTA is indeed input-determined. Visibly pushdown automata is another input-determined class of automata with a stack that is also closed under boolean operations and admits a logical characterization. We propose dtIRVPA as a class of timed automata with a dense-timed stack. Similar to strict IRTA, we define strict dtIRVPA and show that strict dtIRVPA is input-determined where both – stack operations and the values of the integer reset clocks – are determined by the input word, and this helps us to get the monadic second-order (MSO) logical characterization of dtIRVPA. We prove the closure properties of dtIRVPA under union, intersection, complementation, and determinization. Further, we show that reachability of dtIRVPA is PSPACE-complete, i.e. the complexity is no more than that of timed automata.

Keywords: Visibly pushdown automata · Dense-timed stack · Integer reset timed automata · Logical characterization · MSO

1 Introduction

Program verification involves ensuring that a program does not exhibit any unintended behavior. Such verification is often done by building a suitable computational model of the program, which needs to be sufficiently powerful to express program semantics, but without losing decidability of several interesting properties. Analysis and verification of such programs amount to checking various language theoretic properties of their corresponding models. For programs involving timed behaviour, timed automaton [4] is a simple yet powerful computational model. They use a finite set of real-valued variables – called *clocks* – all of which increase at the same rate as time elapses. Clocks can be reset as desired, which

S. Guha—Supported by FP7/2007-2013, ERC grant no 278410.

M. Hague and I. Potapov (Eds.): RP 2017, LNCS 10506, pp. 9–25, 2017.
DOI: 10.1007/978-3-319-67089-8_2

is useful to measure the time delay between various events. From a language theoretic perspective, though timed automata are closed under union and intersection, they are not closed under complementation and determinization. For this reason, it is not possible to verify a program modeled as a timed automaton against specifications given by another timed automaton.

Suman et al. [20] identified integer reset timed automata (IRTA), where a clock can be reset only when the value of some clock in an IRTA is an integer, as a *perfect subclass* of timed automata, that are closed under all the language theoretic operations such as union, intersection, complementation and determinization. Naturally, universality checking and inclusion checking are decidable for IRTA. But interestingly, IRTA are shown to be equivalent to their one clock counterpart.

Apart from timing constraints, presence of function calls and interrupts in the programs make the task of their verification difficult. Pushdown automata (PDA) is a popular formalism for modeling function calls. PDA are closed under union, but not under intersection, which limits their use for verification. A visibly pushdown automaton (VPA) [5] is a perfect subclass of deterministic PDA that is closed under union, intersection and complementation leading to decidable language emptiness and inclusion.

Though IRTA forms a relatively restricted class of timed automata, Mohalik el al. [19] have successfully used it in the latency analysis of distributed real-time systems that synchronize on integer global times. They model tasks which run periodically and communicate asynchronously using buffers with IRTA. Here Fig. 1 shows a self recursive procedure `P` in one such task which implements a boolean lock. Procedure `P` is a handler routine which is activated periodically by a module that uses integer reset clocks. We do not show that module, but instead discuss the usefulness of dense-timed stack in verifying richer properties. In Fig. 1, the dashed transitions correspond to either call or return transitions in different contexts of `P`. Now, consider the following specification: "If a lock is acquired in any context of a procedure, it must be released in the same context within 5 time units". Such requirement is enforced by pushing a symbol α on the stack when a lock is acquired and checking whether the age of α is within 5 units of time while releasing the lock.

Contributions. We consider task models as used in [19] augmented with dense-timed stacks. This motivates us to model recursive time-sensitive tasks using dense-timed integer reset visibly pushdown automata (dtIRVPA) as a model for real-time recursive programs. Our model uses integer reset clocks, visible alphabet and a dense-timed stack. Like VPA, an input symbol determines the stack operations, with the difference that we use a dense-timed stack. We show that the formalism of dtIRVPA enjoys all good properties like closure under union, intersection and determinization that paves the way to decidability of language inclusion based model checking, where the specification is given in terms of dtIRVPA. We consider a canonical form of dtIRVPA called the *strict* dtIRVPA that enjoys input determinacy property, *i.e.* when reading a timed word u, the clock values are completely determined by u. For the ease of presentation, we

```
Bool lock = false;
procedure p() {
  if(lock == false) {
    lock = true;
    P(); // Recursive call
    lock = false;
    // Must unlock within 5 time units
  }
}
```

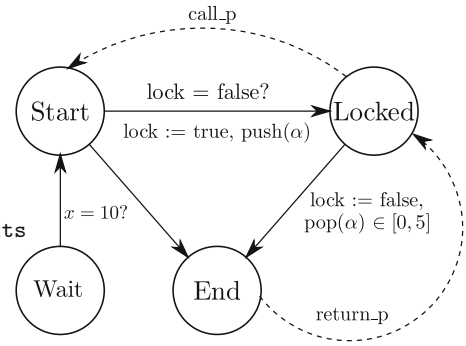

Fig. 1. A recursive program and its dtIRVPA model

first define *strict* IRTA and show that it is equivalent to the class of IRTA. A similar construction can be made for dtIRVPA which implies that the class of dtIRVPA is equivalent to the class of strict dtIRVPA. In this work, we also provide the monadic second-order (MSO) logical characterization for strict dtIRVPA. This allows us to check a system modelled as a dtIRVPA against any specification given by such an MSO formula. Another significant contribution of this work is to show that the location reachability checking is PSPACE-complete for dtIRVPA.

Related Work. Various models are known that combine clocks and pushdown stack to model programs with timing constraints on function calls. The earliest such model is a pushdown timed system [9]. It uses a set of global continuous variables and a timeless stack. Reachability of a location in the pushdown timed system is shown to be decidable. Dang [11] studied a model called pushdown timed automata that uses global clocks, but the clocks here are never pushed on the stack. Trivedi and Wojtczak [21] extended timed automata with recursion where they permitted pushing the clocks onto a stack using various mechanisms like pass-by-value and pass-by-reference and studied reachability and termination problems for this model. The model of dense-timed pushdown automata (dtPDA) by Abdulla et al. [1] is also closely related to our work. Whenever a stack symbol is pushed on the stack of a dtPDA, a real value called *age* is associated with the stack symbol. Ages of all symbols present in the stack increase uniformly with time. The problem of checking whether given a dtPDA and a location ℓ in it, there exists a run from an initial location of the dtPDA to location ℓ is shown to be decidable by Abdulla et al. [1] and is EXPTIME-complete. We mention here that in dtPDA, the age of a newly pushed stack symbol is initialized to a non-deterministic value while in the model of dtIRVPA introduced here, the age is initialized to zero. However, this difference is merely syntactic in nature and does not affect expressiveness [10]. Another noteworthy contribution is nested timed automata (NeTA) proposed by Li et al. [16]. In their model, an instance of timed automata can be pushed on the stack along with the clocks. Clocks of timed automata continue to run while on the stack. They have shown

the decidability of the reachability problem by reducing it to dtPDA. Recently they have further explored the model of NeTA where clocks on the stack are either frozen or progressing [17]. However reachability for this extension is undecidable for multiple global clocks.

Timed matching logic proposed by Droste and Perevoshchikov [12] is an existential fragment of second-order logic characterizing dtPDA. They effectively reduce a formula in timed matching logic to dtPDA such that it is satisfiable iff the language accepted by the corresponding dtPDA is non-empty. Recently Bhave et al. [7] proposed dense-timed visibly pushdown automata (dtVPA) as a perfect subclass of timed context-free languages in the sense that it is closed under union, intersection, complementation and determinization. Language inclusion is shown to be decidable for dtVPA. Further an equivalent MSO logic has been proposed for dtVPA. In [8], they have also studied a perfect subclass of context-sensitive timed languages called dense-timed multistack visibly pushdown automata (dtMVPA). Informally, a round of multistack computation accesses each stack once. dtMVPA has been shown to enjoy all good properties as that of dtVPA for the words having k-rounds of computations. For multistack dense-timed pushdown systems, Akshay et al. [3] have proposed a tree automata based technique for reachability problem on words with k-rounds.

Organization. We set up the technical definitions in Sect. 2. In Sect. 3, we give an effective construction to convert any IRTA into a language equivalent canonical *strict* IRTA. We show that strict IRTA are input-determined automata (IDA) that implies input determinacy for strict dtIRVPA, a property that we need for the logical characterization of strict dtIRVPA. In Sect. 4, we define dtIRVPA, discuss its closure properties, show that the model is determinizable and discuss the complexity of checking emptiness of dtIRVPA. In Sect. 5, we give a logical characterization of strict dtIRVPA. We conclude in Sect. 6.

We note that the abstract procedure for determinization of timed automata given in [6] identifies IRTA as a subset of timed automata that can be determinized by following the procedure. It constructs an intermediate symbolic infinite timed tree that satisfies the input-determinacy property. The deterministic symbolic infinite tree is folded back to construct the resulting deterministic timed automaton. The folding into a timed automaton is possible only when the number of *active clocks* in each node of the infinite tree is bounded by some $\gamma \in \mathbb{N}$. The folding back requires mapping the clocks of the infinite tree to a finite set of clocks X_γ. Under the γ-clock-boundedness, the only requirement is that each time a new clock in X_γ is needed for mapping a clock of the infinite tree to a clock in X_γ, one free clock is available in X_γ. This renaming, however, does not preserve the input-determinacy property and any relation to input-determinacy cannot be ascertained from the procedure in [6].

Also, once we show that strict IRTA is input-determined, from [13], it automatically implies that there exists an MSO logical characterization for strict IRTA and such a logical characterization can be derived from the framework given in [13] for any generic input-determined timed automaton. In [13], the authors define a timed MSO (TMSO) and show the equivalence between the

language of an IDA and a set of timed words satisfying a TMSO formula using a *proper symbolic alphabet*. The logic in [13] uses an input-determined operator Δ that has a semantic function of the signature $[\![\Delta]\!] : (T\Sigma^\omega \times \mathbb{N}) \mapsto 2^{\mathcal{I}_\mathbb{Q}}$, where $T\Sigma^\omega$ is the set of infinite timed words over an alphabet Σ, and $\mathcal{I}_\mathbb{Q}$ is the set of rational-bounded intervals over the non-negative reals. We use an operator $\zeta^k : \mathbb{N} \mapsto \mathcal{I}_\mathbb{Q} \cup \{\top\}$, where the symbol $\top$ denotes the clock values greater than some integer k.

2 Preliminaries

A timed automaton (TA) is a non-deterministic automaton that allows modeling of events to take place at specific time instants or within a time interval. It allows modeling the passage of time by a finite number of clock variables. All the clocks increase at the same rate. Lower case letters x, y, z will be used to denote clock variables and C will denote the set of clock variables. Clock variables are assigned non-negative real values.

Let $\uplus$ denote the disjoint union of sets. For a given $k \in \mathbb{N}$, let $\mathcal{I}^k = \{(p, p+1) \mid 0 \leq p < k\} \cup \{[p,p] \mid 0 \leq p \leq k\} \cup \{(k, \infty)\}$, where $p \in \mathbb{N}$, be a set of real disjoint intervals. Let $x \in C$ be a clock variable. Whenever it is clear from the context, we use $\mathcal{I}$ instead of $\mathcal{I}^k$. A clause is of the form $x \in I$. We say that a clause $(x \in I)$ holds true iff the current value of a clock variable x is in the interval I. A guard is a conjunction of finitely many such clauses and its syntax is given as $g := g \wedge g | (x \in I)$ where $x \in C$ and $I \in \mathcal{I}$. Let $g[(x+p)/x]$ be an expression obtained by replacing the variable x by the expression $(x+p)$ in the formula g. Let $\Phi(C)$ be the set of all guards.

A clock valuation or simply a valuation is a function $v : C \mapsto \mathbb{R}_{\geq 0}$. For a clock $x \in C$ and a valuation v, we use $v(x)$ to denote the value of clock x in v. We use $\lfloor v(x) \rfloor$ to denote the integer part of $v(x)$ while $frac(v(x))$ is used to denote the fractional part of $v(x)$. We define $\lceil v(x) \rceil = \lfloor v(x) \rfloor + 1$ if $frac(v(x)) \neq 0$, else $\lceil v(x) \rceil = \lfloor v(x) \rfloor$.

For a clock valuation v, we use $v+d$ to denote the clock valuation where every clock is being increased by an amount $d \in \mathbb{R}_{\geq 0}$. Formally, for each $d \in \mathbb{R}_{\geq 0}$, the valuation $v + d$ is defined as $(v + d)(x) = v(x) + d$, for each $x \in C$.

For a clock valuation v, we use $v_{[R \leftarrow \overline{0}]}$ to denote the clock valuation where every clock in $R \subseteq C$ is set to zero, while the value of the remaining clocks remain the same as in v. Formally, for each $R \subseteq C$, the valuation $v_{[R \leftarrow \overline{0}]}$ is defined by $v_{[R \leftarrow \overline{0}]}(x) = 0$ if $x \in R$, else $v_{[R \leftarrow \overline{0}]}(x) = v(x)$. We say that a valuation v satisfies a guard g, denoted $v \models g$, if for each clock x appearing in g, the formula obtained by replacing x with $v(x)$ is valid.

A timed automaton is defined by the tuple $(L, L_0, \Sigma, C, E, L_f)$ where L is a finite set of locations, L_0 is a non-empty set of initial locations, Σ is a finite alphabet disjoint from $\mathbb{R}_{\geq 0}$, C is a finite set of clocks, $E \subseteq L \times \Sigma \times \Phi(C) \times 2^C \times L$ is a finite set of edges and $L_f \subseteq L$ is a set of accepting locations. Note that our definition of guards is not succinct but it is equally expressive as the definition of [4]. The definition that we use in this paper allows us to have cleaner proofs.

A timed transition system [14], (TTS for short), $S = \langle Q, Q_0, \Sigma, \rightarrow, \hookrightarrow, Q_F \rangle$, where Σ is a finite alphabet, Q is a set of states, $q_0 \in Q_0$ is an initial state, $\rightarrow \subseteq Q \times \mathbb{R}_{\geq 0} \times Q$ is a set of delay transition relations, and $\hookrightarrow \subseteq Q \times \Sigma \times Q$ is a set of discrete transition relations. We write $q \xrightarrow{d} q'$ if $(q, d, q') \in \rightarrow$ and $q \overset{a}{\hookrightarrow} q'$ if $(q, a, q') \in \hookrightarrow$.

Let $\mathcal{A} = \langle L, L_0, \Sigma, C, E, L_f \rangle$ be a timed automaton. The semantics of a timed automaton is described by a TTS. The timed transition system $T(\mathcal{A})$ generated by $\mathcal{A}$ is defined as $T(\mathcal{A}) = \langle Q, Q_0, \Sigma, \rightarrow, \hookrightarrow, Q_F \rangle$, where

- $Q = \{(\ell, v) \mid \ell \in L,\ v \text{ is a clock valuation}\}$, is a set of states. Note that due to the real-valued nature of time, this set is generally uncountable.
- Let v_{init} denote the valuation such that $v_{init}(x) = 0$ for all $x \in C$. Then $Q_0 = \{(\ell_0, v_{init}) \mid l_0 \in L_0\}$.
- $\rightarrow = \{((\ell, v), (\ell, v + d)) \mid (\ell, v), d, (\ell, v + d) \in Q\}$ for all $d \in \mathbb{R}_{\geq 0}$.
- $\hookrightarrow = \{((\ell, v), a, (\ell', v'))$ such that $(\ell, v), (\ell', v') \in Q$ and there is an edge $e = (\ell, a, g, R, \ell') \in E$ and $v \models g$ and $v' = v_{[R \leftarrow \overline{0}]}\}$. From a state (ℓ, v), if $v \models g$, then there exists a $a \in \Sigma$ transition to a state (ℓ', v'); after this, the clocks in R are reset while the values of the clocks in $C \backslash R$ remain unchanged.
- $Q_F = \{(\ell, v) \mid l \in L_f$ and v is a clock valuation$\}$.

For a timed automaton state $p = (\ell, v)$, we denote by $v_x(p)$ the value of clock x for state p. A run of a timed automaton is of the form $\pi = (\ell_0, v_0) \xrightarrow{d_0} (\ell_0, v_0 + d_0) \xrightarrow{a_0} (\ell_1, v_1) \xrightarrow{d_1} (\ell_1, v_1 + d_1) \overset{a_1}{\hookrightarrow} (\ell_2, v_2) \ldots \xrightarrow{d_k} (\ell_k, v_k + d_k) \overset{a_k}{\hookrightarrow} (\ell_{k+1}, v_{k+1})$ where for all $i \geq 0$, we have $d_i \in \mathbb{R}_{\geq 0}$ and $a_i \in \Sigma$. Note that π is a continuous run in the sense that for a delay transition $(\ell_i, v_i), \xrightarrow{d_i} (\ell_i, v_i + d_i)$, it includes all states $(\ell_i, v_i + d)$ for all $0 \leq d \leq d_i$. A run is said to be *initial* if $\ell_0 \in L_0$ and $v_0 = v_{init}$. An initial run is accepting if it ends in an accepting location. A timed word $w = (a_0, t_0)(a_1, t_1) \ldots (a_k, t_k)$ is said to be read on π whenever $t_i = \sum_{j=0}^{i} d_j$ for every $1 \leq i \leq k$. The timed word w is said to be accepted by $\mathcal{A}$ if there is an initial and accepting run of $\mathcal{A}$ that reads u. We write $L(\mathcal{A})$ for the set of timed words (or timed language) accepted by $\mathcal{A}$. We say that a TA $\mathcal{A}$ is deterministic whenever every timed word w produces at most one unique run. The set of finite words over Σ is denoted by Σ^*. Additionally, we denote the set of all finite timed words by $T\Sigma^*$.

An *integer reset timed automaton* (IRTA) [20] is a timed automaton $\mathcal{A} = \langle L, L_0, \Sigma, C, E, L_f \rangle$ with the restriction that for every edge $e = \langle \ell, a, g, R, \ell' \rangle$, if $R \neq \emptyset$, then g has a clause of the form $x \in I$ for some $x \in C$ and I is of the form $[p, p]$ for $p \in \mathbb{N}$. Such clauses in the guard ensure that all resets happen at integer time units. A consequence of this is that at any time, for any run of an IRTA, the fractional parts of the values of all the clocks are the same [20].

It is known that given an IRTA $\mathcal{A}$, it can be determinized to produce another IRTA $\mathcal{B}$ whose size is exponential in the number of locations of $\mathcal{A}$ [18]. Further, given an IRTA $\mathcal{A}$ with an arbitrary number of clocks, it can be converted to an IRTA $\mathcal{B}$ such that $\mathcal{B}$ has a single clock and the number of locations in $\mathcal{B}$ is exponential in the number of clocks of $\mathcal{A}$ [18]. In both the above constructions, the timed language accepted by $\mathcal{A}$ is preserved.

A *strong timed simulation* relation between two timed systems $T_i = \langle Q_i, Q_{0,i}, \Sigma, \rightarrow_i, \hookrightarrow_i, Q_{F,i}\rangle$, for $i \in \{1,2\}$ is a relation $\mathcal{R} \subseteq Q_1 \times Q_2$ such that if $(q_1, q_2) \in \mathcal{R}$, and $q_1 \overset{\alpha}{\rightarrowtail} q_1'$, where $\rightarrowtail = \rightarrow$ and $\alpha \in \mathbb{R}_{\geq 0}$, or $\rightarrowtail = \hookrightarrow$ and $\alpha \in \Sigma$, then there exists $q_2' \in Q_2$ such that $q_2 \overset{\alpha}{\rightarrowtail} q_2'$ and $(q_1', q_2') \in \mathcal{R}$. A strong timed bisimulation relation between two timed systems T_i for $i \in \{1,2\}$ is a relation $\mathcal{R} \subseteq Q_1 \times Q_2$ such that both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are strong timed simulation relations. We say that two timed automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are timed bisimilar if for every initial state q_1 of $T(\mathcal{A}_1)$, there exists an initial state q_2 of $T(\mathcal{A}_2)$ such that there exists a strong timed bisimulation containing q_1 and q_2, and for every initial state q_2 of $T(\mathcal{A}_2)$, there exists an initial state q_1 of $T(\mathcal{A}_1)$ such that there exists a strong timed bisimulation containing q_1 and q_2.

Visibly pushdown automata [5] are a determinizable subclass of pushdown automata that operate over words that dictate the stack operations. This notion is formalized by giving an explicit partition of the alphabet into three disjoint sets of *call*, *return*, and *local* symbols and the visibly pushdown automata must push one symbol to the stack while reading a call symbol, and must pop one symbol (given the stack is non-empty) while reading a return symbol, and must not touch the stack while reading the local symbol. A *visibly pushdown alphabet* is a tuple $\Sigma = \langle \Sigma_c, \Sigma_r, \Sigma_l\rangle$where Σ is partitioned into a *call* alphabet Σ_c, a *return* alphabet Σ_r, and a *local* alphabet Σ_l. A visibly pushdown automaton over $\Sigma = \langle \Sigma_c, \Sigma_r, \Sigma_l\rangle$ is a tuple$(L, \Sigma, \Gamma, L^0, E, L_f)$ where L is a finite set of locations including a set $L^0 \subseteq L$ of initial locations, a finite stack alphabet Γ with special end-of-stack symbol $\vdash$, and $E \subseteq (L \times \Sigma_c \times L \times (\Gamma \setminus \vdash)) \cup (L \times \Sigma_r \times \Gamma \times L) \cup (L \times \Sigma_l \times L)$ and $L_f \subseteq L$ is a set of final locations. Alur and Madhusudan [5] showed that visibly pushdown automata are determinizable and closed under Boolean operations, concatenation, Kleene closure, and projection. They also showed that the language accepted by visibly pushdown automata can be characterized by MSO over words augmented with a binary matching predicate first studied in [15].

3 Transformation to Strict IRTA

In this section, we first define strict IRTA and show that the class of strict IRTA is equivalent to IRTA. Then we show that strict IRTA are input-determined automata.

Definition 1. *An IRTA is said to be* strict *iff* (i) *it has only one clock and* (ii) *every edge having a guard condition of the form* $[p, p]$ *with* $p \in \mathbb{N}$ *resets the clock.*

Assume wlog that we are given a one-clock (not necessarily deterministic), but non-strict IRTA $\mathcal{A} = (L, L_0, \Sigma, \{x\}, E, L_f)$. We now describe a construction that yields a strict IRTA $\mathcal{B}$ which is timed bisimilar to $\mathcal{A}$. For the following discussion, we assume k to be the maximum constant appearing in the guards of $\mathcal{A}$.

First, we present an intuition behind our construction with some observations. Consider the one clock IRTA $\mathcal{A}$ in Fig. 2 having a single clock x. We do

not show the labels on the edges with the letters from Σ in the figure. Clearly $\mathcal{A}$ is not a strict IRTA as the edge from ℓ_2 to ℓ_3 has an integer guard, but it does not reset x. Suppose we forcefully reset x on the ℓ_2 to ℓ_3 edge, then we need to suitably change the guards on all the edges that may be taken thereafter to preserve the timed language. For example, one of the changes that we can immediately identify is to make the guard in the edge from ℓ_3 to ℓ_1 as $x \in (0,1)$ instead of earlier $x \in (3,4)$ as the new value of x lags by 3. Clearly, the other side effects of resetting x must be taken care of appropriately. These side effects are *path sensititive*, *i.e.* for each location the amount of lag in x introduced because of additional reset of x depends completely on the incoming path to that location.

As a general principle, consider an edge $\ell \xrightarrow{a, x\in[m,m], \emptyset} \ell'$ which denotes a transition from a location ℓ to ℓ' on an input symbol a and checks whether the value of the clock x is m, but does not reset x. Suppose we modify it by adding a reset to it, clock x will then lag by m units along the run until x is reset again. Intuitively our construction keeps track of such a time lag introduced along the run in the locations. Let $\mathbb{N}_{\leq m} = \{0, 1, \ldots, m\}$ denote the set of natural numbers less than or equal to m. We maintain sets of lags $X_\ell \subseteq \mathbb{N}_{\leq k}$ for each location $\ell \in L$, where k is the maximum integer appearing in the guards of the IRTA. Let $\gamma : E \mapsto 2^{\mathbb{N}_{\leq k}}$ give the lag introduced by each edge. Recall that a guard I on an edge u of the one clock IRTA $\mathcal{A}$ is of the form $[m,m]$ or $(m, m+1)$ for $m \in \mathbb{N}$.

$$\gamma(u) = \begin{cases} \{m\} & \text{if } u = (\ell_1, x \in [m,m], a, \emptyset, \ell_2) \\ \{0\} & \text{if } u = (\ell_1, x \in [m,m], a, \{x\}, \ell_2) \\ X_{\ell_1} \cap \mathbb{N}_{\leq \inf(I)} & \text{if } u = (\ell_1, x \in I, a, \emptyset, \ell_2), \text{ where } I \neq [m,m] \text{ for } m \in \mathbb{N}_{\leq k} \end{cases}$$

Let $pre(\ell)$ be the set of all incoming edges to location ℓ. For computing the possible set of lags at each location, we write the set of fixed point equations for each location ℓ as $X_\ell = \bigcup_{u \in pre(\ell)} \gamma(u)$. For initial locations, there is no lag initially, so we initialize X_ℓ to $\{0\}$ for $\ell \in L_0$. Fixed point solution to these sets of equations exists as the sets are finite and set union is monotonicity preserving. Note that in each iteration $\gamma(u)$ is computed by intersecting with a constant set. Let $\overline{X}_\ell$ denote the fixed point solutions.

Now we construct a strict IRTA $\mathcal{B} = (L^{\mathcal{B}}, L_0^{\mathcal{B}}, \Sigma, \{x\}, E^{\mathcal{B}}, L_f^{\mathcal{B}})$. We record the time lag along the run in the locations of $\mathcal{B}$ such that $L^{\mathcal{B}} = \{(\ell, p) \mid \ell \in L \text{ and } p \in \overline{X}_\ell\}$ and initial locations are $L^{\mathcal{B}} = L_0 \times \{0\}$. Final locations are $L_f^{\mathcal{B}} = \{(\ell, p) \mid \ell \in L_f \text{ and } (\ell, p) \in L^{\mathcal{B}}\}$. The set of edges of $\mathcal{B}$ are given by

$$E^{\mathcal{B}} = \begin{array}{l} \{\big((\ell,p), a, (x + p \in I), \emptyset, (\ell', p)\big) \mid (\ell, x \in I, a, \emptyset, \ell') \in E \\ \qquad \text{and } I \neq [m,m] \text{ for } m \in \mathbb{N}_{\leq k} \text{ and } p \leq \inf(I)\} \cup \\ \{\big((\ell,p), a, (x + p \in [m,m]), \{x\}, (\ell', m)\big) \mid (\ell, (x \in [m,m]), a, \emptyset, \ell') \in E \\ \qquad \text{and } p \leq m\} \cup \\ \{\big((\ell,p), a, (x + p \in [m,m]), \{x\}, (\ell', 0)\big) \mid (\ell, (x \in [m,m]), a, \{x\}, \ell') \in E \\ \qquad \text{and } p \leq m\} \end{array}$$

Remark 1. Note that since the IRTA $\mathcal{A}$ has only one clock and its guards belong to $\mathcal{I}$, the size of the IRTA $\mathcal{B}$ is polynomial in the size of $\mathcal{A}$. Further, from $E^{\mathcal{B}}$, we

note that the maximum constant with which the clock is compared to in $\mathcal{B}$ is no more than the maximum constant with which the clock is compared to in $\mathcal{A}$.

We now apply our construction on the automaton in Fig. 2. We compute the set X_{ℓ_i} for each location ℓ_i where $0 \leq i \leq 3$. Initially X_{ℓ_0} is set to $\{0\}$, while for $i \neq 0$, we set $X_{l_i} = \emptyset$. Next X_{ℓ_1} is set to $\{0\}$ as it propagates from X_{ℓ_0} due to the presence of the edge from ℓ_0 to ℓ_1. Then X_{ℓ_2} is set to $\{0\}$ as it propagates from X_{ℓ_1} because of the edge from ℓ_1 to ℓ_2. Then X_{ℓ_3} is set to $\{3\}$ because of the edge from ℓ_2 to ℓ_3. Again X_{ℓ_1} is modified to $\{0, 3\}$ since 3 is propagated from X_{ℓ_3} due to the presence of the edge from ℓ_3 to ℓ_1. Thus we reach the following fixed point: $\overline{X}_{\ell_0} = \{0\}$, $\overline{X}_{\ell_1} = \{0, 3\}$, $\overline{X}_{\ell_2} = \{0\}$ and $\overline{X}_{\ell_3} = \{3\}$. Note that 3 is not added to X_{ℓ_2} since 3 is greater than the infimum of the guard on the edge from ℓ_1 to ℓ_2 which is 1. The edges are added following the definition of $E^{\mathcal{B}}$ given above. For example, for the edge from ℓ_3 to ℓ_1, consider the locations $(\ell_3, 3)$ and $(\ell_1, 3)$. In the strict IRTA that is obtained, we have the edge $\big((\ell_3, 3), \cdot, x{+}3 \in (3, 4), (\ell_1, 3)\big)$. The strict IRTA obtained is shown in Fig. 3.

Theorem 1. *Given a one clock IRTA $\mathcal{A}$, the construction presented above produces a strict IRTA $\mathcal{B}$ such that $\mathcal{A}$ and $\mathcal{B}$ are timed bisimilar.*

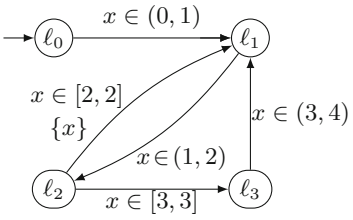

Fig. 2. An IRTA that is not strict.

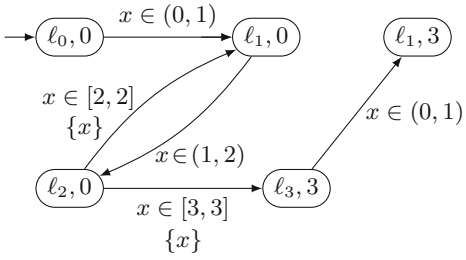

Fig. 3. A strict IRTA obtained from the one in Fig. 2.

We say that the class of timed automata has the property of *input determinacy* if the values of the clocks along a run are completely determined by the input timed word alone and do not depend on any specific instance of a timed automaton. Clearly the class of IRTA as a whole does not have this property, however, we claim that the strict IRTA does have the input determinacy property.

Lemma 1. *Strict IRTA have the input determinacy property.*

4 Dense-Timed Integer Reset Visibly Pushdown Automata

We introduce dense-timed integer reset visibly pushdown automata (dtIRVPA) as an IRTA augmented with a dense-timed stack having a visibly pushdown

alphabet. Let an input alphabet $\Sigma = \Sigma_c \uplus \Sigma_r \uplus \Sigma_l$ be partitioned into call, return and local symbols respectively. Let $\vdash$ be the special stack symbol denoting the bottom of the stack. We now formally define dtIRVPA and describe their semantics.

Definition 2 (dtIRVPA). *A dense-timed integer reset visibly pushdown automaton (dtIRVPA) over $\Sigma = \Sigma_c \uplus \Sigma_r \uplus \Sigma_l$ is a tuple $\mathcal{A} = (L, L_0, \Sigma, \Gamma, C, E = E_c \uplus E_r \uplus E_l, L_f)$, where L is the finite set of locations, $L_0, L_f \subseteq L$ are the set of initial and final locations respectively, Γ is the finite stack alphabet with special end-of-stack symbol $\vdash$, $E_c \subseteq (L \times \Sigma_c \times \Phi(C) \times 2^C \times L \times (\Gamma \setminus \vdash))$ is the finite set of call transitions, $E_r \subseteq (L \times \Sigma_r \times \Phi(C) \times 2^C \times \Gamma \times \mathcal{I} \times L)$ is the finite set of return transitions, and $E_l \subseteq (L \times \Sigma_l \times \Phi(C) \times 2^C \times L)$ is the finite set of local transitions. Additionally, the set of clocks that get reset in a transition is nonempty only if its guard condition contains a clause of the form $(x \in [p, p])$ where x is some clock in C and $p \in \mathbb{N}$.*

Each symbol on the stack has an associated clock-like real value (called *age*) which increments uniformly with time. When a symbol is pushed on the stack, its age is initialized to zero. As time elapses, ages of all stack symbols increase uniformly. A pop transition checks the age of the topmost stack symbol. We denote the contents of the stack in a configuration using a timed word itself. Let σ be such a timed word where $untime(\sigma) \in \Gamma^*$ is the string of (untimed) stack symbols while $age(\sigma)$ is the string of real valued ages. We define the age of stack bottom symbol $\vdash$ to be of undefined value, denoted by the special symbol $\perp$ which is not affected by the passage of time. Hence $(\perp + m)$ is defined to be $\perp$ for any $m \in \mathbb{R}_{\geq 0}$. We introduce a string concatenation operation :: for both types of strings for convenience in the next discussion.

Let $w = (a_1, t_1), \cdots, (a_n, t_n)$ be a timed word. A configuration of dtIRVPA is a tuple $(\ell, v, (\gamma\sigma, age(\gamma\sigma)))$, where ℓ is the current location of the dtIRVPA, v is the clock valuation, $\gamma\sigma \in \Gamma\Gamma^*$ is the content of the stack with γ being the topmost symbol and σ is the untimed word representing the stack content below γ, while $age(\gamma\sigma)$ is a string of real numbers encoding the ages of all the stack symbols (the time elapsed since each of them was pushed on to the stack).

The run of a dtIRVPA on a timed word $w = (a_1, t_1), \ldots, (a_n, t_n)$ is a sequence of configurations given as follows: $(\ell_0, v_0, (\langle\vdash\rangle, \langle\perp\rangle)), (\ell_1, v_1, (\sigma_1, age(\sigma_1))), \ldots,$ $(\ell_n, v_n, (\sigma_n, age(\sigma_n)))$ where $\ell_i \in L, \sigma_i \in \Gamma^*, \ell_0 \in L_0$, $t_0 = 0$, and for each i, $1 \leq i \leq n$, we have:

- If $a_i \in \Sigma_c$, then there is a transition $(\ell_{i-1}, a_i, g, R, \ell_i, \gamma) \in E$ such that $v_{i-1} + (t_i - t_{i-1}) \models g$. The clock valuation $v_i = (v_{i-1} + (t_i - t_{i-1}))_{[R \leftarrow \bar{0}]}$. The symbol $\gamma \in \Gamma \setminus \{\vdash\}$ is then pushed onto the stack, and its age is initialized to zero, obtaining $(\sigma_i, age(\sigma_i)) = (\gamma :: \sigma_{i-1}, 0 :: (age(\sigma_{i-1}) + (t_i - t_{i-1})))$. Note that the age of all symbols in the stack excluding the topmost one increases by $t_i - t_{i-1}$.
- If $a_i \in \Sigma_r$, then there is a transition $(\ell_{i-1}, a_i, g, R, \gamma, I, \ell_i) \in E$. The configuration $(\ell_{i-1}, v_{i-1}, (\sigma_{i-1}, age(\sigma_{i-1})))$ evolves to $(\ell_i, v_i, (\sigma_i, age(\sigma_i)))$ iff $v_{i-1} + (t_i - t_{i-1}) \models g$, $\sigma_{i-1} = \gamma :: \kappa \in \Gamma\Gamma^*$ and $age(\gamma) + (t_i - t_{i-1}) \in I$. Then

we obtain $\sigma_i = \kappa$, with $age(\sigma_i) = age(\kappa) + (t_i - t_{i-1})$. However, if $\gamma = \vdash$, the symbol is not popped, and the attached interval I is irrelevant. The clock valuation $v_i = (v_{i-1} + (t_i - t_{i-1}))_{[R \leftarrow \bar{0}]}$.

– If $a_i \in \Sigma_l$, then there is a transition $(\ell_{i-1}, a_i, g, R, \ell_i) \in E$ such that $v_{i-1} + (t_i - t_{i-1}) \vDash g$. The clock valuation $v_i = (v_{i-1} + (t_i - t_{i-1}))_{[R \leftarrow \bar{0}]}$. In this case the stack remains unchanged i.e. $\sigma_i = \sigma_{i-1}$, and $age(\sigma_i) = age(\sigma_{i-1}) + (t_i - t_{i-1})$. All symbols in the stack age by $t_i - t_{i-1}$.

A run ρ of a dtIRVPA $\mathcal{A}$ is accepting if it terminates in a final location. A timed word w is an accepting word if there is an accepting run of $\mathcal{A}$ on w. The language $L(\mathcal{A})$ of a dtIRVPA $\mathcal{A}$, is the set of all timed words w accepted by $\mathcal{A}$.

Definition 3 (Deterministic dtIRVPA). *A dtIRVPA $\mathcal{A} = (L, L_0, \Sigma, \Gamma, C, E, L_f)$ is said to be deterministic if it has exactly one start location, and for every location and input symbol pair exactly one transition is enabled at all times. Formally, we have the following conditions: (i) for call transitions $(\ell, a, g_1, R_1, \ell', \gamma_1), (\ell, a, g_2, R_2, \ell', \gamma_2) \in E_c$, we have $g_1 \wedge g_2$ is unsatisfiable. (ii) for return transitions $(\ell, a, g_1, R_1, \gamma_1, I_1, \ell'), (\ell, a, g_2, R_2, \gamma_2, I_2, \ell') \in E_r$, either $g_1 \wedge g_2$ is unsatisfiable or $I_1 \cap I_2 = \emptyset$. (iii) for local transitions $(\ell, a, g_1, R_1, \ell'), (\ell, a, g_2, R_2, \ell') \in E_l$, we have $g_1 \wedge g_2$ is unsatisfiable.*

Now we state one of the central results of the paper.

Theorem 2. *dtIRVPA are determinizable and closed under union, intersection and complementation. Also, their language emptiness and inclusion is decidable.*

Determinization. Although IRTA and VPA are individually determinizable, their determinization techniques cannot be easily combined for determinizing dtIRVPA due to the presence of a dense-timed stack. However technique used for determinization of dtVPA in [7] is helpful for handling dense-timed stack. We have the following result about the determinization of dtIRVPA.

Lemma 2. *dtIRVPA are determinizable.*

Closure properties and decision problems. The closure of dtIRVPA under union follows from the non-deterministic union of two dtIRVPA, while the intersection follows from their cross product construction. To obtain a complement of a given dtIRVPA, we first determinize it and then interchange final and non-final locations. We state the following results about the closure properties of dtIRVPA.

Lemma 3. *dtIRVPA are closed under union, intersection and complementation.*

The reachability problem for a dtPDA amounts to checking whether there exists a run starting from an initial configuration to a given configuration in the given dtPDA. If the given configuration is a final one, then this amounts to checking whether the language accepted by the dtPDA is empty. Abdulla et al. [1] proved that the configuration reachability checking problem for dtPDA is EXPTIME-complete. We now state the following theorem for the reachability checking of dtIRVPA.

Theorem 3. *The reachability checking for dtIRVPA is PSPACE-complete.*

PSPACE-hardness follows after a minor change in the initialization edges of the constructed timed automaton to yield an IRTA in the proof of the reachability problem of timed automata as given in [2]. An NPSPACE algorithm for emptiness checking is given below.

We first define the notion of regions for dtIRVPA, then state the construction of a region graph. Let $\mathcal{A} = (L, L_0, \Sigma, \Gamma, C, E, L_f)$ be a dtIRVPA with k as a maximum constant used in the guards. We define the regions for $\mathcal{A}$ as follows: The region is a triple (ℓ, u, b) where $\ell \in L$ is a location, $u : C \mapsto \mathbb{N} \cup \{k^+\}$ is a function that abstracts a clock valuation, and $b \in \{0, 1\}$ is a flag which denotes whether the fractional parts of the values of the clocks is zero. Here k^+ is a special symbol that represents clock values greater than k. We use $b = 0$ to denote that the fractional part of all clocks is zero and $b = 1$ to denote non-zero value of fractional part. Recall that Lemma 5 ensures that the value of fractional parts of all integer reset clocks is the same. For a valuation v, for each clock $x \in C$, we have $u(x) = k^+$ if $v(x) > k$, else $u(x) = \lfloor v(x) \rfloor$. We say that a state (ℓ, v, σ), where σ is the stack content, *belongs* to a region (ℓ, u, b) if for each clock $x \in C$, we have $u(x) = \lfloor v(x) \rfloor$ when $v(x) <= k$, and $u(x) = k^+$ when $v(x) > k$. Let $\mathcal{R}$ be the set of all such regions.

The region graph G induced by $\mathcal{A}$ is a graph in which the vertices are regions in $\mathcal{R}$ and the directed edges are labeled either by symbols in Σ or by intervals in $\mathcal{I}$. Edges are labeled by input symbols for discrete transitions between regions. If there is an edge labeled $I \in \mathcal{I}$ from region r_1 to r_2, it means that for every state $(\ell_1, v_1, \sigma_1) \in r_1$, there exists a $t \in I$ such that there is a concrete run from (ℓ_1, v_1, σ_1) that reaches some state $(\ell_2, v_2, \sigma_2) \in r_2$ after time $t \in I$.

Example 1. Consider a dtIRVPA $\mathcal{A}$ with a maximum constant $k = 2$ in the guards. This yields the set of intervals $\mathcal{I} = \big\{[0,0], (0,1), [1,1], (1,2), [2,2], (2,\infty)\big\}$. The set of clocks of $\mathcal{A}$ is $C = \{x, y, z\}$. Let the stack symbols be $\Gamma = \{\alpha, \beta\}$. Consider a state $s_1 = \big(\ell_1, v_1 = \langle 1.2, 2.2, 0.2\rangle, \sigma_1 = \langle(\alpha, 1.2), (\beta, 0.1)\rangle\big)$ of $\mathcal{A}$. Recall that all the clocks in an IRTA always have the same fractional part. The region r_1 to which s_1 belongs is $r_1 = (\ell_1, u_1 = \langle 1, 2^+, 0\rangle, 1)$. Note that the stack contents are not recorded in the regions. Now consider the effect of a time delay of 0.8 units on s_1. We get $s_2 = \big(\ell_1, v_2 = \langle 2.0, 3.0, 1.0\rangle, \sigma_2 = \langle(\alpha, 2.0), (\beta, 0.9)\rangle\big)$. The state s_2 belongs to region $r_2 = (\ell_1, u_2 = \langle 2, 2^+, 1\rangle, 1)$. This gives us an edge in the region graph $r_1 \xrightarrow{(0,1)} r_2$. In fact for every state $s \in r_1$, we can always choose some $t \in (0, 1)$ such that by delaying time t we reach some state in r_2.

Construction of region graph. We now use an alternative definition of timed words that uses delay intervals instead of global timestamps. For a timed word $w = (a_1, \delta_1) \ldots (a_n, \delta_n)$, duration of w is $\Sigma_{i=1}^{n} \delta_i$. Duration of a run is the duration of the timed word that induces the run. We define an *interval reachability* relation R_I between the regions such that if $(r_1, r_2) \in R_I$ then for every state $s_1 \in r_1$, there exists a $t \in I$ and a concrete run that reaches some state $s_2 \in r_2$ after time t. A region graph is constructed using the reachability relation $R = \bigcup_{I \in \mathcal{I}} R_I$.

Definition 4. *We define interval reachability relation R_I recursively as follows. We say $(r, r') \in R_I$ if one of the following cases holds.*

- $\big[$*Case* $I\big]$ *there are states* $(\ell, v_1, \sigma_1) \in r$ *and* $(\ell, v_2, \sigma_2) \in r'$ *and a time delay* $t \in I$ *such that* $v_2 = v_1 + t$ *and* $\sigma_2 = \sigma_1 + t$. *This case is for reachability by delay transition.*
- $\big[$*Case* $I \circ L\big]$ $(r, r_1) \in R_I$ *and there is a local transition from* r_1 *to* r'
- $\big[$*Case* $C \circ I \circ R\big]$ *all of the following hold*
  - *there is a call transition from* r *to* r_1 *that pushes a symbol* α *on the stack*
  - *there exists some* I*-labeled edge from* r_1 *to* r_2 *i.e.* $(r_1, r_2) \in R_I$
  - *there is a return transition from* r_2 *to* r' *with stack condition* $pop(\alpha) \in I$
- $\big[$*Case* $I \circ I\big]$ *there exist* $(r, r_1) \in R_{J_1}$ *and* $(r_1, r') \in R_{J_2}$ *and there exist* $t_1 \in J_1, t_2 \in J_2$ *and* $t \in I$ *such that* $t_1 + t_2 = t$.

We now give an NPSPACE algorithm to check if a region r_2 is reachable from a region r_1. The size of the input is $O(log\, k)$ bits as the input contains the description of a given dtIRVPA. Each abstract value of a clock requires $O(log\, k)$ bits and hence the function u needs $|C| \cdot O(log\, k)$ bits of memory. Thus a region requires $O(log\, k)$ bits which is polynomial in the size of the input. We systematically guess the regions at each transition along the run such that only $O(1)$ regions are stored in each step. Our algorithm performs the following steps until no new relation can be inferred further. (i) It add $(r_1, r_2) \in R_I$ for delay transitions using case I. (ii) It guesses regions r_1, r_2 and r_3 such that $(r_1, r_2) \in R_{I_1}$ and $(r_2, r_3) \in R_{I_2}$. It then concatenates them to get reachability from r_1 to r_3 using case $I \circ I$. (iii) It guesses regions $(r_1, r_2) \in R_I$ and guesses call and return transitions such that $r \to r_1$ is a call transition and $r_2 \to r'$ is return transition. It then infers that $(r, r') \in R_I$ using case $C \circ I \circ R$. (iv)It guesses regions r_1, r_2 and r_3 such that $(r_1, r_2) \in R_{I_1}$ and (r_2, r_3) is a local transition. It then infers the reachability from r_1 to r_3 using case $I \circ L$.

A timed word is called *well-matched* if there is no call position without a matching return position and vice versa. Our reachability algorithm can as well be extended to words that are not well-matched by treating unmatched calls and returns as local variables.

The problem of language inclusion checking for dtIRVPA is stated as whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$ for dtIRVPA $\mathcal{A}$ and $\mathcal{B}$ and this can be done by testing if $L(\mathcal{A}) \cap L(\overline{\mathcal{B}}) = \emptyset$. The language inclusion checking for VPA is EXPTIME-complete [5]. As VPA is a proper subclass of dtIRVPA, we have the following lemma.

Lemma 4. *For dtIRVPA* $\mathcal{A}$ *and* $\mathcal{B}$*, checking whether* $L(\mathcal{A}) \subseteq L(\mathcal{B})$ *is decidable and is EXPTIME-hard.*

5 MSO Characterization for Strict dtIRVPA

We can extend Definition 1 to dtIRVPA to obtain a *strict* dtIRVPAin a straightforward way. We restate the following key lemmas regarding the properties of IRTA that are instrumental in obtaining a strict IRTA from any given IRTA and given a dtIRVPA, we can obtain a strict dtIRVPA analogously.

Lemma 5. *[20] Let $\mathcal{A}$ be an IRTA and v be a clock valuation in any given run in $\mathcal{A}$. Then for all clocks x and y of $\mathcal{A}$, we have $frac(v(x)) = frac(v(y))$.*

Lemma 6. *[18] Given an IRTA $\mathcal{A}$, a deterministic one clock IRTA $\mathcal{B}$ can be constructed such that $L(\mathcal{A}) = L(\mathcal{B})$.*

Using the lemmas stated above and following a construction similar to the one presented in Sect. 3, we have the following lemma.

Lemma 7 (Clock reduction and strictness). *For every dtIRVPA $\mathcal{A}$, there is a strict dtIRVPA $\mathcal{B}$ such that $L(\mathcal{A}) = L(\mathcal{B})$.*

Theorem 4. *Strict dtIRVPA have input determinacy property.*

The above theorem allows us to have an MSO characterization for strict dtIRVPA.

Let $w = (a_1, t_1) \dots (a_m, t_m)$ be a finite timed word accepted by a strict dtIRVPA $\mathcal{A}$. Let $\mathcal{D}^w = \{1, \dots, m\}$ be the set of positions in w called the *domain* of w. Owing to Theorem 4, the value of the (only) clock at each position of w can be computed and is given by a function $\zeta^k : \mathcal{D}^w \mapsto [0, k] \cup \top$, where $[0, k]$ is the set of reals from zero to k and $\top$ is an artificially added symbol which intuitively denotes the clock values greater than k. We let $t_0 = 0$ to be the initial timestamp. Formally $\zeta^k(0) = 0$ and for $j > 0$,

$$\zeta^k(j) = \begin{cases} t_j - t_i \text{ if } i \text{ is the largest position with } 0 \leq i < j \text{ and having integer} \\ \qquad t_i \text{ and } t_j - t_i \leq k \text{ and } \zeta^k(i) \neq \top \\ \top \qquad \text{otherwise.} \end{cases}$$

We now define the syntax of $\mathsf{MSO}(\Sigma, k)$ formulas over alphabet Σ and parameter $k \in \mathbb{N}$ using the following syntax.

$$\varphi ::= Q_a(n) \mid \zeta_I^k(n) \mid \mu_I(n_1, n_2) \mid n_1 < n_2 \mid n \in X \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \exists n\, \varphi(n) \mid \exists X\, \varphi(X)$$

For every timed word w, we associate a word model $\underline{w}$ on which $\mathsf{MSO}(\Sigma, k)$ formulas are evaluated. Predicate $Q_a(n)$ checks whether the symbol $a \in \Sigma$ occurs at position n in w. Predicate $\zeta_I^k(n)$ checks whether the value of function ζ^k is in the interval $I \in \mathcal{I} \cup \{\top\}$ at position n. The ordering relation $<$ and the set membership relation $\in$ over the set positions $\mathcal{D}^w$ have their usual meaning. We introduce a stack matching predicate $\mu_I(n_1, n_2)$ which holds true when n_1 is a call position that pushes a stack symbol γ on the stack and n_2 is a matching return position which pops the same γ from the stack. Further, the time delay between the positions n_1 and n_2 is in the interval I. This ensures that the age of the topmost stack symbol at position n_2 is in I.

Consider an $\mathsf{MSO}(\Sigma, k)$ formula $\varphi(n_1, \dots, n_p, X_1, \dots, X_q)$ having p first order and q second order free variables. Let Var be the set of variables. Let $\mathbb{I} : Var \mapsto \mathcal{D}^w \cup 2^{\mathcal{D}^w}$ be the function which assigns values to all the free variables in φ. We denote the interpretation $\mathbb{I} = (k_1, k_2, \dots, k_p, K_1, K_2, \dots, K_q)$ such that it assigns

k_i to the first order free variable n_i and K_i to the second order free variable X_i in φ respectively.

The model of the formula $\varphi(n_1, \ldots, n_p, X_1, \ldots, X_q)$ is a tuple $\langle w, \mathbb{I} \rangle$ where $w \in T\Sigma^*$ is a finite timed word and $\mathbb{I}$ is an interpretation. This model is obtained by extending $T\Sigma$ with a bit vector of size $p+q$. For this purpose, we let $\Sigma_c^{p+q} = \Sigma_c \times \{0,1\}^{p+q}$, $\Sigma_r^{p+q} = \Sigma_r \times \{0,1\}^{p+q}$, and $\Sigma_l^{p+q} = \Sigma_l \times \{0,1\}^{p+q}$ be extended call, extended return and extended local alphabets respectively. Let $\Sigma^{p+q} = \Sigma_c^{p+q} \cup \Sigma_r^{p+q} \cup \Sigma_l^{p+q}$. Similarly we define $T\Sigma^{p+q} = T\Sigma \times \{0,1\}^{p+q}$. Let $w = (a_1, t_1), \ldots, (a_{|w|}, t_{|w|}) \in T\Sigma^*$ be a timed word. We encode $\mathbb{I}$ into an extended timed word $u = (\alpha_1, t_1), \ldots, (\alpha_{|w|}, t_{|w|}) \in (T\Sigma^{p+q})^*$ whose untimed alphabet set is extended to Σ^{p+q}. This encoding is done as follows.

- Both w and u have the same length.
- for each position $i \in \mathcal{D}^w$, $\alpha_i = (a_i, b_i^1, \ldots, b_i^p, c_i^1, \ldots, c_i^q) \in \Sigma^{p+q}$ such that
  - $b_i^j = 1$ iff $\mathbb{I}(n_j) = i$
  - $c_i^j = 1$ iff $i \in \mathbb{I}(X_j)$

We define the language of φ as $L(\varphi) = \{u \in (T\Sigma^{p+q})^* \mid \langle w, \mathbb{I} \rangle \models \varphi\}$.

The semantics of an $\mathsf{MSO}(\Sigma, k)$ formula in this system is given as follows.

$$
\begin{array}{ll}
\langle \underline{w}, \mathbb{I} \rangle \models Q_a(n) & \text{iff } a \text{ occurs at the position } \mathbb{I}(n) \\
\langle \underline{w}, \mathbb{I} \rangle \models \zeta_I^k(n) & \text{iff } \zeta^k(\mathbb{I}(n)) \in I \\
\langle \underline{w}, \mathbb{I} \rangle \models \mu_I(n_1, n_2) & \text{iff } \mathbb{I}(n_1) \text{ is a call and } \mathbb{I}(n_2) \text{ is matching return and} \\
 & \quad (t_{\mathbb{I}(n_2)} - t_{\mathbb{I}(n_1)}) \in I \\
\langle \underline{w}, \mathbb{I} \rangle \models n_1 < n_2 & \text{iff } \mathbb{I}(n_1) < \mathbb{I}(n_2) \\
\langle \underline{w}, \mathbb{I} \rangle \models n \in X & \text{iff } \mathbb{I}(n) \in \mathbb{I}(X) \\
\langle \underline{w}, \mathbb{I} \rangle \models \neg\varphi & \text{iff } \underline{w}, \mathbb{I} \not\models \varphi \\
\langle \underline{w}, \mathbb{I} \rangle \models \varphi \vee \varphi' & \text{iff } \langle \underline{w}, \mathbb{I} \rangle \models \varphi \text{ or } \langle \underline{w}, \mathbb{I} \rangle \models \varphi' \\
\langle \underline{w}, \mathbb{I} \rangle \models \exists n\, \varphi(n) & \text{iff there exists an } i \in D^w \text{ such that } \langle \underline{w}, \mathbb{I}[i/n] \rangle \models \varphi(n) \\
\langle \underline{w}, \mathbb{I} \rangle \models \exists X\, \varphi(X) & \text{iff there exists an } S \subseteq D^w \text{ such that } \langle \underline{w}, \mathbb{I}[S/X] \rangle \models \varphi(X)
\end{array}
$$

where $\mathbb{I}[i/n]$ denotes $\mathbb{I}(n) := i$ and $\mathbb{I}[S/X]$ denotes $\mathbb{I}(X) := S$.

Theorem 5. *L is a timed language over Σ accepted by a dtIRVPA with k as a maximum constant used in its guards iff there is an $\mathsf{MSO}(\Sigma, k)$ sentence φ that defines L.*

Let $\mathsf{MSO}^1(\Sigma, k)$ be the MSO logic obtained by removing atom $\mu_I(n_1, n_2)$ from $\mathsf{MSO}(\Sigma, k)$. One proof of the MSO^1 characterization of strict IRTA follows from Lemma 1 and the work by D'Souza and Tabareau [13]. As strict IRTA are a proper subclass of strict dtIRVPA, we get another proof of their MSO^1 characterization based on Theorem 5 and we state it as the following corollary.

Corollary 1. *L is a timed language over Σ accepted by a IRTA with k as a maximum constant used in its guards iff there is an $\mathsf{MSO}^1(\Sigma, k)$ sentence φ that defines L.*

6 Conclusion

The class of dense-timed integer reset timed automata introduced in this paper is a *perfect* subclass of timed context-free languages. The decidability of their inclusion checking paves the way for the model checking programs described as dtIRVPA against the subclass of richer timed context-free specifications. The other novelty is that their emptiness checking is PSPACE-complete which is same as that of timed automata. This is significant as timed automata cannot describe context free specifications.

References

1. Abdulla, P.A., Atig, M.F., Stenman, J.: Dense-timed pushdown automata. In: LICS, pp. 35–44 (2012)
2. Aceto, L., Laroussinie, F.: Is your model checker on time? on the complexity of model checking for timed modal logics. J. Log. Algebr. Program. **52–53**, 7–51 (2002)
3. Akshay, S., Gastin, P., Krishna, S.N.: Analyzing timed systems using tree automata. In: CONCUR, pp. 27:1–27:14 (2016)
4. Alur, R., Dill, D.: A theory of timed automata. Theor. Comput. Sci. **126**, 183–235 (1994)
5. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: STOC, pp. 202–211 (2004)
6. Baier, C., Bertrand, N., Bouyer, P., Brihaye, T.: When are timed automata determinizable? In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5556, pp. 43–54. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02930-1_4
7. Bhave, D., Dave, V., Krishna, S.N., Phawade, R., Trivedi, A.: A logical characterization for dense-time visibly pushdown automata. In: Dediu, A.-H., Janoušek, J., Martín-Vide, C., Truthe, B. (eds.) LATA 2016. LNCS, vol. 9618, pp. 89–101. Springer, Cham (2016). doi:10.1007/978-3-319-30000-9_7
8. Bhave, D., Dave, V., Krishna, S.N., Phawade, R., Trivedi, A.: A perfect class of context-sensitive timed languages. In: Brlek, S., Reutenauer, C. (eds.) DLT 2016. LNCS, vol. 9840, pp. 38–50. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53132-7_4
9. Bouajjani, A., Echahed, R., Robbana, R.: On the automatic verification of systems with continuous variables and unbounded discrete data structures. Hybrid Syst. **II**, 64–85 (1995)
10. Clemente, L., Lasota, S.: Timed pushdown automata revisited. In: LICS, pp. 738–749 (2015)
11. Dang, Z.: Binary reachability analysis of pushdown timed automata with dense clocks. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 506–517. Springer, Heidelberg (2001). doi:10.1007/3-540-44585-4_48
12. Droste, M., Perevoshchikov, V.: A logical characterization of timed pushdown languages. In: Beklemishev, L.D., Musatov, D.V. (eds.) CSR 2015. LNCS, vol. 9139, pp. 189–203. Springer, Cham (2015). doi:10.1007/978-3-319-20297-6_13
13. D'Souza, D., Tabareau, N.: On timed automata with input-determined guards. In: Lakhnech, Y., Yovine, S. (eds.) FORMATS/FTRTFT -2004. LNCS, vol. 3253, pp. 68–83. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30206-3_7

14. Henzinger, T.A., Majumdar, R., Prabhu, V.S.: Quantifying similarities between timed systems. In: Pettersson, P., Yi, W. (eds.) FORMATS 2005. LNCS, vol. 3829, pp. 226–241. Springer, Heidelberg (2005). doi:10.1007/11603009_18
15. Lautemann, C., Schwentick, T., Thérien, D.: Logics for context-free languages. In: Pacholski, L., Tiuryn, J. (eds.) CSL 1994. LNCS, vol. 933, pp. 205–216. Springer, Heidelberg (1995). doi:10.1007/BFb0022257
16. Li, G., Cai, X., Ogawa, M., Yuen, S.: Nested timed automata. In: Braberman, V., Fribourg, L. (eds.) FORMATS 2013. LNCS, vol. 8053, pp. 168–182. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40229-6_12
17. Li, G., Ogawa, M., Yuen, S.: Nested timed automata with frozen clocks. In: Sankaranarayanan, S., Vicario, E. (eds.) FORMATS 2015. LNCS, vol. 9268, pp. 189–205. Springer, Cham (2015). doi:10.1007/978-3-319-22975-1_13
18. Manasa, L., Krishna, S.N.: Integer reset timed automata: clock reduction and determinizability. CoRR, abs/1001.1215 (2010)
19. Mohalik, A., Rajeev, C., Dixit, M.G., Ramesh, S., Suman, P.V., Pandya, P.K., Jiang, S.: Model checking based analysis of end-to-end latency in embedded, real-time systems with clock drifts. In: DAC (2008)
20. Suman, P.V., Pandya, P.K., Krishna, S.N., Manasa, L.: Timed automata with integer resets: language inclusion and expressiveness. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 78–92. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85778-5_7
21. Trivedi, A., Wojtczak, D.: Recursive timed automata. In: Bouajjani, A., Chin, W.-N. (eds.) ATVA 2010. LNCS, vol. 6252, pp. 306–324. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15643-4_23