A Logical Characterization of Timed Pushdown Languages

Manfred Droste[⋆] and Vitaly Perevoshchikov[⋆⋆]

Universität Leipzig, Institut für Informatik,
04109 Leipzig, Germany
{droste,perev}@informatik.uni-leipzig.de

Abstract. Dense-timed pushdown automata with a timed stack were introduced by Abdulla et al. in LICS 2012 to model the behavior of real-time recursive systems. In this paper, we introduce a quantitative logic on timed words which is expressively equivalent to timed pushdown automata. This logic is an extension of Wilke's relative distance logic by quantitative matchings. To show the expressive equivalence result, we prove a decomposition theorem which establishes a connection between timed pushdown languages and visibly pushdown languages of Alur and Mudhusudan; then we apply their result about the logical characterization of visibly pushdown languages. As a consequence, we obtain the decidability of the satisfiability problem for our new logic.

Keywords: Timed pushdown automata, visibly pushdown languages, timed languages, relative distance logic, matchings

1 Introduction

Timed automata introduced by Alur and Dill [3] are a prominent model for the specification and analysis of real-time systems. Timed pushdown automata (TPDA) with a stack were studied in [6, 8, 12] in the context of the verification of real-time recursive systems. Recently, Abdulla, Atig and Stenman [1] proposed TPDA with a timed stack which keeps track of the age of its elements.

Since the seminal Büchi-Elgot theorem [7, 11] establishing the expressive equivalence of nondeterministic automata and monadic second-order logic, a significant field of research investigates logical descriptions of language classes appearing from practically relevant automata models. On the one hand, logic provides an intuitive way to describe the properties of systems. On the other hand, logical formulas can be translated into automata which may have interesting algorithmic properties. Furthermore, logic provides good insights into the understanding of the automata behaviors. The goal of this paper is to provide a logical characterization for timed pushdown automata, i.e., to design a logic on timed words which is expressively equivalent to TPDA.

For our purpose, we introduce a *timed matching logic*. As in the logic of Lautemann, Schwentick and Thérien [13], we handle the stack functionality by means of a binary

[⋆] Part of this work was done while the author was visiting Immanuel Kant Baltic Federal University, Kaliningrad, Russia.

[⋆⋆] Supported by DFG Graduiertenkolleg 1763 (QuantLA).

matching predicate. As in the logic of Wilke [18], we use *relative distance predicates* to handle the functionality of clocks. Moreover, to handle the ages of stack elements, we lift the binary matchings to the timed setting, i.e., we can compare the time distance between matched positions with a constant. The main result of this paper is the expressive equivalence of TPDA and timed matching logic.

Here, we face the following difficulties in the proof of our main result. The class of timed pushdown languages is most likely not closed under intersection and complement (as the class of context-free languages). Moreover, we cannot directly follow the approaches of [13] and [18], since the proof of [13] appeals to the logical characterization result for trees [17] (but, there is no suitable logical characterization for regular timed tree languages) and the proof of [18] appeals to the classical Büchi-Elgot result [7, 11] (and, this way does not permit to handle matchings). We solve this problem by establishing a connection between TPDA and *visibly pushdown automata* of Alur and Madhusudan [4].

We show our expressive equivalence result as follows.

– We prove a Nivat-like decomposition theorem for TPDA (cf. [15, 5]) which may be of independent interest; this theorem establishes a connection between timed pushdown languages and untimed visibly pushdown languages of [4] by means of operations like renamings and intersections with simple timed pushdown languages. So we can separate the continuous timed part of the model of TPDA from its discrete part. The main difficulty here is to encode the infinite time domain, namely $\mathbb{R}_{\geq 0}$, as a finite alphabet. We will show that it suffices to use several partitions of $\mathbb{R}_{\geq 0}$ into intervals to construct the desired extended alphabet. On the one hand, we interpret these intervals as components of the extended alphabet. On the other hand, we use them to control the timed part of the model.
– In a similar way, we separate the quantitative timed part of timed matching logic from the qualitative part described by MSO logic with matchings over a visibly pushdown alphabet [4].
– Then we can deduce our result from the result of [4].

Since our proof is constructive and the reachability for TPDA is decidable [1], we can also decide the satisfiability for our timed matching logic.

2 Timed Pushdown Automata

In this section, we consider *timed pushdown automata* which have been introduced and investigated in [1]. These machines are nondeterministic automata equipped with finitely many *global* clocks (like timed automata) and a stack (like pushdown automata). In contrast to untimed pushdown automata, in the model of TPDA we push together with a letter a *local* clock whose initial age can be an arbitrary real number from some interval. Like in timed automata, the values of global clocks and the ages of local clocks grow in time. Then, we can pop this letter only if its age belongs to a given interval. Note that, when considering all possible runs of a TPDA, the number of used local clocks is in general not bounded by any constant. We slightly extend the definition of TPDA presented in [1] by allowing labels of edges. This, however, does not harm the decidability of the reachability problem which was shown in [1]. Note that the model of TPDA of [1] extends the model of timed automata with untimed stack proposed in [6].

An *alphabet* is a non-empty finite set. Let Σ be a non-empty set (possibly infinite). A *finite word* over Σ is a finite sequence $a_1...a_n$ where $n \geq 0$ and $a_1, ..., a_n \in \Sigma$. If $n = 0$, then we say that w is *empty* and denote it by ε. Otherwise, we call w *non-empty*. Let Σ^* denote the set of all words over Σ and Σ^+ denote the set of all non-empty words over Σ. Let $\mathbb{R}_{\geq 0}$ denote the set of all non-negative real numbers. A *finite timed word* over Σ is a finite word over $\Sigma \times \mathbb{R}_{\geq 0}$. Let $\mathbb{T}\Sigma^* = (\Sigma \times \mathbb{R}_{\geq 0})^*$, the set of all finite timed words over Σ, and $\mathbb{T}\Sigma^+ = (\Sigma \times \mathbb{R}_{\geq 0})^+$, the set of all non-empty finite timed words over Σ. Any set $\mathcal{L} \subseteq \mathbb{T}\Sigma^+$ of finite timed words is called a *timed language*. For $w = (a_1, t_1)...(a_n, t_n) \in \mathbb{T}\Sigma^+$, let $|w| = n$, the *length* of w and $\langle w \rangle = t_1 + ... + t_n$, the *time length* of w. For $0 \leq i < j \leq n$, let $\langle w \rangle_{i,j} = t_{i+1} + ... + t_j$.

Let $\mathcal{I}$ denote the class of all intervals of the form $[a, b]$, $(a, b]$, $[a, b)$, (a, b), $[a, \infty)$ or (a, ∞) where $a, b \in \mathbb{N}$. Let C be a finite set of *clock variables* ranging over $\mathbb{R}_{\geq 0}$. A *clock constraint* over C is a mapping $\phi : C \to \mathcal{I}$ which assigns an interval to each clock variable. Let $\mathcal{I}^C$ be the set of all clock constraints over C. A *clock valuation* over C is a mapping $\nu : C \to \mathbb{R}_{\geq 0}$ which assigns a value to each clock variable. Let $\mathbb{R}^C_{\geq 0}$ denote the set of all clock valuations over C. For $\nu \in \mathbb{R}^C_{\geq 0}$ and $\phi \in \mathcal{I}^C$, we write $\nu \models \phi$ if $\nu(c) \in \phi(c)$ for all $c \in C$.

For $t \in \mathbb{R}_{\geq 0}$, let $\nu + t : C \to \mathbb{R}_{\geq 0}$ be defined for all $c \in C$ by $(\nu + t)(c) = \nu(c) + t$. For $\Lambda \subseteq C$, let $\nu[\Lambda := 0] : C \to \mathbb{R}_{\geq 0}$ be defined by $\nu[\Lambda := 0](c) = 0$ for all $c \in \Lambda$ and $\nu[\Lambda := 0](c) = \nu(c)$ for all $c \in C \setminus \Lambda$. If Γ is an alphabet, $u = (g_1, t_1)...(g_n, t_n) \in \mathbb{T}\Gamma^*$ and $t \in \mathbb{R}_{\geq 0}$, let $u + t = (g_1, t_1 + t)...(g_n, t_n + t) \in \mathbb{T}\Gamma^*$.

We denote by $\mathcal{S}(\Gamma) = (\{\downarrow\} \times \Gamma \times \mathcal{I}) \cup \{\#\} \cup (\{\uparrow\} \times \Gamma \times \mathcal{I})$ the set of *stack commands* over Γ.

Definition 2.1. *Let Σ be an alphabet. A* timed pushdown automaton (TPDA) *over Σ is a tuple $\mathcal{A} = (L, \Gamma, C, L_0, E, L_f)$ where L is a finite set of* locations, *Γ is a finite* stack alphabet, *C is a finite set of* clocks, *$L_0, L_f \subseteq L$ are sets of* initial *resp.* final *locations, and $E \subseteq L \times \Sigma \times \mathcal{S}(\Gamma) \times \mathcal{I}^C \times 2^C \times L$ is a finite set of* edges.

Let $e = (\ell, a, s, \phi, \Lambda, \ell') \in E$ be an edge of $\mathcal{A}$ with $\ell, \ell' \in L$, $a \in \Sigma$, $s \in \mathcal{S}(\Gamma)$, $\phi \in \mathcal{I}^C$ and $\Lambda \subseteq C$. We will denote e by $\ell \xrightarrow[s]{a,\phi,\Lambda} \ell'$. We say that a is the *label* of e and denote it by $\mathrm{label}(e)$. We also let $\mathrm{stack}(e) = s$, the stack command of e. Let $E^{\downarrow} \subseteq E$ denote the set of all *push* edges e with $\mathrm{stack}(e) = (\downarrow, \gamma, I)$ for some $\gamma \in \Gamma$ and $I \in \mathcal{I}$. Similarly, let $E^{\#} = \{e \in E \mid \mathrm{stack}(e) = \#\}$ be the set of *local* edges and $E^{\uparrow} = \{e \in E \mid \mathrm{stack}(e) = (\uparrow, \gamma, I)$ for some $\gamma \in \Gamma$ and $I \in \mathcal{I}\}$ the set of *pop* edges. Then, we have $E = E^{\downarrow} \cup E^{\#} \cup E^{\uparrow}$.

A *configuration* c of $\mathcal{A}$ is described by the present location, the values of the clocks, and the stack, which is a timed word over Γ. That is, c is a triple $\langle \ell, \nu, u \rangle$ where $\ell \in L$, $\nu \in \mathbb{R}^C_{\geq 0}$ and $u \in \mathbb{T}\Gamma^*$. We say that c is *initial* if $\ell \in L_0$, $\nu(x) = 0$ for all $x \in C$ and $u = \varepsilon$. We say that c is *final* if $\ell \in L_f$ and $u = \varepsilon$. Let $\mathcal{C}_{\mathcal{A}}$ denote the set of all configurations of $\mathcal{A}$, $\mathcal{C}^0_{\mathcal{A}}$ the set of all initial configurations of $\mathcal{A}$ and $\mathcal{C}^f_{\mathcal{A}} \subseteq \mathcal{C}_{\mathcal{A}}$ the set of all final configurations.

Let $c = \langle \ell, \nu, u\rangle$ and $c' = \langle \ell', \nu', u'\rangle$ be two configurations with $u = (\gamma_1, t_1)(\gamma_2, t_2)...(\gamma_k, t_k)$ and let $e = (q, a, s, \phi, \Lambda, q') \in E$ be an edge. We say that $c \vdash_e c'$ is a *switch transition* if $\ell = q$, $\ell' = q'$, $\nu \models \phi$, $\nu' = \nu[\Lambda := 0]$, and:

- if $s = (\downarrow, \gamma, I)$ for some $\gamma \in \Gamma$ and $I \in \mathcal{I}$, then $u' = (\gamma, \tau)u$ for some $\tau \in I$;
- if $s = \#$, then $u' = u$;
- if $s = (\uparrow, \gamma, I)$ with $\gamma \in \Gamma$ and $I \in \mathcal{I}$, then $k \geq 1$, $\gamma = \gamma_1$, $t_1 \in I$ and $u' = (\gamma_2, t_2)...(\gamma_k, t_k)$.

For $t \in \mathbb{R}_{\geq 0}$, we say that $c \vdash_t c'$ is a *delay transition* if $\ell = \ell'$, $\nu' = \nu + t$ and $u' = u + t$. For $t \in \mathbb{R}_{\geq 0}$ and $e \in E$, we write $c \vdash_{t,e} c'$ if there exists $c'' \in \mathcal{C}_{\mathcal{A}}$ with $c \vdash_t c''$ and $c'' \vdash_e c'$.

A *run* ρ of $\mathcal{A}$ is an alternating sequence of delay and switch transitions which starts in an initial configuration and ends in a final configuration, formally, $\rho = c_0 \vdash_{t_1,e_1} c_1 \vdash_{t_2,e_2} ... \vdash_{t_n,e_n} c_n$ where $n \geq 1$, $c_0 \in \mathcal{C}^0_{\mathcal{A}}$, $c_1, ..., c_{n-1} \in \mathcal{C}_{\mathcal{A}}$, $c_n \in \mathcal{C}^f_{\mathcal{A}}$, $t_1, ..., t_n \in \mathbb{R}_{\geq 0}$ and $e_1, ..., e_n \in E$. The *label* of ρ is the timed word $\text{label}(\rho) = (\text{label}(e_1), t_1)...(\text{label}(e_n), t_n) \in \mathbb{T}\Sigma^+$. Let $\mathcal{L}(\mathcal{A}) = \{w \in \mathbb{T}\Sigma^+ \mid \text{there exists a run } \rho \text{ of } \mathcal{A} \text{ with } \text{label}(\rho) = w\}$, the timed language *recognized* by $\mathcal{A}$. We say that a timed language $\mathcal{L} \subseteq \mathbb{T}\Sigma^+$ is a *timed pushdown language* if there exists a TPDA $\mathcal{A}$ over Σ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}$.

Note that every timed automaton $\mathcal{A} = (L, C, I, E, F)$ can be considered as a TPDA $\mathcal{A} = (L, \Gamma, C, I, E, F)$ where Γ is an arbitrary alphabet and $E = E^{\#}$.

Example 2.2. Here, we consider a timed extension of the well-known *Dyck languages*. Let $\Sigma = \{a_1, ..., a_m\}$ be a set of opening brackets and $\overline{\Sigma} = \{\overline{a}_1, ..., \overline{a}_m\}$ a set of corresponding closing brackets. Let $I_{a_1}, ..., I_{a_m} \in \mathcal{I}$ be intervals. We will consider the *timed Dyck language* $\mathcal{D}_\Sigma(I_{a_1}, ..., I_{a_m}) \subseteq \mathbb{T}(\Sigma \cup \overline{\Sigma})^+$ of timed words $w = (a_1, t_1)...(a_n, t_n)$ where $a_1...a_n$ is a sequence of correctly nested brackets and, for every $i \in \{1, ..., m\}$, the time distance between any two matching brackets a_i and $\overline{a}_i$ is in I_{a_i}. It is not difficult to see that the timed language $\mathcal{D}_\Sigma(I_{a_1}, ..., I_{a_m})$ is a timed pushdown language. We illustrate this on the following example. Let $\Sigma = \{a, b\}$, $\overline{\Sigma} = \{\overline{a}, \overline{b}\}$, $I_a = (0, 1)$ and $I_b = [0, 2]$. Consider the TPDA $\mathcal{A} = (L, \Gamma, \emptyset, L_0, E, L_f)$ with $L = L_0 = L_f = \{1\}$, $\Gamma = \{\gamma_a, \gamma_b\}$; $E = \{e_\alpha \mid \alpha \in \Sigma \cup \overline{\Sigma}\}$ such that, for $\alpha \in \Sigma$, $e_\alpha = \big(1 \xrightarrow[(\downarrow, \gamma_\alpha, [0,0])]{\alpha, \emptyset, \emptyset} 1\big)$ and $e_{\overline{\alpha}} = \big(1 \xrightarrow[(\uparrow, \gamma_\alpha, I_\alpha)]{\overline{\alpha}, \emptyset, \emptyset} 1\big)$. Note that $\mathcal{A}$ does not contain any global clocks. Then, $\mathcal{L}(\mathcal{A}) = \mathcal{D}_\Sigma(I_a, I_b)$. Consider, for instance, the timed word $w = (b, 0)(a, 0.2)(\overline{a}, 0.9)(\overline{b}, 0.9) \in \mathbb{T}(\Sigma \cup \overline{\Sigma})^+$. Then,

$$\langle 1, \varepsilon\rangle \vdash_0 \langle 1, \varepsilon\rangle \vdash_{e_b} \langle 1, (\gamma_b, 0)\rangle \vdash_{0.2} \langle 1, (\gamma_b, 0.2)\rangle \vdash_{e_a} \langle 1, (\gamma_a, 0)(\gamma_b, 0.2)\rangle$$
$$\vdash_{0.9} \langle 1, (\gamma_a, 0.9)(\gamma_b, 1.1)\rangle \vdash_{e_{\overline{a}}} \langle 1, (\gamma_b, 1.1)\rangle \vdash_{0.9} \langle 1, (\gamma_b, 2)\rangle \vdash_{e_{\overline{b}}} \langle 1, \varepsilon\rangle$$

is an accepting run of $\mathcal{A}$ with the label w. Note that here we omit the empty clock valuation of configurations.

3 Timed Matching Logic

The goal of this section is to develop a logical formalism which is expressively equivalent to TPDA defined in Sect. 2. Our new logic will incorporate Wilke's *relative distance*

Table 1. The semantics of $\text{TMSO}(\Sigma)$-formulas

$(w,\sigma) \models P_a(x)$	iff	$a_{\sigma(x)} = a$
$(w,\sigma) \models x \leq y$	iff	$\sigma(x) \leq \sigma(y)$
$(w,\sigma) \models \mathcal{X}(x)$	iff	$\sigma(x) \in \sigma(\mathcal{X})$
$(w,\sigma) \models \mathrm{d}^I(D,x)$	iff	$(\sigma(D),\sigma(x)) \in \mathrm{d}^I(w)$
$(w,\sigma) \models \mu^I(x,y)$	iff	$(\sigma(x),\sigma(y),\sigma(\mu)) \in \mu^I(w)$
$(w,\sigma) \models \varphi_1 \vee \varphi_2$	iff	$(w,\sigma) \models \varphi_1$ or $(w,\sigma) \models \varphi_2$
$(w,\sigma) \models \neg\varphi$	iff	$(w,\sigma) \models \varphi$ does not hold
$(w,\sigma) \models \exists x.\varphi$	iff	$\exists j \in \mathrm{dom}(w) : (w,\sigma[x/j]) \models \varphi$
$(w,\sigma) \models \exists X.\varphi$	iff	$\exists J \subseteq \mathrm{dom}(w) : (w,\sigma[X/J]) \models \varphi$

logic [18] for timed automata as well as logic with *matchings* [13] introduced by Lautemann, Schwentick and Thérien for context-free languages. Moreover, we augment our logic with the possibility to measure the time distance between matched positions.

Let $V_1, V_2, \mathcal{D}$ denote the countable and pairwise disjoint sets of first-order, second-order and relative distance variables, respectively. We also fix a *matching variable* $\mu \notin V_1 \cup V_2 \cup \mathcal{D}$. Let $\mathcal{U} = V_1 \cup V_2 \cup \mathcal{D} \cup \{\mu\}$.

Let Σ be an alphabet. The set $\text{TMSO}(\Sigma)$ of *timed matching MSO formulas* is defined by the grammar

$$\varphi \ ::= \ P_a(x) \mid x \leq y \mid \mathcal{X}(x) \mid \ \mathrm{d}^I(D,x) \mid \mu^I(x,y) \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists x.\varphi \mid \exists X.\varphi$$

where $a \in \Sigma$, $x,y \in V_1$, $X \in V_2$, $D \in \mathcal{D}$, $\mathcal{X} \in V_2 \cup \mathcal{D}$ and $I \in \mathcal{I}$. The formulas of the form $\mathrm{d}^I(D,x)$ are called *relative distance predicates* and the formulas of the form $\mu^I(x,y)$ are called *distance matchings*. For $\mu^{[0,\infty)}(x,y)$, we will write simply $\mu(x,y)$.

The $\text{TMSO}(\Sigma)$-formulas are interpreted over timed words over Σ and assignments of variables. Let $w \in \mathbb{T}\Sigma^+$ be a timed word. Recall that $\mathrm{dom}(w) = \{1,...,|w|\}$ is the domain of w. A $(w,\mathcal{U})$-*assignment* is a mapping $\sigma : \mathcal{U} \to \mathrm{dom}(w) \cup 2^{\mathrm{dom}(w)} \cup 2^{(\mathrm{dom}(w))^2}$ such that $\sigma(V_1) \subseteq \mathrm{dom}(w)$, $\sigma(V_2 \cup \mathcal{D}) \subseteq 2^{\mathrm{dom}(w)}$ and $\sigma(\mu) \subseteq 2^{(\mathrm{dom}(w))^2}$. Let σ be a $(w,\mathcal{U})$-assignment. For $x \in V_1$ and $j \in \mathrm{dom}(w)$, the *update* $\sigma[x/j]$ is the $(w,\mathcal{U})$-assignment defined by $\sigma[x/j](x) = j$ and $\sigma[x/j](y) = \sigma(y)$ for all $y \in \mathcal{U} \setminus \{x\}$. Similarly, for $\mathcal{X} \in V_2 \cup \mathcal{D}$ and $J \subseteq \mathrm{dom}(w)$, we define the update $\sigma[\mathcal{X}/J]$ and, for $M \subseteq (\mathrm{dom}(w))^2$, the update $\sigma[\mu/M]$.

Let $w \in \mathbb{T}\Sigma^+$ be a timed word and $I \in \mathcal{I}$ an interval. Recall that, for $j \in \mathrm{dom}(w)$ and $J \subseteq \mathrm{dom}(w)$, we write $(J,j) \in \mathrm{d}^I(w)$ if $\langle w\rangle_{i,j} \in I$ for the greatest value $i \in J \cup \{0\}$ with $i < j$. For $i,j \in \mathrm{dom}(w)$, $M \subseteq (\mathrm{dom}(w))^2$ and $I \in \mathcal{I}$, we will write $(i,j,M) \in \mu^I(w)$ if $i < j$, $(i,j) \in M$ and $\langle w\rangle_{i,j} \in I$.

Given a formula $\varphi \in \text{TMSO}(\Sigma)$, a timed word $w = (a_1,t_1)...(a_n,t_n) \in \mathbb{T}\Sigma^+$ and a $(w,\mathcal{U})$-assignment σ; the satisfaction relation $(w,\sigma) \models \varphi$ is defined inductively on the structure of φ as shown in Table 1. Here, $a \in \Sigma$, $x,y \in V_1$, $X \in V_2$, $D \in \mathcal{D}$, $\mathcal{X} \in V_2 \cup \mathcal{D}$ and $I \in \mathcal{I}$.

For $\varphi \in \text{TMSO}(\Sigma)$ and $y \in V_1$, let $\exists^{\leq 1}y.\varphi$ denote the formula $\neg\exists y.\varphi \vee \exists y.(\varphi \wedge \forall z.(z \neq y \to \neg\varphi[y/z]))$ where $z \in V_1$ does not occur in φ and $\varphi[y/z]$

is the formula obtained from φ by replacing y by z. Let $\textsc{Matching}(\mu) \in \text{tMso}(\Sigma)$ denote the formula

$$\begin{aligned}\textsc{Matching}(\mu) = &\forall x.\forall y.(\mu(x,y) \to x < y) \wedge \forall x.\exists^{\leq 1} y.(\mu(x,y) \vee \mu(y,x)) \wedge \\ &\forall x.\forall y.\forall u.\forall v.((\mu(x,y) \wedge \mu(u,v) \wedge x < u < y) \to x < v < y).\end{aligned}$$

This formula demands that a binary relation μ on a timed word domain is a *matching* (cf. [13]), i.e., it is compatible with $<$, each element of the domain belongs to at most one pair in μ and μ is noncrossing.

The set $\textsc{Tml}(\Sigma)$ of the formulas of *timed matching logic* over Σ is defined to be the set of all formulas of the form

$$\psi = \exists \mu.\exists D_1. \, ... \, \exists D_m.(\varphi \wedge \textsc{Matching}(\mu))$$

where $m \geq 0$, $D_1, ..., D_m \in \mathcal{D}$ and $\varphi \in \text{tMso}(\Sigma)$. Let $w \in \mathbb{T}\Sigma^+$ and σ be a $(w,\mathcal{U})$-assignment. Then, $(w, \sigma) \models \psi$ iff there exist $J_1, ..., J_m \subseteq \text{dom}(w)$ and a matching $M \subseteq (\text{dom}(w))^2$ such that $(w, \sigma[D_1/J_1, ..., D_m/J_m, \mu/M]) \models \varphi$. For simplicity, we will denote ψ by $\exists^{\text{match}}\mu.\exists D_1. \, ... \, \exists D_m.\varphi$.

For a formula $\psi \in \textsc{Tml}(\Sigma)$, the set $\text{Free}(\psi) \subseteq \mathcal{U}$ of *free variables* of ψ is defined as usual. We say that $\psi \in \textsc{Tml}(\Sigma)$ is a *sentence* if $\text{Free}(\psi) = \emptyset$. Note that, for a sentence ψ, the satisfaction relation $(w, \sigma) \models \psi$ does not depend on a $(w,\mathcal{U})$-assignment σ. Then, we will simply write $w \models \psi$. Let $\mathcal{L}(\psi) = \{w \in \mathbb{T}\Sigma^+ \mid w \models \psi\}$, the language *defined* by ψ. We say that a timed language $\mathcal{L} \subseteq \mathbb{T}\Sigma^+$ is $\textsc{Tml}$-*definable* if there exists a sentence $\psi \in \textsc{Tml}(\Sigma)$ such that $\mathcal{L}(\psi) = \mathcal{L}$.

Example 3.1. Consider the *timed Dyck language* $\mathcal{D}_\Sigma(I_{a_1}, ..., I_{a_m}) \subseteq \mathbb{T}(\Sigma \cup \overline{\Sigma})^+$ of Example 2.2. The timed language $\mathcal{D}_\Sigma(I_{a_1}, ..., I_{a_m})$ can be defined by the $\textsc{Tml}(\Sigma)$-sentence

$$\begin{aligned}\exists^{\text{match}}\mu.\Bigg(&\forall x.\exists y.(\mu(x,y) \vee \mu(y,x)) \wedge \\ &\forall x.\forall y.\Bigg(\mu(x,y) \to \bigvee_{j=1}^{m}(P_{a_j}(x) \wedge P_{\overline{a}_j}(y) \wedge \mu^{I_{a_j}}(x,y))\Bigg)\Bigg).\end{aligned}$$

Our main result is the following theorem.

Theorem 3.2. *Let Σ be an alphabet and $\mathcal{L} \subseteq \mathbb{T}\Sigma^+$ a timed language. Then $\mathcal{L}$ is a timed pushdown language iff $\mathcal{L}$ is* $\textsc{Tml}$*-definable.*

Note that Theorem 3.2 extends the result of [13] for context-free languages as well as the result of [18] for regular timed languages. As already mentioned in the introduction, we will use the logical characterization result for visibly pushdown languages [4]. In Sect. 4, for the convenience of the reader, we recall this result. In Sect. 5, we show a Nivat-like decomposition theorem for timed pushdown languages. Finally, in Sect. 6, we give a proof of Theorem 3.2.

It was shown in [1] that the emptiness problem for TPDA is decidable. Moreover, as we will see later, our proof of Theorem 3.2 is constructive. Then, we obtain the decidability of the satisfiability problem for our timed matching logic.

Corollary 3.3. *It is decidable, given an alphabet Σ and a sentence $\psi \in \textsc{Tml}(\Sigma)$, whether there exists a timed word $w \in \mathbb{T}\Sigma^+$ such that $w \models \psi$.*

4 Visibly Pushdown Languages

For the rest of the paper, we fix a special stack symbol $\perp$.

A *pushdown alphabet* is a triple $\tilde{\Sigma} = \langle \Sigma^{\downarrow}, \Sigma^{\#}, \Sigma^{\uparrow} \rangle$ with pairwise disjoint sets $\Sigma^{\downarrow}$, $\Sigma^{\#}$ and $\Sigma^{\uparrow}$ of *push*, *local* and *pop* letters, respectively. Let $\Sigma = \Sigma^{\downarrow} \cup \Sigma^{\#} \cup \Sigma^{\uparrow}$. A *visibly pushdown automaton (VPA)* over $\tilde{\Sigma}$ is a tuple $\mathcal{A} = (Q, \Gamma, Q_0, T, Q_f)$ where Q is a finite set of states, $Q_0, Q_f \subseteq Q$ are sets of initial resp. final states, Γ is a stack alphabet with $\perp \notin \Gamma$, and $T = T^{\downarrow} \cup T^{\#} \cup T^{\downarrow}$ is a set of transitions where $T^{\downarrow} \subseteq Q \times \Sigma^{\downarrow} \times \Gamma \times Q$ is a set of push transitions, $T^{\#} \subseteq Q \times \Sigma^{\#} \times Q$ is a set of local transitions and $T^{\uparrow} \subseteq Q \times \Sigma^{\uparrow} \times (\Gamma \cup \{\perp\}) \times Q$ is a set of pop transitions.

We define the label of a transition $\tau \in T$ depending on its sort as follows. If $\tau = (p, c, \gamma, p') \in T^{\downarrow} \cup T^{\uparrow}$ or $\tau = (p, c, p') \in T^{\#}$, we let $\mathrm{label}(\tau) = c$, so $c \in \Sigma^{\downarrow} \cup \Sigma^{\uparrow}$ resp. $c \in \Sigma^{\#}$.

A *configuration* of $\mathcal{A}$ is a pair $\langle q, u \rangle$ where $q \in Q$ and $u \in \Gamma^*$. Let $\tau \in T$ be a transition. Then, we define the transition relation $\vdash_{\tau}$ on configurations of $\mathcal{A}$ as follows. Let $c = \langle q, u \rangle$ and $c' = \langle q', u' \rangle$ be configurations of $\mathcal{A}$.

- If $\tau = (p, a, \gamma, p') \in T^{\downarrow}$, then we put $c \vdash_{\tau} c'$ iff $p = q$, $p' = q'$ and $u' = \gamma u$.
- If $\tau = (p, a, p') \in T^{\#}$, then we put $c \vdash_{\tau} c'$ iff $p = q$, $p' = q'$ and $u' = u$,
- If $\tau = (p, a, \gamma, p') \in T^{\uparrow}$ with $\gamma \in \Gamma \cup \{\perp\}$, then we put $c \vdash_{\tau} c'$ iff $p = q$, $p' = q'$ and either $\gamma \neq \perp$ and $u = \gamma u'$, or $\gamma = \perp$ and $u' = u = \varepsilon$.

We say that $c = \langle q, u \rangle$ is an *initial* configuration if $q \in Q_0$ and $u = \varepsilon$. We call c a *final* configuration if $q \in Q_f$. A *run* of $\mathcal{A}$ is a sequence $\rho = c_0 \vdash_{\tau_1} c_1 \vdash_{\tau_2} ... \vdash_{\tau_n} c_n$ where $c_0, c_1, ..., c_n$ are configurations of $\mathcal{A}$ such that c_0 is initial, c_n is final and $\tau_1, ..., \tau_n \in T$. Let $\mathrm{label}(\rho) = \mathrm{label}(\tau_1) ... \mathrm{label}(\tau_n) \in \Sigma^+$, the *label* of ρ. Let $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^+ \mid \text{there exists a run } \rho \text{ of } \mathcal{A} \text{ with } \mathrm{label}(\rho) = w\}$. We say that a language $\mathcal{L} \subseteq \Sigma^+$ is a *visibly pushdown language* over $\tilde{\Sigma}$ if there exists a VPA $\mathcal{A}$ over Σ with $\mathcal{L}(\mathcal{A}) = \mathcal{L}$.

Remark 4.1. Note that we do not demand for final configurations that $u = \varepsilon$ and we can read a pop letter even if the stack is empty (using the special stack symbol $\perp$). This permits to consider the situations where some pop letters are not balanced by push letters and vice versa.

We note that the visibly pushdown languages over $\tilde{\Sigma}$ form a proper subclass of the context-free languages over Σ, cf. [4] for further properties.

For any word $w = a_1 ... a_n \in \Sigma^+$, let $\textsc{Mask}(w) = b_1 ... b_n \in \{-1, 0, 1\}^+$ such that, for all $1 \leq i \leq n$, $b_i = 1$ if $a_i \in \Sigma^{\downarrow}$, $b_i = 0$ if $a_i \in \Sigma^{\#}$, and $b_i = -1$ otherwise. Let $\mathbb{L} \subseteq \{-1, 0, 1\}^*$ be the language which contains ε and all words $b_1 ... b_n \in \{-1, 0, 1\}^+$ such that $\sum_{j=1}^{n} b_j = 0$ and $\sum_{j=1}^{i} b_j \geq 0$ for all $i \in \{1, ..., n\}$. Here, we interpret 1 as the left parenthesis, -1 as the right parenthesis and 0 as an irrelevant symbol. Then, $\mathbb{L}$ is the set of all sequences with correctly nested parentheses.

Next, we turn to the logic $\textsc{Mso}_{\mathbb{L}}(\tilde{\Sigma})$ over the pushdown alphabet $\tilde{\Sigma}$ which extends the classical MSO logic on finite words by the binary relation which checks whether a push letter and a pop letter are matching. This logic was shown in [4] to be expressively equivalent to visibly pushdown automata. The logic $\mathrm{MSO}_{\mathbb{L}}(\tilde{\Sigma})$ is defined by the grammar

$$\varphi \ ::= \ P_a(x) \mid x \leq y \mid X(x) \mid \mathbb{L}(x,y) \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists x.\varphi \mid \exists X.\varphi$$

where $a \in \Sigma$, $x, y \in V_1$ and $X \in V_2$. The formulas in $\mathrm{MSO}_{\mathbb{L}}(\tilde{\Sigma})$ are interpreted over a word $w = a_1...a_n \in \Sigma^+$ and a variable assignment $\sigma : V_1 \cup V_2 \to \mathrm{dom}(w) \cup 2^{\mathrm{dom}(w)}$. We will write $(w,\sigma) \models \mathbb{L}(x,y)$ iff $\sigma(x) < \sigma(y)$, $a_{\sigma(x)} \in \Sigma^{\downarrow}$, $a_{\sigma(y)} \in \Sigma^{\uparrow}$ and $\mathrm{MASK}(a_{\sigma(x)+1}...a_{\sigma(y)-1}) \in \mathbb{L}$. For other formulas, the satisfaction relation is defined as usual. If φ is a sentence, then the satisfaction relation does not depend on a variable assignment and we can simply write $w \models \varphi$. For a sentence $\varphi \in \mathrm{MSO}_{\mathbb{L}}(\tilde{\Sigma})$, let $\mathcal{L}(\varphi) = \{w \in \Sigma^+ \mid w \models \varphi\}$. We say that a language $\mathcal{L} \subseteq \Sigma^+$ is $\mathrm{MSO}_{\mathbb{L}}(\tilde{\Sigma})$-*definable* if there exists a sentence $\varphi \in \mathrm{MSO}_{\mathbb{L}}(\tilde{\Sigma})$ such that $\mathcal{L}(\varphi) = \mathcal{L}$.

The following result states the expressive equivalence of visibly pushdown automata and $\mathrm{MSO}_{\mathbb{L}}$-logic.

Theorem 4.2 (Alur, Madhusudan [4]). *Let $\tilde{\Sigma} = (\Sigma^{\downarrow}, \Sigma^{\#}, \Sigma^{\uparrow})$ be a pushdown alphabet, $\Sigma = \Sigma^{\downarrow} \cup \Sigma^{\#} \cup \Sigma^{\uparrow}$, and $\mathcal{L} \subseteq \Sigma^+$ a language. Then, $\mathcal{L}$ is a visibly pushdown language over $\tilde{\Sigma}$ iff $\mathcal{L}$ is $\mathrm{MSO}_{\mathbb{L}}(\tilde{\Sigma})$-definable.*

5 Decomposition of Timed Pushdown Automata

In this section we prove a Nivat-like (cf. [15, 5]) decomposition theorem for timed pushdown automata. This result establishes a connection between timed pushdown languages and visibly pushdown languages. We will use this theorem for the proof of our Theorem 3.2.

The key idea is to consider a timed pushdown language as a renaming of a timed pushdown language over an extended alphabet which encodes the information about clocks and stack; on the level of this extended alphabet we can separate the setting of visibly pushdown languages from the timed setting. Our separation technique appeals to the partitioning of $\mathbb{R}_{\geq 0}$ into finitely many intervals; this finite partition will be used for the construction of the desired extended alphabet.

We fix an alphabet Σ (which we will understand as the alphabet of Theorem 3.2).

Consider a TPDA $\mathcal{A} = (L, \Gamma, C, L_0, E, L_f)$ over Σ. We may assume that $C = \{1, ..., m\}$. Let $X \subseteq \mathbb{N}$ be the set of all natural numbers which are lower or upper bounds of some interval $I \in \mathcal{I}$ appering in E (either in a clock constraint or in a stack command). Clearly, X is a finite set. Let $k = \max(X)$ (if $X = \emptyset$, then we let $k = 0$). Let $\mathbb{P}(k) = \{[0,0], (0,1), [1,1], (1,2), ..., [k,k], (k,\infty)\} \subseteq 2^{\mathcal{I}}$, the k-*interval partition* of $\mathbb{R}_{\geq 0}$. Note that $\mathbb{P}(k)$ is a finite non-empty set since $[0,0] \in \mathbb{P}(k)$ for any $k \in \mathbb{N}$. The extended alphabet for such a TPDA $\mathcal{A}$ will be a pushdown alphabet augmented with the following additional components reflecting the performance of the clocks and the stack:

– the partition of the pushdown alphabet will be induced by the component $\{\downarrow, \#, \uparrow\}$;
– for every global clock $c \in \{1, ..., m\}$, we add two components:

  - a component $\mathbb{P}(k)$ which indicates the interval containing a value of the clock c before taking an edge of $\mathcal{A}$;
  - a component $\{0,1\}$ which indicates whether the clock c was reset after taking an edge of $\mathcal{A}$ or not;
- to handle the local clocks of the stack, we add the component $\mathbb{P}(k)$ which indicates:
  - for all push letters (i.e. with the $\downarrow$-component) the interval containing an initial value of the local clock which will be pushed into the stack;
  - for all pop letters (i.e. with the $\uparrow$-component) the interval containing a value of the clock on the top of the stack.
  - for all letters with $\#$, the stack is not touched and the $\mathbb{P}(k)$-component of this letter is useless. So in this case we can restrict ourselves to the interval $[0,0]$.

Formally, we consider the pushdown alphabet $\tilde{\mathcal{R}}_{m,k} = \langle \mathcal{R}^{\downarrow}_{m,k}, \mathcal{R}^{\#}_{m,k}, \mathcal{R}^{\uparrow}_{m,k} \rangle$ where, for $\delta \in \{\downarrow, \#, \uparrow\}$: $\mathcal{R}^{\delta}_{m,k} = \Sigma \times (\mathbb{P}(k))^m \times \{0,1\}^m \times \mathbb{P}(k) \times \{\delta\}$. Let $\mathcal{R}_{m,k} = \bigcup_{\delta \in \{\downarrow,\#,\uparrow\}} \mathcal{R}^{\delta}_{m,k}$.

Now consider a "simple" TPDA over $\mathcal{R}_{m,k}$ with a single state, a single stack symbol and m clocks $\{1,...,m\}$; for every letter in $\mathcal{R}_{m,k}$, this TPDA processes the clocks and the stack according to the information encoded in the additional components of $\mathcal{R}_{m,k}$. Let $\mathcal{T}_{m,k} \subseteq \mathbb{T}(\mathcal{R}_{m,k})^+$ denote the timed language accepted by this TPDA.

For all $I, I' \in \mathcal{I}$, let $I - I' = \{x - x' \mid x \in I \text{ and } x' \in I'\}$. The timed language $\mathcal{T}_{m,k}$ can be described formally as follows. Let $w = (b_1,t_1)...(b_n,t_n) \in \mathbb{T}(\mathcal{R}_{m,k})^+$ where, for all $i \in \{1,...,n\}$, $b_i = (a_i, \overline{G}_i, \overline{R}_i, s_i, \delta_i)$ with $a_i \in \Sigma$, $\overline{G}_i = (g_i^1, ..., g_i^m) \in (\mathbb{P}(k))^m$ (corresponds to the intervals for the global clocks), $\overline{R}_i = (r_i^1, ..., r_i^m) \in \{0,1\}^m$ (corresponds to the resets of global clocks), $s_i \in \mathbb{P}(k)$ (corresponds to the intervals for the local clocks in the stack), $\delta_i \in \{\downarrow, \#, \uparrow\}$ and $t_i \in \mathbb{R}_{\geq 0}$. Then, $w \in \mathcal{T}_{m,k}$ iff the following hold:

- $\text{MASK}(b_1...b_n) \in \mathbb{L}$ (with respect to the pushdown alphabet $\tilde{\mathcal{R}}_{m,k}$);
- for all $i \in \{1,...,n\}$ and $j \in \{1,...,m\}$, letting $r_0^j = 1$, we have $\langle w \rangle_{i',i} \in g_i^j$ for the greatest $i' \in \{0,1,...,i-1\}$ with $r_{i'}^j = 1$;
- for all $i, i' \in \{1,...,n\}$ with $i < i'$, $\delta_i = \downarrow$, $\delta_{i'} = \uparrow$ and $\text{MASK}(b_{i+1}...b_{i'-1}) \in \mathbb{L}$, we have $\langle w \rangle_{i,i'} \in s_{i'} - s_i$.

Clearly, the timed language $\mathcal{T}_{m,k}$ is a non-empty timed pushdown language. Let Δ be an alphabet, $\mathcal{L} \subseteq \Delta^+$ a language and $\mathcal{L}' \subseteq \mathbb{T}\Delta^+$ a timed language. Let $(\mathcal{L} \cap \mathcal{L}') \subseteq \mathbb{T}\Delta^+$ be the "restriction" of $\mathcal{L}'$ to $\mathcal{L}$, i.e., the timed language consisting of all timed words $w = (b_1,t_1)...(b_n,t_n) \in \mathcal{L}'$ such that $b_1...b_n \in \mathcal{L}$. Let Δ, Δ' be alphabets and $h : \Delta \to \Delta'$ a renaming. For a timed word $w = (b_1,t_1)...(b_n,t_n) \in \mathbb{T}\Delta^+$, let $h(w) = (h(b_1),t_1)...(h(b_n),t_n)$. Then, for a timed language $\mathcal{L} \subseteq \mathbb{T}\Delta^+$, let $h(\mathcal{L}) = \{h(w) \mid w \in \mathcal{L}\}$, so $h(\mathcal{L}) \subseteq \mathbb{T}(\Delta')^+$.

Now we formulate our decomposition theorem. This result permits to separate the discrete part of TPDA from their timed part. We show that the discrete part can be described by visibly pushdown languages whereas the timed part can be described by means of timed languages $\mathcal{T}_{m,k}$ which have the following interesting property. We can decide whether a timed word w belongs to $\mathcal{T}_{m,k}$ by analyzing the components of w. In contrast, if we have a TPDA $\mathcal{A}$ and can use it only as a "black box", then we cannot say whether a timed word w is accepted by this TPDA $\mathcal{A}$ without passing w through $\mathcal{A}$.

Theorem 5.1. *Let Σ be an alphabet and $\mathcal{L} \subseteq \mathbb{T}\Sigma^+$ a timed language. Then the following are equivalent.*

(a) *$\mathcal{L}$ is a timed pushdown language.*
(b) *There exist $m, k \in \mathbb{N}$, a renaming $h : \mathcal{R}_{m,k} \to \Sigma$, and a visibly pushdown language $\mathcal{L}' \subseteq (\mathcal{R}_{m,k})^+$ over the pushdown alphabet $\tilde{\mathcal{R}}_{m,k}$ such that $\mathcal{L} = h(\mathcal{L}' \cap \mathcal{T}_{m,k})$.*

The proof idea for the implication (b) $\Rightarrow$ (a) is the following. Since we work here with the extended alphabet $\mathcal{R}_{m,k}$ (which corresponds to m global clocks), every letter of this alphabet contains the information about the guards and resets of global clocks as well as performance of the timed stack. Then, we can construct a TPDA for the intersection $\mathcal{L}' \cap \mathcal{T}_{m,k}$ by rewriting the transitions of a VPA for $\mathcal{L}'$ (over the extended pushdown alphabet $\tilde{\mathcal{R}}_{m,k}$) as edges of a TPDA (over the extended alphabet $\mathcal{R}_{m,k}$). After that, using the mapping h, we rename the labels of edges of the constructed TPDA and obtain a TPDA for the desired timed language $h(\mathcal{L}' \cap \mathcal{T}_{m,k})$.

The proof idea for the implication (a) $\Rightarrow$ (b) is illustrated in the following example.

Example 5.2. Consider the TPDA $\mathcal{A} = (L, \Gamma, C, L_0, E, L_f)$ over the alphabet $\Sigma = \{a, b\}$ depicted in Fig. 1. Formally, $\mathcal{A}$ is defined as follows:

– $L = \{1, 2\}$, $L_0 = \{1\}$, $L_f = \{2\}$, $\Gamma = \{\gamma\}$, $C = \{x\}$;
– E consists of the following edges: $1 \xrightarrow[(\downarrow,\gamma,(0,1))]{a,\textsc{True},\emptyset} 1$, $1 \xrightarrow[\#]{b,\textsc{True},\{x\}} 1$, $1 \xrightarrow[\#]{a,x\geq 1,\emptyset} 2$, $2 \xrightarrow[(\uparrow,\gamma,[1,1])]{a,\textsc{True},\emptyset} 2$.

The timed language $\mathcal{L}(\mathcal{A})$ can be decomposed in the following way. As already mentioned before, m is the number of global clocks of $\mathcal{A}$, i.e., $m = 1$ and k is the maximal constant appearing in the intervals of $\mathcal{A}$, i.e., $k = 1$. Then, $\mathcal{R}_{1,1} = \Sigma \times \mathbb{P}(1) \times \{0, 1\} \times \mathbb{P}(1) \times \{\downarrow, \#, \uparrow\}$. Then, $\mathcal{L} = h(\mathcal{L}' \cap \mathcal{T}_{m,k})$ where:

– $h : \mathcal{R}_{1,1} \to \Sigma$ is the projection to the first component;
– the language $\mathcal{L}' \subseteq (\mathcal{R}_{1,1})^+$ is recognized by the visibly pushdown automaton $\mathcal{A}_{\mathcal{L}'} = (L, \Gamma, L_0, T', L_f)$ over the pushdown alphabet $\tilde{\mathcal{R}}_{1,1}$ depicted in Fig. 1. Here, the component $*$ in the transition labels means an arbitrary element of $\mathbb{P}(1)$ and idle $= [0, 0]$ denotes the idle stack interval for the letters with the $\#$-component. We also would like to point out that every edge of the TPDA $\mathcal{A}$ is simulated by several transitions of the VPA $\mathcal{A}_{\mathcal{L}'}$. For instance, we simulate the edge from the location 1 to the location 2 of the TPDA $\mathcal{A}$ by two edges, since, for the condition $x \geq 1$, we have in the partition $\mathbb{P}(1)$ two intervals $[1, 1], (1, \infty)$ satisfying this condition.
– The timed language $\mathcal{T}_{1,1} \subseteq \mathbb{T}(\mathcal{R}_{1,1})^+$ (as defined before) can be recognized by the TPDA $\mathcal{A}_{\mathcal{T}_{1,1}} = (\{1\}, \{\alpha\}, C, \{1\}, E', \{1\})$ depicted in Fig. 1. Here, I, J are arbitrary intervals in $\mathbb{P}(1)$. By using a new stack letter α, we want to point out that the stack alphabet of the TPDAs for $\mathcal{T}_{m,k}$ is a singleton alphabet and does not depend on Γ.

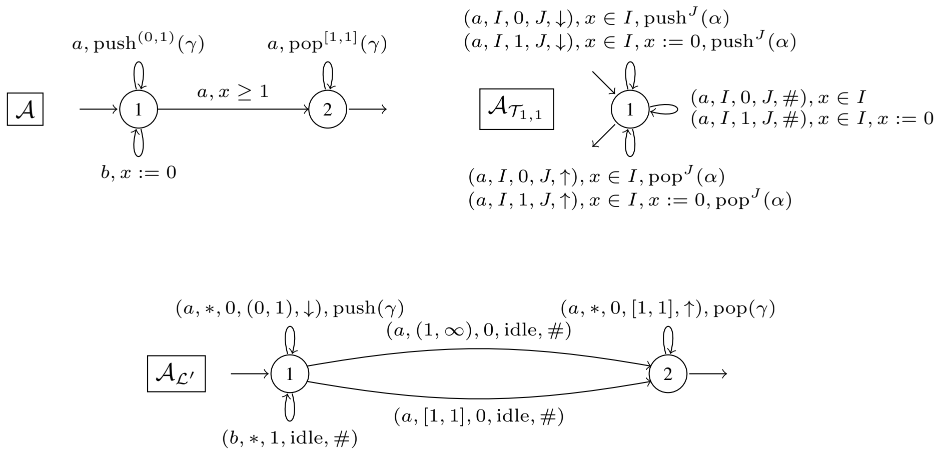
$\mathcal{A}$
$a, \mathrm{push}^{(0,1)}(\gamma)$
$a, \mathrm{pop}^{[1,1]}(\gamma)$
$a, x \geq 1$
$b, x := 0$
$\mathcal{A}_{\mathcal{T}_{1,1}}$
$(a, I, 0, J, \downarrow), x \in I, \mathrm{push}^J(\alpha)$
$(a, I, 1, J, \downarrow), x \in I, x := 0, \mathrm{push}^J(\alpha)$
$(a, I, 0, J, \#), x \in I$
$(a, I, 1, J, \#), x \in I, x := 0$
$(a, I, 0, J, \uparrow), x \in I, \mathrm{pop}^J(\alpha)$
$(a, I, 1, J, \uparrow), x \in I, x := 0, \mathrm{pop}^J(\alpha)$
$\mathcal{A}_{\mathcal{L}'}$
$(a, *, 0, (0,1), \downarrow), \mathrm{push}(\gamma)$
$(a, *, 0, [1,1], \uparrow), \mathrm{pop}(\gamma)$
$(a, (1, \infty), 0, \mathrm{idle}, \#)$
$(a, [1,1], 0, \mathrm{idle}, \#)$
$(b, *, 1, \mathrm{idle}, \#)$
1
2

Fig. 1. TPDA $\mathcal{A}$, $\mathcal{A}_{\mathcal{L}'}$ and $\mathcal{A}_{\mathcal{T}_{1,1}}$ of Example 5.2

Remark 5.3. As it can be observed from the proof of Theorem 5.1, instead of the k-interval partition $\mathbb{P}(k)$, for every global clock or the timed stack, one could take a partition induced by bounds of the intervals which correspond to this clock or timed stack. For instance, if the intervals $(0, 1)$ and $(8, 15)$ appear in the commands for the timed stack, then we could take the partition $\{[0, 0], (0, 1), [1, 1], (1, 8), [8, 8], (8, 15), [15, 15], (15, \infty)\}$. However, for the simplicity of our notations, we considered the same partition $\mathbb{P}(k)$ for all global clocks and the timed stack.

As a corollary of Theorem 5.1 and its proof, we deduce a decomposition theorem for timed automata. These may be considered as TPDA whose sets of push and pop edges are empty (and hence a stack alphabet is irrelevant for their definition). We slightly modify the extended alphabet needed for the decomposition by excluding the components relevant for the stack. Moreover, instead of visibly pushdown languages we consider classical regular languages. For $m, k \in \mathbb{N}$, let $\mathcal{R}^0_{m,k} = \Sigma \times (\mathbb{P}(k))^m \times \{0, 1\}^m$. We define the timed language $\mathcal{T}^0_{m,k} \subseteq \mathbb{T}(\mathcal{R}^0_{m,k})^+$ as follows. Let $w = (b_1, t_1)...(b_n, t_n) \in \mathbb{T}(\mathcal{R}^0_{m,k})^+$ where, for all $i \in \{1, ..., n\}$, $b_i = (a_i, (g^1_i, ..., g^m_i), (r^1_i, ..., r^m_i))$ with $a_i \in \Sigma$, $g^1_i, ..., g^m_i \in \mathbb{P}(k)$ and $r^1_i, ..., r^m_i \in \{0, 1\}$. Then, $w \in \mathcal{T}^0_{m,k}$ iff, for all $i \in \{1, ..., n\}$ and $j \in \{1, ..., m\}$, letting $r^j_0 = 1$, we have $\langle w \rangle_{i',i} \in g^j_i$ for the greatest $i' \in \{0, 1, ..., i - 1\}$ with $r^j_{i'} = 1$.

Corollary 5.4. *Let Σ be an alphabet and $\mathcal{L} \subseteq \mathbb{T}\Sigma^+$ a timed language. Then the following are equivalent.*

(a) $\mathcal{L}$ is recognizable by a timed automaton.
(b) There exist $m, k \in \mathbb{N}$, a renaming $h : \mathcal{R}^0_{m,k} \to \Sigma$ and a regular language $\mathcal{L}' \subseteq (\mathcal{R}^0_{m,k})^+$ such that $\mathcal{L} = h(\mathcal{L}' \cap \mathcal{T}^0_{m,k})$.

6 Definability Equals Recognizability

In this section, we prove Theorem 3.2.

First, we show that TML-definable timed languages are pushdown recognizable. Let $\psi = \exists^{\mathrm{match}}\mu.\exists D_1. \ ... \ \exists D_m.\varphi \in \mathrm{TML}(\Sigma)$ with $m \geq 0$. We may assume that $D_1, ..., D_m \in \mathcal{D}$ are pairwise distinct variables.

We wish to use Theorem 5.1. As preparation for this, we prove the following technical lemma which provides a decomposition of a TML-sentence. For the definitions of $\mathcal{R}_{m,k}$, $\tilde{R}_{m,k}$ and $\mathcal{T}_{m,k}$ we refer the reader to the previous section. To transform ψ into a TPDA, we apply Theorems 5.1 and 4.2 and the following lemma which decomposes the TML-sentence ψ.

Lemma 6.1. *Let $\psi \in \mathrm{TML}(\Sigma)$ be a sentence as defined above. Then, there exist $k \in \mathbb{N}$, a renaming $h : \mathcal{R}_{m,k} \to \Sigma$ and a sentence $\varphi^* \in \mathrm{MSO}_{\mathbb{L}}(\tilde{\mathcal{R}}_{m,k})$ such that $\mathcal{L}(\psi) = h(\mathcal{L}(\varphi^*) \cap \mathcal{T}_{m,k})$.*

Proof (Sketch). For decomposition, we will consider the extended alphabet $\mathcal{R}_{m,k}$ where m is the number of relative distance variables of ψ and k is the maximal natural number which is a lower or upper bound of some interval appearing in ψ (if ψ does not contain any intervals, then we let $k = 0$). So, our extended alphabet is $\Sigma \times (\mathbb{P}(k))^m \times \{0,1\}^m \times \mathbb{P}(k) \times \{\downarrow, \#, \uparrow\}$. The additional components will have the following meaning.

- Using a vector $(g^1, ..., g^m) \in (\mathbb{P}(k))^m$, we will encode the intervals which appear in the relative distance predicates of ψ. Here the component g^i $(i \in \{1, ..., m\})$ is responsible for the relative distance predicates with the variable D_i.
- Using a vector $(r^1, ..., r^m) \in \{0,1\}^m$, we will implement the standard Büchi-encoding of the variables $D_1, ..., D_m$.
- Using the component $\mathbb{P}(k) \times \{\downarrow, \#, \uparrow\}$, we will model quantitative matchings. Here $\mathbb{P}(k)$ is responsible for the intervals of quantitative matchings. The component $\{\downarrow, \#, \uparrow\}$ will have the following task. If a position does not belong to any pair in a matching relation, then it is marked by $\#$. If a position is on the left side in a matched pair, then it is marked by $\downarrow$. If a position is on the right side in a matched pair, then it is marked by $\uparrow$. □

Then, the properties described informally above can be expressed by means of an $\mathrm{MSO}_{\mathbb{L}}(\tilde{\mathcal{R}}_{m,k})$-sentence φ^*.

The following example illustrates the proof of Lemma 6.1.

Example 6.2. Let $\Sigma = \{a, b\}$ and

$$\psi = \exists^{\mathrm{match}}\mu.\exists D.\forall x.(D(x) \wedge [(\exists y.\mu^{(1,\infty)}(x,y)) \vee \mathrm{d}^{(0,1]}(D,x) \vee P_b(x)]).$$

In this example, we have $m = 1$ and $k = 1$, i.e. the extended alphabet is $\mathcal{R}_{1,1} = \Sigma \times \mathbb{P}(k) \times \{0,1\} \times \mathbb{P}(k) \times \{\downarrow, \#, \uparrow\}$. As a renaming $h : \mathcal{R}_{m,k} \to \Sigma$, we take the projection to the Σ-component. The sentence $\varphi^* \in \mathrm{MSO}_{\mathbb{L}}(\tilde{\mathcal{R}}_{1,1})$ is defined as $\forall x.(\varphi_1 \wedge [(\exists y.\varphi_2) \vee \varphi_3 \vee \varphi_4])$ where $\varphi_1 = P_{(*,*,1,*,*)}(x)$, $\varphi_2 = \mathbb{L}(x,y) \wedge$

$P_{(*,*,*,[0,0],\downarrow)}(y) \wedge P_{(*,*,*,(1,\infty),\uparrow)}(y)$, $\varphi_3 = P_{(*,*,(0,1),*,*)}(x) \vee P_{(*,*,[1,1],*,*)}(x)$ and $\varphi_4 = P_{(b,*,*,*,*)}(x)$. Here, we denote by $*$ the components which can take arbitrary values from their domains. Then, $\mathcal{L}(\psi) = h(\mathcal{L}(\varphi^*) \cap \mathcal{T}_{m,k})$.

The part "recognizability implies definability" of Theorem 3.2 follows from Theorems 5.1 and 4.2 and the next lemma.

Lemma 6.3. *Let Σ be an alphabet, $m, k \in \mathbb{N}$, $h : \mathcal{R}_{m,k} \to \Sigma$ a renaming, and $\varphi \in \mathrm{MSO}_{\mathbb{L}}(\tilde{\mathcal{R}}_{m,k})$ a sentence. Then, there exists a sentence $\psi \in \mathrm{TML}(\Sigma)$ such that $\mathcal{L}(\psi) = h(\mathcal{L}(\varphi) \cap \mathcal{T}_{m,k})$.*

Proof (Sketch). For the proof, we follow a similar approach as in the proof of Theorem 6.6 of [10]. Let $\Gamma = \mathcal{R}_{m,k}$. The desired sentence ψ is constructed as

$$\psi = \exists^{\text{match}}\mu.\exists D_1. \,...\, D_m.\exists X_1. \,...\, \exists X_{|\Gamma|}.(\varphi^* \wedge \textsc{Partition} \wedge \textsc{Renaming} \wedge \xi_{\mathcal{T}_{m,k}})$$

where:

- $D_1, ..., D_m \in \mathcal{D}$ are relative distance variable modeling the behavior of global clocks in the timed language $\mathcal{T}_{m,k}$;
- $X_1, ..., X_{|\Gamma|} \in V_2$ are variables describing the renaming h (i.e., we store in these second-order variables the positions of the letters of the extended alphabet Γ before the renaming);
- φ^* is obtained from φ by replacing $\mathbb{L}$ by μ, and all $P_\gamma(x)$ by $P_{h(\gamma)}(x) \wedge X_\gamma(x)$;
- the formula $\textsc{Partition}$ demands that values of $X_1, ..., X_{|\Gamma|}$ form a partition of the domain;
- the formula $\textsc{Renaming}$ correlates values of $X_1, ..., X_{|\Gamma|}$ with the labels of an input word;
- the formula $\xi_{\mathcal{T}_{m,k}}$ describes the properties of $\mathcal{T}_{m,k}$ (using relative distance predicates and quantitative matchings). □

Remark 6.4. Alternatively, the direction "recognizability implies definability" of Theorem 3.2 can be proved by a direct translation of $\mathcal{A}$ into ψ. However, by using Theorem 5.1, it suffices to describe a simpler timed language $\mathcal{T}_{m,k}$ and a projection h to adopt the logical description of a visibly pushdown language of [4]. In particular, here we do not have to describe some technical details like initial, final states as well as concatenations of transitions.

7 Conclusion and Future Work

In this paper, we introduced a timed matching logic and showed that this logic is equally expressive as timed pushdown automata (and hence the satisfiability problem for our timed matching logic is decidable). When proving our main result, we showed a Nivat-like decomposition theorem for timed pushdown automata. This theorem seems to be the first algebraic characterization of timed pushdown languages and may be of independent interest, e.g., it could be helpful to transfer further results from the discrete setting to the timed setting. Based on the ideas presented in [9, 10, 14, 16] and the ideas of this paper, our ongoing research concerns a logical characterization for *weighted timed pushdown automata* [2]. It could be also interesting to investigate such an extension of timed pushdown automata where each edge permits to push or pop several stack elements.

References

[1] Abdulla, P.A., Atig, M.F., Stenman, J.: Dense-timed pushdown automata. In: LICS 2012, pp. 35–44. IEEE Computer Society (2012)

[2] Abdulla, P.A., Atig, M.F., Stenman, J.: Computing optimal reachability costs in priced dense-timed pushdown automata. In: LATA 2014. LNCS, vol. 8370, pp. 62–75. Springer (2014)

[3] Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2). 183–235 (1994)

[4] Alur, R., Madhusudan, P.: Visibly pushdown languages. In: STOC 2004, pp. 202–211. ACM (2004)

[5] Berstel, J.: Transductions and Context-Free Languages. Teubner Studienbücher: Informatik. Teubner, Stuttgart (1979)

[6] Bouajjani, A., Echahed, R., Robbana, R.: On the automatic verification of systems with continuous variables and unbounded discrete data structures. In: Hybrid Systems II. LNCS, vol. 999, pp. 64–85. Springer (1995)

[7] Büchi, J.R.: Weak second order arithmetic and finite automata. Zeitschrift für Mathematische Logik und Grundlagen der Informatik 6, 66–92 (1960)

[8] Dang, Z.: Pushdown timed automata: a binary reachability characterization and safety verification. Theoretical Computer Science 302, 93–121 (2003)

[9] Droste, M., Gastin, P.: Weighted automata and weighted logics. Theoret. Comp. Sci. 380(1-2), 69–86 (2007)

[10] Droste, M., Perevoshchikov, V.: A Nivat theorem for weighted timed automata and relative distance logic. In: ICALP 2014. LNCS, pp. 171–182. Springer (2014)

[11] Elgot, C.C.: Decision problems of finite automata design and related arithmetics. Trans. Amer. Math. Soc. 98, 21–51 (1961)

[12] Emmi, M., Majumdar, R.: Decision problems for the verification of real-time software. In: HSCC 2006. LNCS, vol. 3927, pp. 200–211. Springer (2006)

[13] Lautemann, C., Schwentick, T., Thérien, D.: Logics for context-free languages. In: CSL 1994. LNCS, vol. 933, pp. 205–216. Springer (1994)

[14] Mathissen, Ch.: Weighted logics for nested words and algebraic formal power series. In: ICALP 2008. LNCS, vol. 5126, pp 221–232. Springer (2008)

[15] Nivat, M.: Transductions des langages de Chomsky. Ann. de l'Inst. Fourier 18, 339–456 (1968)

[16] Quaas, K.: MSO logics for weighted timed automata. Formal Methods in System Design 38(3), 193–222 (2011)

[17] Thatcher, J.W., Wright, J.B.: Generalized finite automata theory with an application to a decision problem of second-order logic. Mathematical System Theory, 2:57–81 (1968)

[18] Wilke, T.: Specifying timed state sequences in powerful decidable logics and timed automata. In: Formal Techniques in Real-Time and Fault-Tolerant Systems 1994. LNCS, vol. 863, pp. 694 – 715. Springer (1994)