Event-Clock Visibly Pushdown Automata

Nguyen Van Tang and Mizuhito Ogawa

Graduate School of Information Science
Japan Advanced Institute of Science and Technology
{tang_nguyen,mizuhito}@jaist.ac.jp

Abstract. We introduce the class of *event-clock* visibly pushdown automata (ECVPAs) as an extension of event-clock automata. The class of ECVPAs, on one hand, can model simple real-time pushdown systems and, on the other hand, is determinizable and closed under Boolean operations. We also show that for a *timed* visibly pushdown automaton A and an ECVPA B, the inclusion problem $L(A) \subseteq L(B)$ is decidable.

1 Introduction

Timed automata (TAs) were introduced by Alur and Dill in [2], and have become a standard modeling formalism for real-time systems. A timed automaton is a finite automaton augmented with a finite set of real-valued clocks, in which constraints on the clocks are used to restrict the behaviors of an automaton. The theory of timed automata allows the solution of certain verification problems for real-time systems [2, 10, 6], e.g., reachability and safety properties. These solutions have been implemented in automatic tools such as UPPAAL [1]. However, the general verification problems (i.e., language inclusion) for timed automata is undecidable. Therefore, for the verification purpose, one has to work either with deterministic specifications or with a restricted class of timed automata which has the required closure properties. One such restricted case is the class of *event-clock automata* (ECAs) [3, 12, 13]. The key feature of these automata is that they have a pair of implicit clocks associated with each input symbol. The event clocks record the time elapsed since the last occurrence of the associated symbol, as well as the time that will elapse before the next occurrence of the associated symbol. When an ECA reads a timed word, clock valuations depend only on the input word itself rather than on the choice of nondeterministic transitions. Hence, ECAs are determinizable and closed under Boolean operations.

During the last years, there has been much extensive research on the inclusion problem for timed automata [11, 8, 7]. In particular, it was shown that the inclusion problem $L(A) \subseteq L(B)$, for timed automata A and B, becomes *decidable* if B has at most *one clock* [11]. The key idea of the proof is to encode this inclusion problem as the reachability problem for well-structured transition systems. However, over infinite timed words, one clock is enough to make the inclusion problem undecidable [1].

[1] http://www.uppaal.com/

A *timed pushdown automaton* (TPDA) [5] is a timed automaton augmented with a pushdown stack. Decision problems for TPDAs such as emptiness is decidable [5]. However, the inclusion problem for TPDAs is undecidable, since the corresponding problem is already undecidable for pushdown automata. One, therefore, has to deal with formalism of less expressive power. One such candidate is the class of *visibly* pushdown automata (VPAs) [4], in which the stack pushes and pops are determined explicitly by an input alphabet. VPAs are closed under all Boolean operations, and the inclusion problem for VPAs is decidable. Motivated by real-time software verification, Emmi and Majumdar [8] introduced *timed* visibly pushdown automata (TVPAs) as the timed extension of VPAs. However, for TVPAs A and B, the inclusion problem $L(A) \subseteq L(B)$ is *undecidable* even when B has exactly *one clock* [8].

In this paper, inspired by the ideas of ECAs [3] and VPAs [4], we introduce the class of *event-clock visibly pushdown automata* (ECVPAs). The class of ECVPAs is expressive enough to specify common context-free real-time properties such as "if p holds when a procedure is invoked, then the procedure must return within d time units and q must hold at the return state". Besides, the class of ECVPAs is closed under all Boolean operations. Our results are summarized as follows:

- We show the essence behind the notion of event clocks is that every ECVPA can be translated into an untimed VPA, which interprets timing constraints symbolically, and vice-versa. Therefore, the closure properties and the decidability results of ECVPAs can be reduced to those of VPAs.
- We use the translation technique to prove that the inclusion problem $L(A) \subseteq L(B)$ for a TVPA A and an ECVPA B is decidable.
- We show that class of duration automata (DAs) [14] is a special case of ECVPAs. Thus, the inclusion problem for DAs is decidable.

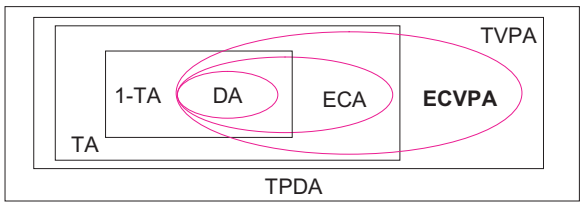

Fig. 1. Relationships between involved classes of TPDAs

Structure of the paper. In Section 2 we introduce the class of ECVPAs. Section 3 shows how to translate ECVPAs into untimed VPAs that express timing constraints symbolically, and vice-versa. The closure properties and decidability results of ECVPAs are reduced to those of VPAs. Section 4 presents subcases of TPDAs with decidable inclusion problems. Section 5 reports some related works. Finally, we conclude the paper in Section 6.

2 Event-Clock Visibly Pushdown Automata

2.1 Event clocks

In this subsection we give a brief description for event clocks. Readers are referred to [3] for more details.

Let R^+ and Q^+ denote the set of non-negative real numbers and non-negative rational numbers, respectively. Given a finite alphabet Σ, a *timed word* $\bar{w}$ over Σ is a finite sequence $(a_0, t_0)(a_1, t_1) \cdots (a_n, t_n)$ of symbols $a_i \in \Sigma$ that are paired with nonnegative real numbers $t_i \in R^+$ such that the sequence $\bar{t} = t_0 t_1 \cdots t_n$ of time-stamps is nondecreasing (i.e., $t_i \leq t_{i+1}$ for all $0 \leq i < n$). We denote the timed word $\bar{w}$ by the pair $(\bar{a}, \bar{t})$, where $\bar{a} \in \Sigma^*$ is an untimed word over Σ. The set of all finite timed words over Σ is denoted by $T\Sigma^*$. A *timed language* is a set of timed words.

Definition 1 (Event Clocks [3]). *For each symbol $a \in \Sigma$, we use two implicit clocks x_a (*event-recording*) and y_a (*event-predicting*). Along a timed word, the clock x_a measures the time since the last occurrence of symbol a, and y_a measures the time to the next occurrence of symbol a. If there are no last (resp., next) occurrence of a, the value of x_a (resp., y_a) is "undefined", denoted by $\perp$.*

Remark 1. The notation $\perp$ denotes the "undefined" value, and the *bottom-of-stack* symbol of visibly pushdown automata (defined in the next subsection).

Let $C_\Sigma = \{x_a | a \in \Sigma\} \cup \{y_a | a \in \Sigma\}$ be the set of event-recording and event-predicting clocks. Define $R^+_\perp = R^+ \cup \{\perp\}$ and $Q^+_\perp = Q^+ \cup \{\perp\}$.

Definition 2 (Event-clock valuation [3]). *For each timed word $\bar{w} = (a_0, t_0)(a_1, t_1) \cdots (a_n, t_n)$, a* clock valuation *over $\bar{w}$ is a function $\nu^{\bar{w}}_j : C_\Sigma \to R^+_\perp$ which specifies the values of the clocks (in C_Σ) at position j in $\bar{w}$.*

$$\nu^{\bar{w}}_j(x_a) = \begin{cases} t_j - t_i & \text{If } \exists i < j : a_i = a, \text{and } \forall k : i < k < j \Rightarrow a_k \neq a \\ \perp & \text{otherwise} \end{cases}$$

$$\nu^{\bar{w}}_j(y_a) = \begin{cases} t_{i'} - t_j & \text{If } \exists i' > j : a_{i'} = a, \text{and } \forall l : j < l < i' \Rightarrow a_l \neq a \\ \perp & \text{otherwise} \end{cases}$$

Definition 3 (Event-clock constraint [3]).

- *The* clock constraints *compare clock values to $Q^+_\perp$, i.e, to rational constants or to the special value $\perp$. The clock constraints over C_Σ are interpreted with respect to the clock-valuation function ν from C_Σ to $R^+_\perp$: the atom $\perp \leq \perp$ evaluates to* True*, and all other comparisons that involve $\perp$ (e.g., $\perp \geq 3$) evaluate to* False.
- *For simplicity, let $\Phi(C_\Sigma)$ denote the set of event-clock constraints over C_Σ.*
- *For the clock-valuation ν and an event-clock constraint $\varphi \in \Phi_\Sigma$, we write $\nu \models \varphi$ (resp., $\nu \not\models \varphi$) to denote that according to ν the constraint φ evaluates to* True *(resp.,* False*).*

2.2 Event-Clock Visibly Pushdown Automata

A *pushdown alphabet* is a set $\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i$ that comprises three *disjoint* finite alphabets in which Σ_c is a finite set of *calls*, Σ_r is a finite set of *returns*, and Σ_i is a finite set of *internal symbols*. We formally define *event-clock visibly pushdown automata* over the pushdown alphabet Σ as follows:

Definition 4 (Event-clock visibly pushdown automaton). *An* event-clock visibly pushdown automaton *(ECVPA) on finite timed words over Σ is a tuple $M = \langle Q, \Sigma, Q_0, \Gamma, \Delta, F\rangle$, where Q is a finite set of locations, $Q_0 \subseteq Q$ is a finite set of initial locations, Γ is a finite stack alphabet that contains a special symbol $\perp$ (bottom-of-stack symbol), $F \subseteq Q$ is a set of final locations, and $\Delta = \Delta_c \cup \Delta_r \cup \Delta_i$ is the transition relation,*

- $\Delta_c \subseteq Q \times \Sigma_c \times \Phi(C_\Sigma) \times Q \times (\Gamma \backslash \{\perp\})$ *is a push-transition relation*
- $\Delta_r \subseteq Q \times \Sigma_r \times \Phi(C_\Sigma) \times \Gamma \times Q$ *is a pop-transition relation*
- $\Delta_i \subseteq Q \times \Sigma_i \times \Phi(C_\Sigma) \times Q$ *is an internal-transition relation.*

The intuition behind the transition relation is briefly explained as follows:

- $(q, a, \varphi, q', \gamma) \in \Delta_c$ is a push-transition, where on reading a when the clock valuation satisfies φ the symbol γ is pushed onto the stack and the location changes to q'.
- $(q, a, \varphi, \gamma, q') \in \Delta_r$ is a pop-transition, where on reading a when the clock valuation satisfies φ, γ is popped from the stack, the location q changes to q' (if $\gamma = \perp$, it is read but not popped).
- $(q, a, \varphi, q') \in \Delta_i$ is an internal-transition, where the location, on reading a when the clock valuation satisfies φ, changes from q to q' without stack operations.

A stack is a nonempty finite sequence from the set $St = \{w\perp \mid w \in (\Gamma \setminus \{\perp\})^*\}$ starting with the top symbol on the left, and ending with the symbol $\perp$ on the right. The *empty stack* is the one that only contains the symbol $\perp$.

Definition 5. *A* configuration *of an ECVPA M is a pair (q, σ) where $q \in Q$, and $\sigma \in St$. For a timed word $\bar{w} = (a_0, t_0) \cdots (a_n, t_n)$, a run of M on $\bar{w}$ is a sequence of configurations $\rho = (q_0, \sigma_0) \cdots (q_{n+1}, \sigma_{n+1})$, where $q_i \in Q$, $\sigma_i \in St$, $q_0 \in Q_0, \sigma_0 = \perp$, and for every $0 \leq i \leq n$ one of the following condition holds:*

- **Push:** *If a_i is a call symbol, then for some $\gamma \in \Gamma$, $(q_i, a_i, \varphi_i, q_{i+1}, \gamma) \in \Delta_c$, $\nu_i^{\bar{w}} \models \varphi_i$, and $\sigma_{i+1} = \gamma.\sigma_i$.*
- **Pop:** *If a_i is a return symbol, then for some $\gamma \in \Gamma$, $(q_i, a_i, \varphi_i, \gamma, q_{i+1}) \in \Delta_r$, $\nu_i^{\bar{w}} \models \varphi_i$, and either $\gamma \in \Gamma$ and $\sigma_i = \gamma.\sigma_{i+1}$, or $\gamma = \sigma_i = \sigma_{i+1} = \perp$.*
- **Internal:** *If a_i is an internal symbol, then $(q_i, a_i, \varphi_i, q_{i+1}) \in \Delta_i$, $\nu_i^{\bar{w}} \models \varphi_i$, and $\sigma_{i+1} = \sigma_i$.*

A run ρ is an *accepting* run if it ends in a final location. A timed word $\bar{w}$ is an accepting word if there is an accepting run of M on $\bar{w}$. The language of an ECVPA M, denoted by $L(M)$, is the set of all accepting timed words $\bar{w}$ of M.

Remark 2. An (untimed) visibly pushdown automaton [4] can be seen as an event-clock visibly pushdown automaton that has no clock constraints on transitions. In the rest of this paper, we mention a VPA as an ECVPA without $\Phi(C_\Sigma)$ component in the transitions. Note also that a VPA is *deterministic* if $|Q_0| = 1$ and, for each configuration (q, σ) and $a \in \Sigma$, there are at most one transition from (q, σ) by a. VPAs are determinizable and closed under Boolean operations [4]. In particular, for a nondeterministic VPA with n states, one can construct an equivalent deterministic VPA with $O(2^{n^2})$ states and $O(2^{n^2}.|\Sigma_c|)$ stack symbols. The inclusion problem for VPAs is EXPTIME-complete [4].

We next present some examples of event-clock visibly pushdown automata.

Example 1. It is easy to see that an ECA [3] is an ECVPA that has only internal symbols, i.e, $\Sigma_c = \Sigma_r = \varnothing$. Thus, the class ECAs is a subclass of ECVPAs.

Example 2. Duration automata (DAs) were studied in [14] for modeling simple component-based real-time systems. A DA is a finite automaton in which each transition must occur in an associated time interval. A duration automaton can be viewed as an one-clock timed automata, where the clock is reset at each transition. The clock valuations of a DA are also explicitly determined by the input timed word. Therefore, the class of DAs can be seen as a special subclass of ECAs, and thus DA is a subclass of ECVPAs.

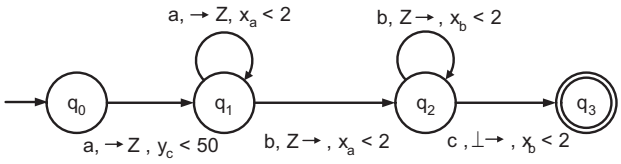

Fig. 2. Event-clock visibly pushdown automaton M

Example 3. Let a be a push. Let b, c be pops, and Z be a stack symbol. The ECVPA M of Figure 1 uses two event-recording clocks x_a and x_b, and an event-predicting clock y_c. The transitions of M are given as follows:

- **Push:** $(q_0, a, y_c < 50, q_1, Z)$, $(q_1, a, x_a < 2, q_1, Z)$.
- **Pop:** $(q_1, b, x_a < 2, Z, q_2)$, $(q_2, b, x_b < 2, Z, q_2)$, $(q_2, c, x_b < 2, \perp, q_3)$.

We describe locations of M as nodes of a graph. We adopt the following conventions to represent edges: for instance, a push-transition (q_i, a, ϕ, q_j, Z) is labeled as $a, \rightarrow Z, \phi$; a pop-transition (q_i, b, ϕ, Z, q_j) is labeled as $b, Z \rightarrow, \phi$.

The clock constraint $y_c < 50$ that is associated with the edge from q_0 to q_1 ensures that c occurs within 50 time units of the first a. The constraint $x_a < 2$

that is associated with the edge from q_1 to q_2 makes sure that the first b occurs within 2 time units of the last a.

The automaton M accepts the set of input timed words: $L(M) = \{(\bar{\alpha}, \bar{t}) \mid \bar{\alpha} = a^n b^n c, n \in N^+, t_{i+1} < t_i + 2. \forall (1 \leq i \leq 2n); t_{2n+1} - t_1 < 50\}$. This timed language, however, cannot be accepted by any timed automaton [2].

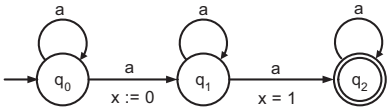

Fig. 3. One clock timed automaton A

The next example shows that ECVPA and timed automata are incomparable.

Example 4. Consider the timed language:

$$L = \{(a^n, \bar{t}) \mid n \geq 2, t_j - t_i = 1, \text{for some } 0 \leq i < j < n\}$$

The language L can be accepted by a nondeterministic one-clock timed automaton (1-TA) A in Figure 3. This language, however, cannot be accepted by any ECVPA.

Definition 6. *An ECVPA* $M = \langle Q, \Sigma, Q_0, \Gamma, \delta, F \rangle$*, is* deterministic *if* $|Q_0| \leq 1$ *and for every* $q \in Q$*, and for every clock valuation* ν*:*

- *if* $(q, a, \varphi_1, q_1) \in \Delta_i$ *and* $(q, a, \varphi_2, q_2) \in \Delta_i$*, then* $\nu \not\models \varphi_1 \wedge \varphi_2$*.*
- *if* $(q, b, \varphi_1, q_1, \gamma_1) \in \Delta_c$ *and* $(q, b, \varphi_2, q_2, \gamma_2) \in \Delta_c$*, then* $\nu \not\models \varphi_1 \wedge \varphi_2$*.*
- *if* $(q, c, \varphi_1, \gamma, q_1) \in \Delta_r$ *and* $(q, c, \varphi_2, \gamma, q_2) \in \Delta_r$*, then* $\nu \not\models \varphi_1 \wedge \varphi_2$*.*

The determinism condition ensures that at each step during a run, the choice of the next transition is uniquely determined by the current location of the ECVPA, the input word, the current stack content, and the current clock-valuation of the ECVPA along the input word. It is easy to check that every deterministic ECVPA has at most one run over any given timed input word.

3 Properties of Event-Clock Visibly Pushdown Automata

3.1 Untimed/Timed Translation between ECVPA and VPA

Similar to the case for event clock automata [12], we show in this section that an arbitrary ECVPA can be translated into an untimed VPA that interprets timing constraints symbolically and exhibits the same behaviors, and vice-versa.

In particular, for a given ECVPA M, let $B = \{r_0, r_1, ..., r_n\}$ be a finite set of constants appearing in the clock constraints of M. Without loss of generality, let us assume that $0 = r_0 < r_1 < \cdots < r_n$. We define $Intv = \{[\bot, \bot]\} \cup \{[r_i, r_i], (r_i, r_{i+1}) \mid 0 \leq i < n\} \cup \{[r_n, r_n], (r_n, \infty)\}$.

Definition 7. *An* interval alphabet *based on Σ is the set $\Pi = \Sigma \times Intv^{|C_\Sigma|}$. We have $|\Pi| = |\Sigma| \times |Intv|^{|C_\Sigma|} = |\Sigma| \times (2r_n + 1)^{|C_\Sigma|}$.*

Elements of an interval alphabet are of the form (a, g) with $a \in \Sigma$ and $g : C_\Sigma \to Intv$. The component g is called *guard*, and it is used to represent the timing constraint: $\bigwedge_{x \in C_\Sigma} x \in g(x)$.

Definition 8. *Define a function $tw : \Pi^* \to 2^{T\Sigma^*}$ where for each $\alpha = (a_0, g_0) \cdots (a_n, g_n) \in \Pi^*$, we have:*
$tw(\alpha) = \{\bar{w} \mid \bar{w} = (a_0, t_0) \cdots (a_n, t_n), \forall x \in C_\Sigma, \forall i (1 \leq i \leq n) : \nu_i^{\bar{w}}(x) \in g_i(x)\}$.

The untimed translation technique is formalized in the next definition:

Definition 9 (Untimed Transformation). *Let $M = \langle Q, \Sigma, Q_0, \Gamma, \Delta, F\rangle$ be an ECVPA. We define a VPA $ut(M) = \langle Q, \Pi, Q_0, \Gamma, \Delta', F\rangle$ in which for each transition e of M with the input symbol a and the clock constraint φ, there exists a natural number k such that e is translated to transitions of $ut(M)$ as follows:*

- *the interval input symbols are $(a, g_i) \in \Pi$, $i = 1 \cdots k$, and*
- *φ is equivalent to $\bigvee_{i=1..k}(\bigwedge_{x \in C_\Sigma} x \in g_i(x))$.*

Example 5. Consider a transition $e = (q, a, x_a < 10, q')$ of M. Suppose that M mentions constants 5 and 10. We have $Intv = \{[0, 0], (0, 5), [5, 5], (5, 10), [10, 10], (10, \infty), [\bot, \bot]\}$. The transition e will be translated to parallel transitions in $ut(M)$ as below:

$$(q, (a, [0, 0]), q'), \quad (q, (a, (0, 5)), q'), \quad (q, (a, [5, 5]), q'), \quad (q, (a, (5, 10)), q').$$

Note that this translation preserves determinism. Similar to the case for event-clock automata [12], the next lemma holds:

Lemma 1. *$tw(L(ut(M))) = L(M)$ for all ECVPAs M. Moreover, if M is a deterministic ECVPA, then $ut(M)$ is a deterministic VPA.*

The reverse of the translation is described in the next definition.

Definition 10 (Timed Transformation). *Let $N = \langle Q, \Pi, Q_0, \Gamma, \Delta', F\rangle$ be a VPA. We define an ECVPA $ec(N) = \langle Q, \Sigma, Q_0, \Gamma, \Delta, F\rangle$ such that each transition of N with the interval input symbol (a, g) is translated to a transition of $ec(N)$ whose input symbol is a and the clock constraint is $\varphi = \bigwedge_{x \in C_\Sigma} x \in g(x)$.*

Example 6 (Continued from Example 5). Now, suppose that we want to translate the VPA $ut(M)$ in Example 5 to an ECVPA. The transitions of $ut(M)$ are translated back to the following transitions:

$$(q, a, x_a = 0, q'), (q, a, 0 < x_a < 5, q'), (q, a, x_a = 5, q'), (q, a, 5 < x_a < 10, q').$$

Observe that this translation also preserves determinism.

For the timed translation, we get the following lemma:

Lemma 2. *$L(ec(N)) = tw(L(N))$ for all VPA N. Moreover, if N is deterministic VPA, then $ec(N)$ is a deterministic ECVPA.*

Proof. – For $\alpha = (a_0, g_0) \cdots (a_n, g_n) \in L(N)$, let $\rho = (q_0, \sigma_0) \cdots (q_{n+1}, \sigma_{n+1})$ be a run of N on α. If $\bar{w} = (a_0, t_0) \cdots (a_n, t_n) \in tw(\alpha)$, then ρ is also a run of $ec(N)$ on $\bar{w}$. Thus, $tw(L(N)) \subseteq L(ec(N))$.
– Conversely, let $\bar{w} = (a_0, t_0) \cdots (a_n, t_n) \in L(ec(N))$. There is an accepting run $\rho = (q_0, \sigma_0) \cdots (q_{n+1}, \sigma_{n+1})$ of $ec(N)$ on $\bar{w}$, $q_n \in F$. Based on the timed translation, there is an untimed word $\alpha = (a_0, g_0) \cdots (a_n, g_n) \in \Pi^*$ such that $\bar{w} \in tw(\alpha)$, and ρ is also a run of N on α. Thus, $L(ec(N)) \subseteq tw(L(N))$.

□

The next theorem immediately follows from Lemmas 1 and 2.

Theorem 1. *$L(ec(ut(M))) = L(M)$ for all ECVPAs M.*

3.2 Closure Properties and Inclusion Problem

From Theorem 1 and the decidability results of VPAs [4], the following theorems hold:

Lemma 3 (Determinization). *For any nondeterministic ECVPA M, there is a deterministic ECVPA $Det(M)$ such that $L(Det(M)) = L(M)$. Moreover, if M has n locations, we can construct $Det(M)$ with $O(2^{n^2})$ locations and $O(2^{n^2} \cdot |\Sigma_c| \cdot (2r)^{2|\Sigma|})$ stack symbols, where r is the largest constant appearing in the clock constraints of M. The set of clocks of $Det(M)$ coincides with that of M.*

Theorem 2 (Closure properties). *The class of ECVPAs is closed under union, intersection, and complementation.*

Theorem 3 (Language Inclusion). *The inclusion problem for ECVPAs is EXPTIME-complete.*

Proof. Consider two ECVPAs A and B such that each automaton has at most n locations, let m be the size of the input alphabet. Let c be the largest integer constant that appears in the clock constraints. To check whether $L(A) \subseteq L(B)$, we first untimed translate B to a VPA B_1, determinize B_1 to B_2, and then timed translate B_2 to an ECVPA B'. The automaton B' has 2^{n^2+n} locations. Let M be the product of A and B'. The ECVPA M has $n.2^{n^2+n}$ locations, and the integer constants that appear in the clock constraints of M are also bounded by c. Now, we can construct a VPA $ut(M)$, and check for its emptiness. Since checking emptiness of VPA is cubic time proportional to its size, it follows that emptiness of $ut(M)$ can be checked in EXPTIME. The proof of hardness is the same as the proof for VPA [4].

□

Remark 3. Büchi VPAs are closed under union, intersection, and complementation [4]. By using a similar technique, we also can translate Büchi ECVPA to Büchi VPA, and vice-versa. Therefore, the result of Theorem 3 can be extended to the case of Büchi ECVPA.

4 Related Classes of Timed Pushdown Automata

4.1 Timed Visibly Pushdown Automata

Let $X = \{x_1, ..., x_n\}$ be a finite set of *clocks*. Define the set $\Phi(X)$ of clock constraints over X by the grammar:

$$\varphi ::= \top \mid x \bowtie c \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2,$$

where $c \in Q^+$, $x \in X$, $\bowtie \in \{<, \leq, \geq, >\}$.

For the set of clocks X, a *clock valuation* is a function $\nu : X \to R^+$ which describes the values of each clock $x \in X$ at an instant. For the clock valuation ν and a clock constraint φ, we write $\nu \models \varphi$ to denote that ν satisfies the constraint φ. Given a set of clocks $\lambda \subseteq X$ and a clock valuation ν, let $\nu \downarrow \lambda$ be a clock valuation defined as follows:

$$(\nu \downarrow \lambda)(x) = \begin{cases} 0 \text{ when } x \in \lambda \\ \nu(x) \text{ otherwise} \end{cases} \tag{1}$$

Given a clock valuation ν and a time $t \in R^+$, define $(\nu + t)(x) = \nu(x) + t$.

Definition 11 (Timed Visibly Pushdown Automaton [8]). *A* timed visibly pushdown automaton *(TVPA) over* pushdown alphabet Σ *is a tuple* $M = (Q, \Sigma, Q_0, \Gamma, X, \delta, F)$*, where* Q *is a finite set of locations,* $Q_0 \subseteq Q$ *is a finite set of initial locations,* Γ *is a finite stack alphabet that contains* $\perp$*,* $F \subseteq Q$ *is a set of final locations,* X *is a finite set of clocks, and* $\delta = \delta_c \cup \delta_i \cup \delta_r$ *is the transition relation:*

- $\delta_c \subseteq Q \times \Sigma_c \times \Phi(X) \times Q \times (\Gamma \setminus \{\perp\}) \times 2^X$ *is the push-transition relation*
- $\delta_r \subseteq Q \times \Sigma_r \times \Gamma \times \Phi(X) \times Q \times 2^X$ *is the pop-transition relation*
- $\delta_i \subseteq Q \times \Sigma_i \times \Phi(X) \times Q \times 2^X$ *is the internal-transition relation.*

Let ν be a clock valuation. We briefly explain the intuition behind the transitions of a TVPA as follows:

- A *push-transition* $(q, a, \phi, q', \gamma, \lambda)$ is a move on the (call) input symbol a from q to q' where ν satisfies ϕ, the clock valuation is updated from ν to $\nu \downarrow \lambda$, and γ is pushed on the stack.
- A *pop-transition* $(q, a, \gamma, \phi, q', \lambda)$ is a move on the (return) input symbol a and stack symbol γ, from q to q' where ϕ is satisfied and ν is updated to $\nu \downarrow \lambda$, and γ is popped from the stack (if the top of stack is $\perp$, then it is read but not popped).
- An *internal-transition* $(q, a, \phi, q', \lambda) \in \delta$ at clock valuation ν is a move on the (internal) input symbol a from the location q to q' such that $\nu \models \phi$ and the resulting clock valuation $\nu' = \nu \downarrow \lambda$.

Remark 4. There are clock reset conditions λ in the transitions of TVPAs. Thus, unlike the case of ECVPAs, the clock valuations of TVPAs not only depend on input timed words but also on transitions of TVPAs.

Definition 12. *A* configuration *of a TVPA M is a triple (q, ν, σ), where $q \in Q$, $\sigma \in St$, and ν is a clock valuation. Given a timed word $\bar{w} = (a_0, t_0) \cdots (a_n, t_n)$, a* run *of M on $\bar{w}$ is a sequence of configurations $\rho = (q_0, \nu_0, \sigma_0) \xrightarrow{(a_0, t_0)} (q_1, \nu_1, \sigma_1) \cdots \xrightarrow{(a_n, t_n)} (q_{n+1}, \nu_{n+1}, \sigma_{n+1})$, where $q_0 \in Q_0$, $\sigma_0 = \perp$, and for every $1 \leq i \leq n$, one of the following conditions holds:*

- **Push:** $(q_i, a_i, \phi_i, q_{i+1}, \gamma, \lambda) \in \delta$, $\nu_i \models \phi_i$, $\nu_{i+1} = (\nu_i \downarrow \lambda) + (t_{i+1} - t_i)$, *and* $\sigma_{i+1} = \gamma\sigma_i$
- **Pop:** $(q_i, a_i, \gamma, \phi_i, q_{i+1}, \lambda) \in \delta$, $\nu_i \models \phi_i$, $\nu_{i+1} = (\nu_i \downarrow \lambda) + (t_{i+1} - t_i)$, *and* $\gamma\sigma_{i+1} = \sigma_i$ *or* $\gamma = \sigma_{i+1} = \sigma_i = \perp$,
- **Internal:** $(q_i, a_i, \phi_i, q_{i+1}, \lambda) \in \delta$, $\nu_i \models \phi_i$, $\nu_{i+1} = (\nu_i \downarrow \lambda) + (t_{i+1} - t_i)$, *and* $\sigma_{i+1} = \sigma_i$

A run ρ is an *accepting* run if it ends in a final state. A timed word $\bar{w}$ is *accepting* if there is an accepting run of M on $\bar{w}$. The language $L(M)$ is the set of timed words accepted by M. It is easy to see that TVPAs is a subclass of timed pushdown automata (TPDA) [5], and a superclass of timed automata [2]. Indeed, a TVPA is a timed automaton if $\Sigma_c = \Sigma_r = \varnothing$. Unlike ECVPA, TVPA are not determinizable and not closed under complementation. Moreover, the next theorem was proved by Emmi and Majumdar [8].

Theorem 4 (Language inclusion [8]). *The inclusion problem $L(A) \subseteq L(B)$, where A is a TVPA and B is a TVPA with at least* one clock, *is undecidable.*

4.2 Translation from ECVPA to TVPA

As shown in [3], every ECA can be translated into a timed automaton that accepts the same timed language. There the basic idea of the translation can be described as follows:

Definition 13 (Translating event-clocks to original clocks [3]). *An event-recording clock x_a can be seen as an original clock that is reset on a transition e if the input symbol of e is a. For event-predicting clocks, consider a given ECA A and the set of all* atomic event-predicting clock constraints *(denoted by Φ_A) of the form $y_a = \perp$ or $y_a \sim c$, where $\sim \in \{\leq, <, >, \geq\}$. Define a nondeterministic timed automaton B as follows:*

- *The states of the target timed automaton B are the pairs (q, Ψ) with $q \in Q_A$ and $\Psi \subseteq \Phi_A$.*
- *The state (q, Ψ) is an initial state of B iff q is the initial state of A and Ψ does not contain a constraint of the form $y_a \sim c$.*
- *The state (q, Ψ) is a final state of B iff $q \in F_A$ and $\Psi = \{y_a = \perp\}$.*
- *For each $\psi \in \Phi_A$, B has a clock z_ψ. A prediction $y_a \sim c$, along a transition in A, on the time difference to the next occurrence of a is replaced in B by a constraint on the clock $z_{(y_a \sim c)}$: the clock $z_{(y_a \sim c)}$ is reset when the prediction is performed, and its value is checked by the constraint $z_{(y_a \sim c)} \sim c$ when the next a occurs.*

Similarly, by using the above translation of the event clocks to the original clocks, the next lemma holds:

Lemma 4. *Every ECVPA can be translated into a TVPA that accepts the same timed language.*

We now arrive at the main theorem of this paper:

Theorem 5 (Language Inclusion). *The inclusion problem $L(A) \subseteq L(B)$, where A is a TVPA and B is an ECVPA, is decidable.*

Proof. $L(A) \subseteq L(B) \iff L(A) \cap L(B)^c = \varnothing$. We first determinize B and then compute its complement B^c. Second, translate the ECVPA B^c into a TVPA B'. Note that $L(B') = L(B)^c$. Third, take the intersection of A and B', and check for emptiness of $L(A \cap B')$. Similar to Theorem 3, the complexity of this inclusion checking is EXPTIME-complete.

□

5 Related Works

Ouaknine and Worrell [11] proved that when B has at most one clock, the inclusion problem $L(A) \subseteq L(B)$ can be encoded as the reachability problem for well-structured transition systems [9]. Thus, the inclusion problem for timed automata becomes decidable when B has at most one clock. However, over infinite timed words, one clock is enough to make the inclusion problem undecidable [1], since this problem can be reduced from the space-bounded recurrent-state problem for alternating channel machines.

The closest related works is the paper of Emmi and Majumdar [8]. There they extended the proof technique of [11], and showed that the inclusion problem $L(A) \subseteq L(B)$ is decidable if A is a TPDA, and B is a TA with at most one clock. However, for TVPAs A and B, the inclusion problem is undecidable even B has exactly one clock [8]. In this paper, we have shown that when A is a TVPA and B is an ECVPA, the inclusion problem $L(A) \subseteq L(B)$ become decidable.

A decidable subclass of real-time logic *so-called* EventClockTL, which corresponds to event-clock automata, was presented in [10]. Furthermore, DSouza [12] showed that the class of event-clock automata admit a logical characterization via a monadic second order logic interpreted over timed words. The proof technique is based on the untimed translation, as in Definition 9, that transform an event-clock automaton to a finite automaton.

6 Conclusion

In this paper, we have introduced the class of ECVPAs by combining the ideas of ECAs [3] and VPAs [4]. We showed that the class of ECVPAs enjoys good closure properties and decidability results. We also showed that the inclusion problem

$L(A) \subseteq L(B)$, where A is a TVPA and B is an ECVPA, is decidable. This provide an algorithm for checking if a TVPA meets a specification that is given as an ECVPA. We hope that the class of ECVPA will be useful for verification of recursive real-time programs. For future work, it would be interesting to study a logical charaterization of the class ECVPAs.

Acknowledgements

We would like to thank anonymous referees for their valuable comments and suggestions in improving the paper. This research is supported by the 21st Century COE program "Verifiable and Evolvable e-Society" of JAIST, funded by the Japanese Ministry of Education, Culture, Sports, Science and Technology.

References

1. P. A. Abdulla, J. Deneux, J. Ouaknine, and J. Worrell. Decidability and complexity results for timed automata via channel machines. In *ICALP'05*, LNCS 3580, pp. 1089-1101, 2005.
2. A. Alur and D. Dill. A theory of timed automata. *Theor. Comp. Sci*, 126: pp. 183-235, 1994.
3. R. Alur, L. Fix, and T. A. Henzinger. Event-Clock Automata: A Determinizable Class of Timed Automata. *Theor. Comp. Sci*, 211: pp.253-273, 1999. A preliminary version appeared in *CAV'94*, LNCS 818, pp. 1-13, 1994.
4. R. Alur and P. Madhusudan. Visibly pushdown languages. In *STOC'04*, pp. 202-211, New York, June 13-15, 2004. ACM Press.
5. A. Bouajjani, R. Echahed, and R. Robbana. On the automatic verification of systems with continuous variables and unbounded discrete data structures. In *Hybrid Systems II*, LNCS 999, pp. 64-85, 1995.
6. P. Bouyer and F. Laroussinie. Model Checking Timed Automata. In *Modeling and Verification of Real-Time Systems*, pages 111-140. ISTE Ltd. - John Wiley & Sons, Ltd., 2008.
7. P. Bouyer, Kim G. Larsen and N. Markey. Model Checking One-clock Priced Timed Automata. Logical Methods in Computer Science 4(2:9), 2008.
8. M. Emmi and R. Majumdar. Decision Problems for the Verification of Real-time Software. In *HSCC'06*, LNCS 3927, pp. 200-211, 2006.
9. A. Finkel and Ph. Schnoebelen. Well-Structured Transition Systems Everywhere!. *Theor. Comp. Sci*, 256(1-2), pp. 63-92, 2001.
10. T. A. Henzinger, J. Raskin, and P. Schobbens. The regular real-time languages. In *ICALP'98*, LNCS 1443, pp. 580-598, 1998.
11. J. Ouaknine and J. Worrell. On the Language Inclusion Problem for Timed Automata: Closing a Decidability Gap. In *LICS'04*, pp. 54-63, 2004. IEEE Press.
12. D. D'Souza. A Logical Characterisation of Event Clock Automata. International Journal of Foundation for Computer Science, Vol. 14(4), pp. 625-640, 2003.
13. D. D'Souza, N. Tabareau. On Timed Automata with Input-Determined Guards. In *FORMATS/FTRTFT'04*, LNCS 3253, pp. 68-83, 2004.
14. N. V. Tang, D. V. Hung, and M. Ogawa. Modeling Urgency in Component-Based Real-time Systems. In *ASIAN'06*, LNCS 4435, pp. 248-255, 2006.