Ramsey-Based Inclusion Checking for Dense-Stack Visibly Pushdown Automata

Devendra Bhave and Ramchandra Phawade
Indian Institute of Technology Bombay, India

Abstract. Visibly pushdown automata are popular as they are closed under Boolean operations and determinization. There exist multiple notions of timed pushdown systems like recursive timed automata, dense-time pushdown automata. We explore a generalization of visibly pushdown automata over infinite words with parity acceptance condition—in which stack elements have real valued time stamps—named dense-stack visibly pushdown automata. We prove its closure under union and intersection and show that inclusion checking for this class is decidable using Ramsey-based techniques.

1 Introduction

Automata theoretic approach to formal verification rests on solving the language inclusion [14] problem “\( L(A) \subseteq L(B) \)”. Here \( A \) is an automata theoretic model for the system \( S \) and an automaton \( B \) represents a property \( P \) to be verified. The set of words in the language of \( A \), denoted as \( L(A) \), represents all the possible behaviors of \( S \), and the set of behaviors of \( B \) is denoted by \( L(B) \). Thus property verification problem reduces to solving decision problem whether \( L(A) \subseteq L(B) \). In practical system verification applications, it is desirable that this problem is decidable for the models of the given class of systems and it can be solved sufficiently faster. For the class of automata that are closed under complementation, the language inclusion problem can be posed as an emptiness question \( L(A) \cap L(C) \neq \emptyset \) where an automaton \( C \) may be obtained by complementing \( B \), which in turn may be solved by determinizing \( B \). So this inclusion based technique depend on determinization and complementation where one needs to explicitly construct automata. For Büchi automata this approach produces a large automata [6]. Although the emptiness problem for Büchi automata is \( \text{NLOGSPACE} \)-complete [15], its complementation has exponential lower bound [11]. And for a richer model of visibly push down automata VPA where input alphabet is partitioned into call, return and local alphabet, the respective lower bound is \( \text{EXPTIME} \) [4].

An alternative technique for inclusion checking is based on Ramsey’s Theorem [12], which solves universality checking first and employs it to do the inclusion checking [3]. These algorithms have exponential running times in the worst case but perform better in the practice than the ones based on determinization and then complementation. This approach has been generalized to VPA\( \text{s with parity acceptance condition} \) [9].
Fig. 1. Dense-stack Visibly Pushdown Automata

From modeling perspective, many rich models for real-time recursive programs have been proposed. Recursive timed automata (RTA)[13] and dense-time pushdown automata (dtPDA)[2] are generalization of timed automata that accept certain real-time extensions of context-free languages. But, language inclusion problem for both of these models is undecidable. We recently proposed dense-timed visibly pushdown automata (dtVPA) [5] which is subclass of dtPDA, but with decidable inclusion problem for finite word languages. For the infinite timed words, the language inclusion problem remains open. However for untimed infinite words, the language inclusion for VPAs is shown decidable[9].

In this paper we study Ramsey-based technique for inclusion checking for a more general VPA model: a class of VPAs equipped with stack symbols having real valued time stamps (called age). This model is inspired from [1], and [5] but this is slightly simpler model because ages of stack symbols are not compared with arbitrary clocks; we allow to check only the age of stack symbol being popped. Another difference is that this model is defined for infinite words unlike before [5], to achieve that we use parity acceptance condition. This model is defined in Section 2.

Example 1. Consider the timed language of the form \((a^n bc^n d)^\omega\) where the first \(c\) comes precisely 1 time-unit after last \(a\) and the first \(a\) and the last \(c\) are 2 time-units apart, and every other matching \(a\) and \(c\) are within (1, 2) time-unit apart. Once the stack is emptied after reading input \(d\), this cycle can be started again to get infinite words of the form \(\{(a^n bc^n d, \langle t_1, \ldots, t_n, t, t'_1, \ldots, t'_n, t'\rangle)^\omega | \ t'_n - t_n = 1, t'_i - t_i = 2, t'_1 - t_1 \in (1, 2) \text{ for all } i \leq n\}\). Given a partition of input alphabet \(\Sigma = \{a, b, c, d\}\) into call symbols \(\Sigma_c = \{a\}\), local symbols \(\Sigma_l = \{b, d\}\), and return symbols \(\Sigma_r = \{c\}\) and stack alphabet \(\Gamma = \{\alpha\}\) this language is accepted by the dsVPA shown in Figure 1. The priorities of the states are given as \(\Omega(l_0) = 1, \Omega(l_1) = 1, \Omega(l_2) = 1,\) and \(\Omega(l_3) = 2.\) Since the highest priority which occurs infinitely often is even (2 in this case) in a run of all words of this language, it is accepted.

In Section 2, we introduce dsVPA model and prove certain closure properties. Ramsey based technique for universality checking of dsVPA is in Section 3 and in Section 4 we extend it to obtain inclusion checking. Additional results and proofs are included in the appendix.
2 Preliminaries

Let $\Sigma$ be a finite alphabet. Let $\mathbb{N}, \mathbb{R}, \mathbb{R}_{>0}$ and $\mathbb{R}_{\geq 0}$ denote the set of natural numbers, real numbers, positive real numbers excluding zero, and non-negative real numbers respectively. A finite word $w$ is an element of $\Sigma^*$, while a finite timed word is a delay annotated sequence $((a_0, \delta_0), (a_1, \delta_1), \ldots, (a_n, \delta_n)) \in (\Sigma \times \mathbb{R}_{\geq 0})^*$ such that all time delays $\delta_i \geq 0$ for $0 \leq i \leq n$. An infinite timed word is represented with a sequence $((a_0, \delta_0), (a_1, \delta_1), \ldots, (a_n, \delta_n)) \in (\Sigma \times \mathbb{R}_{\geq 0})^\omega$ with each $\delta_i \geq 0$ and $\delta_i$ is the delay between the symbols $a_i$ and $a_{i+1}$. We denote the set of infinite timed words over $\Sigma$ by $\mathcal{T}\Sigma^\omega$.

We use following ordering relations on priority values. The priority ordering is denoted by $\preceq$ and defined over the set $S_{\preceq} = \mathbb{N} \cup \uparrow$, such that $0 \preceq 1 \preceq 2 \preceq \cdots \preceq \uparrow$ is the ordering among its elements. $\uparrow$ is the maximum element and $\perp_{\preceq} = 0$ is the minimum element of $S_{\preceq}$. For any nonempty subset $S \subseteq S_{\preceq}$, $\bigcup S$ and $\bigcap S$ denote the maximal and minimal element over $S$ respectively. Also, we use infix notation $a \sqcup b$ to denote $\bigcup\{a, b\}$ and $a \sqcap b$ to denote $\bigcap\{a, b\}$. The reward ordering is denoted by $\preceq_\gamma$ and defined over the set $S_{\preceq}$, such that $\uparrow_\gamma \prec \cdots \prec 5_\gamma \prec 3_\gamma \prec 1_\gamma \prec 0_\gamma \prec 2_\gamma \prec \cdots \prec T_\gamma$ is the ordering among its elements. Intuitively the reward ordering captures the degree of betterness for the acceptance under parity condition. $T_\gamma$ is the maximum element and $\uparrow_\gamma$ is the minimum element of $S_{\preceq}$. For any nonempty subset $S \subseteq S_{\preceq}$, $\forall S$ and $\exists S$ denote the maximal and minimal elements over $S$ respectively. Also, we use infix notation $a \triangleright b$ to denote $\forall\{a, b\}$ and $a \blacktriangleright b$ to denote $\exists\{a, b\}$.

2.1 Dense stack VPA(dsVPA)

We introduce the dense stack visibly pushdown automata (dsVPA) as an VPA equipped with a timed stack, defined over a visibly pushdown alphabet $\Sigma = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$.

Let $I = \{(a, b), [a, b], (a, b), [a, b], (a, \infty), [a, \infty] \mid a, b \in \mathbb{N}\}$ be the set of intervals. The intersection of two intervals is another interval such that $I_1 \cap I_2 = \{t \mid t \in I_1 \text{ and } t \in I_2\}$.

Definition 2. A dense-stack visibly pushdown automata over $\Sigma = \{\Sigma_c, \Sigma_r, \Sigma_l\}$ is a tuple $M=(L, \Sigma, \Gamma, L^0, \Delta = \Delta_c \cup \Delta_r \cup \Delta_l, \Omega)$, where $L$ is a finite set of locations including a set $L^0 \subseteq L$ of initial locations, $\Gamma$ is a finite stack alphabet with special end-of-stack symbol $\bot$, $\Delta_c = (L \times \Sigma_c \times L \times \Gamma \setminus \bot)$ is the set of call transitions, $\Delta_r = (L \times \Sigma_r \times \Gamma \times L)$ is set of return transitions, $\Delta_l = (L \times \Sigma_l \times L)$ is set of local or internal transitions, $\Omega : L \mapsto \mathbb{N}$ is the priority function.

Let $w = (a_0, \delta_0), \ldots, (a_n, \delta_n)$ be a finite timed word. A configuration of the dsVPA is a tuple $(\ell, (\gamma\sigma, \text{age}(\gamma\sigma)))$ where $\ell$ is the current location of the dsVPA, $\gamma\sigma \in \Gamma^*(\bot)$ is the content of the stack with $\gamma$ being the topmost symbol and $\sigma$ is the string representing the stack content below $\gamma$, while $\text{age}(\gamma\sigma)$ is a sequence of real numbers encoding the ages of all the stack symbols (the time elapsed since each of them was pushed on to the stack). We follow the
assumption that \( \text{age}(\bot) = (\bot) \) (undefined). If for some string \( \sigma \in \Gamma^* \) we have that \( \text{age}(\sigma) = \langle \ell_1, \ell_2, \ldots, \ell_n \rangle \) and for \( \tau \in \mathbb{R}_{\geq 0} \) we write \( \text{age}(\sigma) + \tau \) for the sequence \( \langle \ell_1 + \tau, \ell_2 + \tau, \ldots, \ell_n + \tau \rangle \). For a sequence \( \sigma = \langle \gamma_1, \ldots, \gamma_n \rangle \) and a member \( \gamma \) we write \( \gamma :: \sigma \) for \( \langle \gamma, \gamma_1, \ldots, \gamma_n \rangle \).

The run of a dsVPA on a timed word \( w = (a_0, \delta_0), \ldots, (a_n, \delta_n) \) is a sequence of configurations given as follows: \( (\ell_0, (\langle \bot \rangle, (\bot))), (\ell_1, (\sigma_1, \text{age}(\sigma_1))), \ldots, (\ell_{n+1}, (\sigma_{n+1}, \text{age}(\sigma_{n+1}))) \) where \( \ell_i \in L, \sigma_i \in \Gamma^* \langle \bot \rangle, \ell_0 \in L^0 \), and for each \( i, \ 0 \leq i \leq n, \) we have:
- If \( a_i \in \Sigma_c \), then there is a transition \( (\ell_i, a_i, \ell_{i+1}, \gamma) \in \Delta \). The symbol \( \gamma \in \Gamma \setminus \{ \bot \} \) is then pushed onto the stack, and its age is initialized to zero, obtaining \( (\sigma_{i+1}, \text{age}(\sigma_{i+1})) = (\gamma :: \sigma_i, 0 :: (\text{age}(\sigma_i) + \delta_i)). \) Note that all symbols in the stack excluding the topmost age by \( \delta_i. \)
- If \( a_i \in \Sigma_r \), then there is a transition \( (\ell_i, a_i, I, \gamma, \ell_{i+1}) \in \Delta \). The configuration \( (\ell_i, (\sigma_i, \text{age}(\sigma_i))) \) evolves to \( (\ell_{i+1}, (\sigma_{i+1}, \text{age}(\sigma_{i+1}))) \) where \( \sigma_i = \gamma :: k \in \Gamma \Gamma^* \{ \bot \} \) and \( \text{age}(\gamma) + \delta_i \in I \). We call this interval \( I \) as the guard interval. Then we obtain \( \sigma_{i+1} = \kappa \), with \( \text{age}(\sigma_{i+1}) = \text{age}(\kappa) + \delta_i \). However, if \( \gamma = (\bot) \), the symbol is not popped, and the attached interval \( I \) is irrelevant.
- If \( a_i \in \Sigma_t \), then there is a transition \( (\ell_i, a_i, \ell_{i+1}) \in \Delta \). In this case stack remains unchanged i.e. \( \sigma_i = \sigma_{i+1} \), and \( \text{age}(\sigma_{i+1}) = \text{age}(\sigma_i) + \delta_i \). All symbols in the stack age by \( \delta_i \).

The run of a dsVPA on infinite word \( w = (a_0, \delta_0), (a_1, \delta_1), \ldots \) is a sequence of configurations \( \rho = (\ell_0, (\langle \bot \rangle, (\bot))), (\ell_1, (\sigma_1, \text{age}(\sigma_1))), \ldots \) for which transitions rules are the same as that of finite timed words. Let \( \text{inf}(\rho) \) be the set of locations that occur infinitely often in \( \rho \). The run \( \rho \) is said to be accepting if \( \max\{ \text{delay}(\ell) | \ell \in \text{inf}(\rho) \} \) is even. A timed word \( w \) is accepted if there is an accepting run of \( M \) on \( w \). The language \( L(M) \) is the set of all timed words \( w \) accepted by \( M \).

Note that we use clock contraints only in the return transitions, and those only check whether the age of the popped symbol falls in the guard interval or not. Let \( G \) be the set of all guard intervals that appear in the transitions of \( M \). We specify the bounds of an interval using either natural numbers or special symbol \( \infty \). Let \( k \) be the maximal natural number (and not special symbol \( \infty \)) occurring in the bounds of intervals in \( G \). For any two age values greater than \( k \), guard condition evaluates identically. We use an interval \((k, \infty)\) to mark delays corresponding to such age values. We define proper interval set, \( \mathcal{I}_{pr} = \{(n, n] \mid n \in \mathbb{N} \text{ and } n \leq k\} \cup \{(i, i + 1) \mid i \in \mathbb{N} \text{ and } i \leq k\} \cup \{(k, \infty)\} \) which partitions delay values into finitely many intervals. We also define \( k \)-interval set, \( \mathcal{I}_k = \{(n, n] \mid n \in \mathbb{N} \text{ and } n \leq k\} \cup \{(i, j) \mid i, j \in \mathbb{N} \text{ and } i < j \leq k\} \cup \{(i, \infty) \mid i \in \mathbb{N} \text{ and } i \leq k\} \).

We define function \( \text{part}(I) \) which maps any interval \( I \) to some element in \( \mathcal{I}_k \).

**Definition 3.** We define partition of an interval as follows where \( a < b \),

\[
\begin{align*}
\text{part}((a, b]) &= (\min(a, k), b) \quad \text{if } b \leq k \\
\text{part}((a, b]) &= (\min(a, k), \infty) \quad \text{if } b > k \\
\text{part}([n, n]) &= [n, n] \quad \text{if } n \leq k \\
\text{part}([n, n]) &= (k, \infty) \quad \text{if } n > k
\end{align*}
\]
We define addition of intervals over $I_k$ as follows:

\[
[m, m] \oplus [n, n] = \text{part}([m + n, m + n]) \\
(a, b) \oplus [c, d] = \text{part}((a + c, b + d)) \\
(a, b) \oplus [n, n] = \text{part}((a + n, b + n))
\]

Using the function $\text{part}(I)$ and interval additions we get the following remark:

**Remark 4.** Following intervals are idempotents (i.e. $I \oplus I = I$) with respect to interval addition: $(k, \infty)$, $(0, \infty)$, $[0, 0]$.

The class of dsVPA is closed under union and intersection, is given by Theorem 5 whose proof is given in Appendix.

**Theorem 5 (Closure Property Theorem).** The class of dsVPA is closed under union and intersection operations.

TheLemma 6 allows us to contract an infinite run to get a shorter one in which highest priorities which occur infinitely often are preserved.

**Lemma 6 ([9]).** Let $\rho \in C^\omega$, where $C \subseteq \mathbb{N}$ in some finite set of priorities. For any strictly increasing sequence of natural numbers $i_0 < i_1 < i_2 < \cdots$, create another sequence $\rho' = \rho'_{i_0}, \rho'_{i_1}, \cdots$ such that $\rho'_{i_j} := \bigcup \{\rho_{i_j}, \rho_{i_{j+1}}, \cdots, \rho_{i_{j+1}-1}\}$ for $j \in \mathbb{N}$. Then $\bigcap \inf(\rho) = \inf(\rho')$ holds.

### 3 Universality Checking

In this section we describe an algorithm for universality checking for dsVPA. Universality checking algorithm critically depends on following classical theorem.

**Theorem 7 (Ramsey[12]).** Let $F$ be a finite set and $\chi : \mathbb{N}^2 \mapsto F$ be the $F$-coloring of $\mathbb{N}^2$. Then there is some $M \subseteq \mathbb{N}$ and element $k \in F$ such that $|M| = \infty$ and $\chi(i, j) = k$ for all $i, j \in M$ with $i < j$.

For this discussion, let $A = (L, \Sigma, \Gamma, L^0, F, \Delta = \Delta_c \cup \Delta_r \cup \Delta_l, \Omega)$ be the dsVPA which will be referred to in this section. Let $w \in T \Sigma^*$ be a finite timed word of the following form $w = (a_1, \delta_1), (a_2, \delta_2), \cdots, (a_n, \delta_n)$. Here $a_i$ is a symbol that occurs at position $i$ and $\delta_i$ is the timed delay between $a_i$ and $a_{i+1}$. The position $i$ is said to be a call position in $w$ where $a_i$ is a call symbol. Return and local positions are defined in the similar manner. Owing to the visibility criteria of the dsVPA, for every call position its corresponding matching position is known for a given input and vice versa. This permits us to define a notion of well-matching on the words. Definition 8 also applies to infinite timed words.

**Definition 8.** A word $w$ is said to be well-matched if

- every call position has a corresponding matching return position in $w$,
- every return position has a corresponding matching call position in $w$,
- every call position precedes its matched return position,
- all call-returns are well nested i.e. there are no positions in $w$ of the form $c_1 < c_2 < r_1 < r_2$ where $c_1, r_1$ are matching and $c_2, r_2$ are matching.
3.1 For well-matched words

We now present an algorithm which determines whether \( \mathcal{A} \) accepts all well-matched infinite timed words. Let \( \text{WM}(T_{\Sigma^\omega}) \) be the set of all well-matched infinite timed words. Our algorithm checks whether \( L(\mathcal{A}) = \text{WM}(T_{\Sigma^\omega}) \). We now define the functions called transition profiles (TP) similar to those defined by Friedmann et al \[10, 9\]. These transition profiles describe the behavior of \( \mathcal{A} \) on finite timed words. Let \( P = \{ \Omega(\ell) \mid \ell \in L \} \cup \{ \dagger \} \) be the codomain of TP values. Transition profiles can be atomic or composite and have one of the following type – call, return or local. We now formally define these notions.

**Definition 9.** For a given input symbol with delay \((a, \delta)\), we define three types of atomic transition profiles.

- A Call transition profile (CTP) is a function of type \( L \times I \times L \mapsto P \) such that
  \[
  f^1_a(\ell, \gamma, \ell') := \begin{cases} 
  \Omega(\ell') & \text{if there is a call transition } (\ell, a, \ell', \gamma) \in \Delta_c \\
  \dagger & \text{otherwise.}
  \end{cases}
  \]

- A Return transition profile (RTP) is a function of type \( L \times I \times L \mapsto P \) such that
  \[
  f^1_a(\ell, J, \ell') := \begin{cases} 
  \Omega(\ell') & \text{if there is a return transition } (\ell, a, J, \gamma, \ell') \in \Delta_r \\
  \dagger & \text{otherwise.}
  \end{cases}
  \]

- A Local transition profile (LTP) is a function of type \( L \times L \mapsto P \) such that
  \[
  f^1_a(\ell, \ell') := \begin{cases} 
  \Omega(\ell') & \text{if there is a local transition } (\ell, a, \ell') \in \Delta_l \\
  \dagger & \text{otherwise.}
  \end{cases}
  \]

Consider a unit length word \( w = (a_1, \delta_1) \) such that \( a_1 \) is a call symbol. When \( w \) is presented as input, \( \mathcal{A} \) takes some call transition which results in pushing \( \gamma \) on the stack and changes current location to \( \ell_1 \). Then delay \( \delta_1 \) occurs at \( \ell_1 \) and age of \( \gamma \) increases by \( \delta_1 \). This behavior of \( \mathcal{A} \) is captured by TP \( f_w = f^{L_1}(\cdot, \cdot, \cdot) \) where \( \delta_1 \in I_1 \). Likewise, transition profiles for \( w \) can be constructed if \( a_1 \) is a return or a local symbol. For convenience we denote the set of all atomic CTPs by \( T_{\text{Call}} \), the set of all atomic RTPs by \( T_{\text{Return}} \) and the set of all atomic LTPs by \( T_{\text{Local}} \). Now we describe the method of stitching together unit length word transition profiles to get transition profiles for longer words. We define the composition operation on transition profiles which intuitively does that stitching job. As in \[9\], we permit at most one unmatched call or return while composing.

**Definition 10.** When \( f^{L_1} \) and \( g^{L_2} \) are LTPs, their composition is also LTP and is defined as

\[
(f \circ g)^{L_1 \oplus L_2}(\ell, \ell') = \bigvee \left\{ f^{L_1}(\ell, \ell'') \cup g^{L_2}(\ell'', \ell') \mid \ell'' \in L \right\}
\]

For a CTP \( f^{L_1} \) and an LTP \( g^{L_2} \), their composition is a CTP and is defined as

\[
(f \circ g)^{L_1 \oplus L_2}(\ell, \gamma, \ell') = \bigvee \left\{ f^{L_1}(\ell, \gamma, \ell'') \cup g^{L_2}(\ell'', \ell') \mid \ell'' \in L \text{ and } \gamma \in I \right\}
\]
For a CTP $f^1$, and an RTP $g^2$, their composition is an LTP and is defined as

$$(f \circ g)^{(I_1 \cap J)} = (f_1 \circ g^2)(\ell, \ell') = \bigcup I \{ f^1(\ell, \gamma, \ell'') \cup g^2(\ell'', \gamma, J, \ell') \mid \ell'' \in L \text{ and } \gamma \in \Gamma \text{ and } I_1 \cap J \neq \emptyset \}$$

where, the interval $J \in I_{pr}$ is the stack guard interval, associated with RTP $g^2$.

Let $w = (a_1, \delta_1), (a_2, \delta_2), \ldots, (a_n, \delta_n)$ be a finite timed word. We define the duration of $w$ as $[w] = \sum_{i=1}^{n} \delta_i$. Let $\rho = (\ell_0, \langle \cdot \rangle), (\ell_1, \langle \sigma_1, \text{age}(\sigma_1) \rangle), \ldots, (\ell_m, \langle \sigma_m, \text{age}(\sigma_m) \rangle)$ be a run of a finite timed word $w$ on $A$. We use $\ell_0 \xrightarrow{w} \ell_m$ to denote such $\rho$, and we use $\ell_0 \xrightarrow{w} \ell_m$ to denote that $p$ is the maximal priority occurring along this run. Let $\text{WM}(T \Sigma^*)$ be the set of all finite well-nested timed words. Following properties of the composition of TPs allow us to forget ordering of composition for LTPs.

**Proposition 11.** Let $f, g$ and $h$ be LTPs. If $(f \circ g) \circ h$ and $f \circ (g \circ h)$ are both defined, then $(f \circ g) \circ h = f \circ (g \circ h)$.

**Proposition 12.** Let $f$ be a CTP and $g, h$ be LTPs. If $(f \circ g) \circ h$ and $f \circ (g \circ h)$ are both defined, then $(f \circ g) \circ h = f \circ (g \circ h)$.

**Proposition 13.** Let $f$ be a composite LTP. Then $f$ can be decomposed in at least one of the following ways: (i) $f = f_1 \circ f_2$, where $f_1$ and $f_2$ are LTPs, (ii) $f = f_c \circ f_r$ where $f_c$ is atomic CTP and $f_r$ is atomic RTP, or (iii) $f = f_c \circ g \circ f_r$ where $f_c$ is atomic CTP, $g$ is LTP and $f_r$ is atomic RTP.

We associate of a finite timed word with a TP.

**Lemma 14.** Let $\ell, \ell' \in L$ be the locations of $A$ and $p \neq \dagger$ be parity. If for a LTP $f^1$ we have $f^1(\ell, \ell') = p$ then there is a well-matched finite timed word $w$ such that $\ell \xrightarrow{w} p \ell'$ and $[w] \in I$.

**Lemma 15.** Let $w$ be a finite well-nested timed word. Let $\ell, \ell'$ be the locations of $A$ with priorities $p$ and $p'$ respectively. If $\ell \xrightarrow{w} p \ell'$ then there exists an LTP $f^1(\ell, \ell') = p'$ with $p \leq p'$ and $[w] \in I$.

We extend the composition operation for the sets of TPs as $F \circ G = \{ f \circ g \mid f \in F \text{ and } g \in G \text{ such that } f \circ g \text{ is defined} \}$.

**Definition 16.** We define a set $X$ using the following fixed point equation. $X = T_{\text{local}} \cup T_{\text{call}} \circ T_{\text{return}} \cup T_{\text{call}} \circ X \circ T_{\text{return}} \cup X \circ X$.

As number of local TPs is finite, powerset lattice of elements of set $X$ is finite. Composition $\circ$ and set union are monotonic functions. Thus, least fixed point exists for the above equation and is given as $\exists$. We now establish connection between the set $\exists$ and the set of all well-matched finite timed words.

**Lemma 17.** A transition profile $f \in \exists$ iff there is a well-matched word $w$ in $\text{WM}(T \Sigma^*)$ such that $f = f_w$. 

For LTP $f$ and location $\ell$, we define $f(\ell) = \{\ell' \mid f(\ell, \ell') \neq \dagger\}$. Let $L' \subseteq L$, we define $f(L') = \bigcup_{\ell' \in L'} \{\ell' \mid f(\ell, \ell') \neq \dagger\}$.

**Definition 18. Idempotency:** A TP $f$ is idempotent if $f \circ f = f$.

**Badness:** Given a subset $L' \subseteq L$ of locations and an idempotent TP $f$, the TP $f$ is bad for the set $L'$ if $f(\ell, \ell)$ is either $\dagger$ or odd for all $\ell \in f(L')$.

**Remark 19.** An idempotent TP $g'$ is an LTP and $I$ is $(k, \infty)$, $(0, 0)$ or $(0, \infty)$.

**Theorem 20.** $L(A) \neq WM(T\Sigma^\omega)$ if and only if there are TPs $g, h \in \mathcal{I}$ such that $h$ is idempotent and bad for $g(L_0)$.

**Proof.** ($\Leftarrow$): We assume that we have two TPs $g^{I_0}$ and $h^{I_0}$ from $\mathcal{I}$ transition profiles such that $h^{I_0}$ is idempotent and bad for $g(L_0)$. By Lemma 14 we get a well matched word $u \in WM(T\Sigma^+)$ corresponding to TP $g^{I_0}$ such that there is a run $(\ell_0, ([\bot], \dagger)), (\ell_1, (\sigma_1, \text{age}(\sigma_1))), \cdots (\ell_n, ([\bot], \dagger))$ such that $u \in I_g$ and $g^{I_0}_{u} = g^{I_0}$; and another well matched word $v \in WM(T\Sigma^+)$ corresponding to TP $h^{I_0}$ having a run $(\ell_{i_0}, ([\bot], \dagger)), (\ell_{i_1}, (\pi_1, \text{age}(\pi_1))), \cdots (\ell_{i_k}, ([\bot], \dagger))$ such that $v \in I_h$ and $I_h \neq [0, 0]$, with $h^{I_0}_{u} = h^{I_0}$. Since $h^{I_0}$ is an idempotent using Remark 19 we get that $I_h = (k, \infty)$ or $(0, 0)$ or $(0, \infty)$. For the following discussion we take $I_h = (k, \infty)$, and the cases $(0, \infty)$ and $[0, 0]$ are dealt in the similar manner.

From these two words we construct $u^{\omega}$ which is infinite and well-matched.

We now need to prove that $u^{\omega} \notin L(A)$, which we prove by contradiction. So we assume $u^{\omega} \in L(A)$. Let $\rho = (\ell_0, ([\bot], \dagger)), (\ell_1, (\sigma_1, \text{age}(\sigma_1))), \cdots (\ell_n, ([\bot], \dagger)), (\ell_{i_0+1}, (\pi_1, \text{age}(\pi_1))), \cdots (\ell_{i_k+1}, ([\bot], \dagger)) \cdots$ be an accepting run of $A$ on it. Since $A$ has a run on $u$, $g^{I_0}(\ell_0, \ell_{i_0}) \neq \dagger$; although it may be even or odd. Therefore from definition of the set $g^{I_0}(\ell_0)$, we get $\ell_{i_0} \in g^{I_0}(\ell_0)$. Extending this by $v$, we have word $uv$ on which $A$ has a run therefore, $h^{I_0}(\ell_{i_0}, \ell_{i_0+1}) \neq \dagger$. Hence we get $\ell_{i_0+1} \in h^{I_0}(\ell_{i_0+1})$, and $\ell_{i_0+1} \in h^{I_0} \circ g^{I_0}(\ell_0)$. We repeat this procedure for each subword $v_j$, $j \in \mathbb{N}$, and since $h^{I_0}$ for each segment $v_j$ exists we get $h(\ell_{i_0+j+1}, \ell_{i_0+j+1+1}) \neq \dagger$.

Note that as every finite prefix $uv^R$ is well-matched, stack is empty in all $(\ell_{i_0+j+1}, ([\bot], \dagger))$ configurations in the run. Let $R = \{r_0, r_1, \cdots\}$ be an infinite set of positions in run $\rho$ where stack becomes empty and for each word $v_{r_1} \cdots v_{r_j}$, its duration is more than $k$ (the maximal constant occurring in pop constraints).

Note that $r_0 = 0$ is the initial position and each word $v_{r_1} \cdots v_{r_j}$ such that $j < j'$ is well-matched for all $j, j'$ in $\mathbb{N}$. Since each such word is well-matched, TP associated is well-defined. So the TP associated with the word $v_{r_1} \cdots v_{r_j}$ is $h^{I_0}_{v_{r_1} \cdots v_{r_j}}$. Here $I_{r_1} \circ \cdots \circ I_{r_j}$ is the time interval associated with it, and by Remark 19 it is either $(k, \infty)$ or $(0, 0)$ or $(0, \infty)$. For ease of the following discussion we use $I$ to denote the interval $(k, \infty)$, the other cases are dealt in similar manner.

Let $S$ be the infinite sequence of locations corresponding to positions in $R$ in the run $\rho$. As the set of locations $L$ is finite some location $\ell'$ occurs at $\ell_{i_0+j+1}$ for infinitely many values of $j$ in the infinite sequence $S$. Let $i_0 < i_1 < i_2 < \cdots$ be the sequence of indices at which $\ell'$ occurs. We create another sequence $C = c_0, c_1, c_2, \cdots$ where $c_m = \bigcup \{\Omega(s_{i_m}), \cdots, \Omega(s_{i_{(m+1)-1}})\}$, for
all m in N. Applying Lemma 6, we get \( \bigcup \inf(S) = \bigcup \inf(C) \) if \( i_0 \) is chosen sufficiently large. As \( h^j \) is idempotent, \( (h^j)^{i+1-i_j} = h^j \) for all \( j \) in N. Hence, \( c_j \leq (h^j)^{i+1-i_j} = h^j \) for all \( j \). As the run is accepting, \( \bigcup \inf(S) = \bigcup \inf(C) \) is even. Thus, infinitely many \( c_j \)'s must be even resulting in \( h^j (\ell', \ell') \) be even, which contradicts our earlier assumption of \( h^j \) being bad.

\((\Rightarrow)\): We are given that \( L(A) \neq WM(T \Sigma^\omega) \). Thus there exists an infinite well-matched timed word \( w = (a_0, \delta_0), (a_1, \delta_1), (a_2, \delta_2), \ldots \) which is not in \( L(A) \). Let \( \rho = q_0 \xrightarrow{(a_0, \delta_0)} q_1, \xrightarrow{(a_1, \delta_1)} q_2, \ldots \) be some run of \( w \) on \( A \) with configurations \( q_i = (\ell_i, (\sigma_i, age(\sigma_i))) \). As \( w \) is well-matched there are no pending call positions, hence there are infinitely many positions \( j \) such that stack is empty in \( q_j \). Let \( q_m, q_n \) with \( m < n \) be two such configurations in \( \rho \). Let \( w[m, n] = (a_m, \delta_m), (a_{m+1}, \delta_{m+1}), \ldots, (a_n, \delta_n) \) denote the subword of \( w \) from position \( m \) to \( n \). Since \( w \) is progressive timed word, we can construct the sequence of infinitely many positions \( i_0 < i_1 < i_2 < \cdots \) where at all configurations \( q_{i_j} \) dsVPA \( A \) have empty stack and subword \( w[i_j, i_j+1] \) allows us to go from configuration \( q_{i_j} \) to \( q_{i_{j+1}} \), and has duration \( |w[i_j, i_j+1]| > k \). Let \( R := \{i_0, i_1, i_2, \ldots \} \) be this set at which stack becomes empty.

Note that all \( w[i_j, i_j+1] \) are well-matched as \( q_{i_j} \) and \( q_{i_{j+1}} \) have empty stacks. By Lemma 15, LTPs for \( w[i_j, i_j+1] \) exist and we denote it by \( f_{[i_j, i_{j+1}]}^{(k, \infty)} \) respectively. Since the duration of corresponding subword is \( > k \), we have that interval \( I_{j,j+1} \) to be \((k, \infty)\) for all \( j, j+1 \in N \). so we denote previous TPs as \( f_{[i_j, i_{j+1}]}^{(k, \infty)} \) respectively.

Using Lemma 17 transition profile \( f_{[i_j, i_{j+1}]}^{(k, \infty)} \) must be in \( \mathcal{Z} \). Here each LTP is subscripted with the position indices for simplifying the discussion. Let \( R^{(2)} := \{(i_j, i'_j) \mid j, j' \in N \land j < j'\} \). Consider coloring \( \chi : R^{(2)} \rightarrow \mathcal{Z} \) which is defined as \( \chi(i, i') = f_{[i_j, i_{j+1}]}^{(k, \infty)} \). As \( \mathcal{Z} \) is finite, using Theorem 7 we get an infinite subset \( J \) of \( R \), and an LTP \( g \in \mathcal{Z} \) such that \( \chi(j, j') = g \) for all \( j, j' \in J \) and \( j < j' \).

Let \( J = \{j_0, j_1, j_2, \cdots \} \) with \( 0 < j_0 < j_1 < j_2 < \cdots \). Let \( f = \chi(0, j_0) \) be LTP. We claim that \( g \) is idempotent as \( g \circ g = \chi(j_0, j_1) \circ \chi(j_1, j_2) = g_{[j_0, j_2]}^{(k, \infty)} \circ g_{[j_0, j_1]}^{(k, \infty)} = g_{[j_0, j_2]}^{(k, \infty)} = g_{[j_0, j_1]}^{(k, \infty)} \). Now we show that \( g \) is bad for \( f(L^0) \) using contradiction.

Suppose \( g \) is not bad for \( f(L^0) \) the there is a location \( \ell \) such that \( \ell \) is in \( g(\ell') \), \( \ell' \) in \( f(L_0) \), and \( g(\ell, \ell) = m \) where \( m \) is some even parity. This allows us to construct a run \( \rho = (\ell_0, (\bot, \bot)) \xrightarrow{m_0} (\ell_1, (\bot, \bot)) \xrightarrow{m_1} (\ell_2, (\bot, \bot)) \xrightarrow{m_2} \cdots \). Thus, the maximum parity of infinitely occurring location is \( m \), which is even. Hence, this run is accepted by \( A \), which is a contradiction.

Using Theorem 20 we give Algorithm 1 to decide universality of given dsVPA. This can be extended for non well-matched words as done in Friedmann’s [9].

Theorem 21. Let \( A \) be a dsVPA over visible alphabet \( \Sigma \), with \( n = |Q| \) number of states and stack alphabet \( \Gamma \). Let \( m = \max\{1, |\Sigma|, |\Gamma|\} \). Let \( k \) be the index and \( d \) be the maximal constant used in the time constraints over stack elements. The maximum time taken by Algorithm 1 is \( O(2^d n^d \log(n) m^3 n^3 (d + 1)^4) \).
Algorithm 1: Universality checking for dsVPA

Input: $T_{\text{call}}, T_{\text{return}}, T_{\text{local}}$, for dsVPA $A$
Output: “No” if $\text{Lang}(A) \neq \text{WM}(T\Sigma^*)$; “Yes” otherwise.

1. Initializations: $X = T_{\text{local}} \cup T_{\text{call}} \circ T_{\text{return}}$; $T = X$.
2. while $X \neq \emptyset$ do
   3. for $f^I_{\ell}$ and $g^I_{\ell}$ in $T \times X \times X \times T$ do
      4. if $g^I_{\ell}$ is idempotent and bad for $f^I_{\ell}$ then
         5. return “No”;
      6. $X = (X \circ T \circ T \circ T_{\text{call}} \circ T_{\text{return}}) \setminus T$;
         // reaching fixed point if possible by adding new TPs
      7. $T = T \cup X$;
   8. return “Yes”;

4 Inclusion Checking of dsVPA

In this section we describe how to check language inclusion of two dsVPA s $A_1$ and $A_2$, using ideas similar to [9]. First we extend transition profiles of an automaton with the transitions to get tagged transition profiles (TP)s.

A TP associated with a local transition of a dsVPA is of the type: $(L \times \Omega(L) \times L) \times (\Delta L \times L \rightarrow \Omega(L))$. As an example, consider a TP $\langle (\ell, c, \ell'), f^I \rangle$ where $f^I$ is a TP. On a word $u$, TP $f^I$ describes all runs of machine $A_2$ on $u$, while $(\ell, c, \ell')$ describes some run of machine $A_1$, which starts at location $\ell$, ends in location $\ell'$, with maximal priority $c$, and happens in interval $I$.

A TP associated with a call transition of a dsVPA is of the following type: $(L \times \Omega(L) \times \Gamma \times L) \times (\Delta L \times \Gamma \times L \rightarrow \Omega(L))$. For example $\langle (\ell, c, \gamma, \ell', f^I) = (I, \ell_1, \gamma_1, \ell_2, d) \rangle$ is a TTP of call-type in which $f^I$ is call type TP. As in the case of local type TTP, $(\ell, c, \gamma, \ell')$ denotes some run of $A_1$ on $u$ starting in location $\ell$, ending in location $\ell'$, with maximal occurring priority $c$, and in interval $I$, while TP $f^I$ denote all runs of $A_2$ on $u$. Therefore, we do not have TTPs with in which tags have different intervals and TP has a different interval, as it is trivially ruled out for inclusion checking.

A TP associated with a return transition of a dsVPA is of the following type: $(L \times \Omega(L) \times \Gamma \times L) \times (\Delta L \times \Gamma \times L \rightarrow \Omega(L))$. For example $\langle (\ell, c, \gamma, \ell', f^I) = (I, \ell_1, \gamma_1, \ell_2, d) \rangle$ is a TTP of return-type in which $f^I$ is return-type TP. As in the case of local type TTP, $(\ell, c, \gamma, \ell')$ denotes some run of $A_1$ on $u$ starting in location $\ell$, ending in location $\ell'$, with maximal occurring priority $c$, and in interval $I$, while TP $f^I$ denote all runs of $A_2$ on $u$.

As in the case of TPs we associate TTP s with a letter $a$ denoted as $F^I_a$, as follows: Here $f^I_a$ is a TP of machine $A_2$ for letter $a$.

- If $a \in \Sigma_t$, $F^I_a = \{((\ell, c, \ell'), f^I_a) \mid \ell, \ell' \in L_1 \text{ and } (p, a, p') \in \Delta^I_1\}$.}
- If $a \in \Sigma_c$, $F^I_a = \{((\ell, c, \gamma, \ell'), f^I_a) \mid \ell, \ell' \in L_1 \text{ and } (p, a, p') \in \Delta^I_1\}$.}
- If $a \in \Sigma_r$, $F^I_a = \{((\ell, c, \gamma, \ell'), f^I_a) \mid \ell, \ell' \in L_1 \text{ and } (p, a, p') \in \Delta^I_1\}$.}
Now we define compositions of TTPs to associate TTP with a word. These compositions are defined when the composition of underlying TPs is defined with the added restriction that “tags” of the component TTPs should also match.

- If \((\ell, c, \ell', f^I)\) and \((\ell_1, d, \ell_2, g^I)\) are two TTPs of local type then their composition \((\ell, c, \ell', f^I) \circ (\ell_1, d, \ell_2, g^I) = (\ell, [c \cup d], \ell_2, (f \circ g)^I)\) is defined if \(\ell' = \ell_2\) and \(I = I'\).
- If \((\ell, c, \gamma, \ell', f^I)\) is a call TTP and \((\ell_1, d, \ell_2, g^I)\) is a local TTP then their composition is a call TTP and is defined as follows: \((\ell, c, \gamma, \ell', f^I) \circ (\ell_1, d, \ell_2, g^I) = (\ell, [c \cup d], \gamma, \ell_2, (f \circ g)^I)\) if \(\ell' = \ell_2\) and \(I = I'\).
- If \((\ell, c, \gamma, \ell', f^I)\) is a return TTP and \((\ell_1, d, \ell_2, g^I)\) is a local TTP then their composition is a return TTP defined as follows: \((\ell, c, \gamma, \ell', f^I) \circ (\ell_1, d, \ell_2, g^I) = (\ell, \{c, d\}, \gamma, \ell_2, (f \circ g)^I)\) if \(\ell' = \ell_2\) and \(I = I'\).
- If \((\ell, c, \gamma, \ell', f^I)\) is a call TTP and \((\ell_1, d, \gamma', \ell_2, g^I)\) is a return TTP then their composition, \((\ell, c, \gamma, \ell', f^I) \circ (\ell_1, d, \gamma', \ell_2, g^I) = (\ell, \{c, d\}, \gamma, \ell_2, (f \circ g)^I)\) if \(\ell' = \ell_2\) and \(I = I'\) such that: (1) \(d\) is an even priority and (2) the TP \(g^I\) is an idempotent and bad for the set \(f^I(\ell_0^I)\).

For two sets \(F\) and \(G\) of TTPs their composition \(F \circ G = \{ f \circ g \mid f \in F \text{ and } g \in G, f \circ g \text{ is defined} \}\). Let \(T_{local}, T_{call}\) and \(T_{return}\) denote all TTPs of local type, call type, and return type respectively. Let \(T = T_{local} \cup T_{call} \cup T_{return}\). As in the case of TPs we define a set \(X\) as a least fixed point of the function \(X = T_{local}, T_{call} \circ T_{return} \cup T_{call} \circ X \cup T_{return} \cup X \circ X\).

**Theorem 22.** \(\text{Lang}(A_1) \nsubseteq \text{Lang}(A_2)\) iff there exist two TTPs \((\ell_0^I, c, \ell', f^I)\) and \((\ell_1, d, \ell_1^I), g^I)\) in \(T\) such that: (1) \(d\) is an even priority and (2) the TP \(g^I\) is an idempotent and bad for the set \(f^I(\ell_0^I)\).

Based on Theorem 22 we give an Algorithm 2, for checking language inclusion of two dsVPA\(s\), whose time complexity is given in the Theorem 23.

**Algorithm 2:** Inclusion checking for dsVPA

**Input:** \(T_{call}, T_{return}, T_{local}\) for dsVPA \(A = A_1 \sqcup A_2\).

**Output:** “No” if \(\text{Lang}(A_1) \nsubseteq \text{Lang}(A_2)\); “Yes” otherwise.

1. **Initializations:** \(X = T_{local} \cup T_{call} \circ T_{return}; T = X;\)
2. **while** \(X \neq \emptyset\) **do**
3.  **for** \((\ell, c, \ell', f^I)\) and \((r, d, r', g^I)\) in \(T \times X \cup X \times T\) **do**
4.  **if** \(\ell = \ell_0, \ell' = r = r'\) and \(d\) is even and \(g^I_w\) idempotent and bad for \(f^I_{w'}\) **then**
5.     **return** “No”;
6.  \(X = (X \circ T \cup T \circ X \cup T_{call} \circ X \circ T_{return}) \setminus T;\)
7.  // reaching fixed point if possible by adding new TTPs in the iteration
8.  **return** “Yes”;
Theorem 23. Let $A_1$ and $A_2$ over visible alphabet $\Sigma$, be two dsVPA s with $n_1 = |Q_1|$ and $n_2 = |Q_2|$ number of states and stack alphabets $\Gamma_1$ and $\Gamma_2$ respectively. Let $m = \max\{1, |\Sigma|, |\Gamma_i|\}$. Let $d_1$ and $d_2$ be the index of $A_1$ and $A_2$ respectively. Let $k_1$ and $k_2$ be the maximal constants used in the time constraints over stack elements. The maximum time taken by Algorithm 2 is $O(n_1^3 \cdot n_2^5 \cdot d_1^3 \cdot d_2^2 \cdot m_2 \cdot k_1^2 \cdot 2^{O(n_2^2 d_2^2 \log (k_2))})$.

5 Conclusion

We explored a Ramsey-based algorithm to decide language inclusion for dsVPA over infinite timed words. Antichains approach [16, 8] is an alternative method for inclusion checking of VPA [7]. We would like to explore both these techniques to dsVPA extended with multiple stacks.

References

Appendix

Proof of inclusion checking (Theorem 22)

Proof. We assume a single dsVPA $A$ obtained by taking disjoint union of $A_1$ and $A_2$.

$(\Rightarrow)$ Let $w \in \text{Lang}(A_1) \cap \text{WM}(\Sigma) \setminus \text{Lang}(A_2)$. So we have an accepting run of $A_1$ on $w$ and hence of $A$ starting at location $l^A_0$. That gives us a maximal priority occurring infinitely often in the run as some even number $d$. Let us choose a position in the run after which $d$ is the maximal priority and before that let $c$ be the maximal occurring priority. Since $w$ is a well matched infinite word, in the associated run of $A_1$ its stack becomes empty infinitely often. Let $R = \{r_0, r_1, \ldots\}$ be an infinite set of positions in the run at which stack of $A_1$ becomes empty and for each word $v_{r_1} \ldots v_{r_j}$, its duration is more than $k$ (the maximal constant occurring in pop constraints). For simplifying proof, choose $r_1$ to be greater than $m$. Note that $i_0 = 0$ is the initial position, and each word $v_{r_1} \ldots v_{r_j}$ such that $j < j'$ is well-matched for all $j, j'$ in $\mathbb{N}$. Since each such word is well-matched, TP associated is well-defined.

Now we color pairs $(r_j, r'_j)$ with TP $f^{r_j r'_j}$, the TP associated with the word $v_{r_j} \ldots v_{r'_j}$. Here $I_{r_j r'_j}$ is the time interval associated with it, and by Remark 19 it is either $(k, \infty)$ or $(0, 0)$ or $(0, \infty)$. But since we are assuming non-zeno behaviors we have $I_{r_j r'_j} = (k, \infty)$ or $(0, \infty)$. To simplify notations we use $I$ to denote the interval $(k, \infty)$, in the following discussion, the case of $(0, \infty)$ is dealt in similar manner. Now we use Theorem 7 to get an infinite subset $B = \{b_0, b_1, \ldots\}$ of positions $R$ such that a TP color of each pair $(b_j, b'_j)$ is a TP $g^I$ such that $b_j, b'_j$ in $B$ and $j < j'$. By Theorem 20 we know that this TP is idempotent and bad for $f^I(l^I_0)$.

Using $f^I$ and $g^I$ in the rest of proof we construct required TTPs as follows. In the run of $A$ on $w$ by Pigeon hole principle some state must occur infinitely often. In particular consider two positions $j$ and $j'$ such that $j < j'$ and $j$ is the first time when this location occurs after position $b_0$ from $B$, which ensures that $j > m$. Let the set of positions where location $l$ occurs is $J = \{j_0, j_1, \ldots\}$ where $j_0 = j$ and $j' = j_1$. Now from word $w$ we take the initial segment $u = a_{j_0} \ldots a_{j_{-1}}$ and the later segment $v_i = a_{j_i} \ldots a_{j_{i+1}} - 1$ for all $i > 0$. Now the TP for the initial segment $u$ is composed of $f^I$ and some $j' - j$ times $g^I$. Let us call it $f^I = f^I \circ g^I \circ \ldots g^I$. Now for the segment $v_{j'-j} = a_{j_0} \ldots a_j$ we have TP $g^I \circ \ldots \circ g^I$ composed $j' - j$ times. Since $g^I$ is idempotent we get $g^I_{a_{j_0} \ldots a_{j_{-1}}} = g^I$.

Take $g^I = g^I$ if $j = 0$ then we get $f^I = f^I$ else $f^I = f^I \circ g^I$.

Most importantly TP we have $g^I$ is bad for $f^I(l^I_0)$, else it contradicts with that fact that $g^I$ was bad for $f^I(l^I_0)$ taking $p$ as its witness. Therefore we have TPs $f^I$ and $g^I$ such that $f^I$ can be tagged with $(l^I_0, c, l)$ and $g^I$ can be tagged with $(l, d, l)$, to get TTPs $(l^I_0, c, l), f^I)$ and $(l, d, l), g^I)$ respectively as required, whose compositions are well defined.

$(\Leftarrow)$ Let $(l^I_0, c, l), f^I)$ and $(l, d, l), g^I)$ be the TTPs satisfying conditions (1) and (2). Let $u$ be one matched word corresponding to $f^I$ and let $v$ be a well matched word corresponding to $g^I$. Then we have a run of $A_1$ on $uv^\omega$ as follows:
\( \rho = (\ell_0, \langle \cdot \rangle), (\ell_1, \langle \sigma_1, \text{age}(\sigma_1) \rangle), \cdots (\ell_{|u|}, \langle \cdot \rangle), (\ell_{|u|+1}, \langle \pi_1, \text{age}(\pi_1) \rangle), \cdots (\ell_{|u|+|v|}, \langle \cdot \rangle) \) ...

... So we have that stack contents are empty in the beginning, after reading \( u \) and afterwards whenever \( v^t \) is processed, because \( u \) and \( v \) are well matched.

For this infinite run one priority which occurs infinitely is \( d \), which is even by condition (1). Since \( d \) is also maximal, we get that \( uvw^2 \) is accepted by \( A_1 \). Now condition (2) using Theorem 20 ensures that \( uvw^2 \) is not accepted by \( A_2 \).

\[ \square \]

**Time complexity analysis of Algorithm 1 and Algorithm 2**

**Proof (Theorem 21).** Maximum time required for checking equality of two TPs is \( O(n^2) \). Composition of two TPs takes time \( O(n^3m) \). To check if the TP is idempotent takes time \( O(n^3m) \). To check if the TP is bad takes time \( O(n) \).

Total number of LTPs is bounded by \( 2^{O(n^2d^2\log(k))} \). We use bit vector notation to store TPs which need a bit vector of length \( O(n^2d^2\log(k)) \). Total time needed to add or remove one TP requires one iteration over the bit vector requiring \( O(n^2d^2\log(k)) \) time. Therefore to compute set difference or union of two sets of TP take at most \( O(n^2d^2\log(k)) \) time. For line 3 to line 5 the while loop is iterated at most \( |N| \) times, which is bounded by \( 2^{O(n^2d^2\log(k))} \). So the total time spend is \( 2^{O(n^2d^2\log(k))} \).

In line 6, the total number of compositions is bounded by \( O(|N| \cdot |T| + |T_{\text{call}}| \cdot |N| \cdot |T_{\text{return}}|) \) which is bounded by \( O(|T|^2 + |N|m^2(d + 1)^4) \), since \( |N| \leq |T| \) and \( |T_{\text{call}}| \leq |\Sigma|(d + 1)^2 \) and \( |T_{\text{return}}| \leq |\Sigma|(d + 1)^2 \).

But \( |T| \) is bounded by total number of LTPs \( 2^{O(n^2d^2\log(k))} \), a composition takes \( O(n^3m) \) time and operations set difference, set union taking time at most \( O(n^2d^2\log(k)) \) time, the total time spend in line 6 and 7 is \( O(2^{O(n^2d^2\log(k))}n^{3}\log^{3}(d + 1)^4) \).

\[ \square \]

**Proof (Theorem 23).** Maximum time required for checking equality of two TTPs is \( O(n_2^3) \). Composition of two TTPs takes time \( O(n_2^2m_2) \). To check if the TTP is idempotent takes time \( O(n_2^3m_2) \). To check if the TTP is bad takes time \( O(n_2^3) \).

Total number of local type TTPs is bounded by \( z = (n_2^2k_1(d_1+1)^2)2^{O(n_2^2d_2^2\log(k_2))} \).

We use bit vector notation to store TTPs so that To store any set of TTPs we need a bit vector of length \( \log(n_2^2k_1(d_1+1)^2)) + O(n_2^2d_2^2\log(k_2)) \), which is \( b = O(n_2^2d_2^2d_1k_1\log(k_2)) \).

Total time needed to add or remove one TTP requires one iteration over the bit vector requiring \( b \) time. Therefore to compute set difference or union of two sets of TTPs take at most \( O(b) \) time. For line 3 to line 5 the while loop is iterated at most \( |N| \) times, which is bounded by \( z \). So the total time spend is \( z \). In line 6, the total number of compositions is bounded by \( O(|N| \cdot |T| + |T_{\text{call}}| \cdot |N| \cdot |T_{\text{return}}|) \) which is bounded by \( O(|T|^2 + |N|m_2^2(d_2 + 1)^4) \), since \( |N| \leq |T| \) and \( |T_{\text{call}}| \leq |\Sigma|(d_2 + 1)^4 \) and \( |T_{\text{return}}| \leq |\Sigma|(d_2 + 1)^4 \).

But \( |T| \) is bounded by total number of internal TTPs \( z \), a composition takes \( O(n_2^2m_2) \) time and operations set difference, set union takes at most \( O(b) \).
Therefore the total time spend in line 6 and 7 is $z \cdot b \cdot O(n_1^2 m_2)$, which is $n_1^3 \cdot n_2^5 \cdot d_1^3 \cdot d_2^2 \cdot m_2 \cdot k_1^2 \cdot 2^O(n_2^{2d_2 \log(k_1)})$, as claimed.

\[ \square \]

**Proofs from Section 3**

**Proof (Lemma 14).** We prove stronger claim using the induction on the length of timed word. Our induction hypothesis is if $f^l = f^l_1 \circ f^l_2 \circ \cdots \circ f^l_m$ is LTP with all $f^l_i$ are atomic TPs, then there is a well-matched finite timed word $w$ of length $m$ such that $\ell \xrightarrow{w} \ell'$ and $[w] \in I$.

There are two possible cases either $f^l$ is atomic or composite. When $f^l$ is atomic LTP, $f^l_i$ must belong to set of all atomic LTPs - $T_{\text{local}}$ such that $f^l_i$ is $f^l_i$ for some local symbol $a$. Thus, we choose timed word $w = (a, \delta)$ such that $\delta \in I$. As $f^l_i(\ell, \ell') = p$, Definition 9 ensures that there must be transition from $\ell$ to $\ell'$ on $a$ and $\Omega(\ell') = p$. Hence we construct following run for $w$: $(\ell, \{\sigma, \text{age} (\sigma)\}), (\ell', \{\sigma, \text{age} (\sigma)\})$. Induction hypothesis clearly holds for this base case.

When $f^l$ is composite we get three possible cases as per Proposition 13. Case (i): When $f^l = g^{h_1} \circ g^{h_2}$ such that $g^{h_1} = f^l_1 \circ \cdots \circ f^l_i$ and $g^{h_2} = f^l_{i+1} \circ \cdots \circ f^l_m$ are LTPs. By induction hypothesis we get following runs corresponding to $g^{h_1}$ and $g^{h_2}$. Let the TP $g^{h_1}$ yield a run $\rho_1 \colon (\ell_0, \{\sigma, \text{age} (\sigma)\}) \cdots (\ell_i, \{\sigma, \text{age} (\sigma)\})$ with parity $p_1$ on word $w_1$ with length $t$ and $[w_1] \in I_1$. Let the TP $g^{h_2}$ have a run $\rho_2 \colon (\ell_{i+1}, \{\sigma, \text{age} (\sigma)\}) \cdots (\ell_m, \{\sigma, \text{age} (\sigma)\})$ with parity $p_2$ on word $w_2$ with length $m-t$ and $[w_2] \in I_2$. By Definition 10, $I = I_1 + I_2$. As composition $f^l$ is defined, there will be at least on location $\ell''$ such that $f^l(\ell, \ell') = g^{h_1}(\ell, \ell'') \circ g^{h_2}(\ell'', \ell')$, otherwise composition $f^l$ is undefined. Definition 10 permits us to choose that $\ell''$ where $p = p_1 \sqcup p_2$ holds and then we concatenate $p_1$ with $p_2$ to construct a run $(\ell_0, \{\sigma, \text{age} (\sigma)\}) \cdots (\ell_i, \{\sigma, \text{age} (\sigma)\}) \cdots (\ell_m, \{\sigma, \text{age} (\sigma)\})$ for timed word $w = w_1 \cdot w_2$ of length $m$ and $[w_1, w_2] = ([w_1] + [w_2]) \in I$.

Case (ii): When $f^l = f^l_1 \circ f^l_2$ such that $f^l_1$ is atomic CTP, and $f^l_2$ is atomic RTP. As both $f^l_1$ and $f^l_2$ are atomic, induction hypothesis clearly holds from the Definition 9.

Case (iii): When $f^l = f^l_1 \circ g^{h_s} \circ f^l_m$ such that $f^l_1$ is atomic CTP, $g^{h_s} = f^l_2 \circ \cdots \circ f^l_{m-1}$ is LTP, and $f^l_m$ is atomic RTP. From induction hypothesis TP $g^{h_s}$ yield a run $\rho_0 \colon (\ell_0, \{\tau_0, \sigma, \text{age} (\tau_0) \cdot \text{age} (\sigma)\}) \cdots (\ell_{m-1}, \{\tau_0, \sigma, \text{age} (\tau_0) \cdot \text{age} (\sigma)\})$ with parity $p_0$ on word $w_g$ with length $g$ and $[w_g] \in I_g$. Here $\sigma$ is the top of the stack and $\tau_0$ is the stack contents below the $\sigma$. Applying induction hypothesis on TPs $f^l_1$ and $f^l_2$ we get runs $\rho_1 \colon (\ell_0, \{\tau_0, \sigma, \text{age} (\tau_0) \cdot \text{age} (\sigma)\}) \cdots (\ell_{m-1}, \{\tau_0, \sigma, \text{age} (\tau_0) \cdot \text{age} (\sigma)\})$ and $\rho_2 \colon (\ell_{m-1}, \{\tau_0, \gamma, \text{age} (\tau_0) \cdot \text{age} (\gamma)\})$. Being atomic TPs, $f^l_1$ must belong to $T_{\text{call}}$ and $f^l_2$ must belong to $T_{\text{return}}$. Let $a, b$ be respective call and return symbols. Let $w_1 = (a, \delta_1)$ with $\delta_1 \in I_1$ and $w_2 = (b, \delta_2)$. As the $f^l_1 \circ g^{h_s} = h^{f_1 + h_s}$ is defined, there must exist $\ell'$ and $\gamma$ such that run $z_0 = z_0$. Hence we get a run for word $w' = (a, \delta_1) \cdot w_g$ with $[w'] \in I_1 + I_g$ having parity $p' = p_1 \sqcup p_0$. Applying RTP rule of Definition 10, we get $(I_1 + I_g) \cap I_2 \neq \emptyset$, and $\sigma = \gamma$. Thus, $\ell''$ can be chosen such that $p = p' \sqcup p_2 = p_1 \sqcup p_2 \sqcup p_2$. We get corresponding timed word $w = w' \cdot w_2 = (a, \delta_1) \cdot w_g \cdot (b, \delta_2)$.

\[ \square \]
Proof (Sketch of Lemma 15). Given a run $\rho = \ell \xrightarrow{w} \ell'$, we first identify matching call and returns. We then guess TP corresponding to each transition of the $\rho$. Proof proceeds by constructing TPs by repeatedly applying composition rules given in Definition 10 and using associativity of compositions given by Proposition 11 and Proposition 12.

Proof (Sketch of Lemma 17). If $w$ is well-matched word, then by Lemma 15, $f$ exists and is an LTP. Further using induction we show it to be in $\mathcal{I}$. The structural induction on $\mathcal{I}$ using Definition 16 proves that $\mathcal{I}$ only contains only well-matched words.

Proof (Theorem 5). Given two dsVPA $s A_1 = (L_1, \Gamma_1, L_0_1, \Delta_1 = \Delta_1^c \cup \Delta_1^l \cup \Delta_1^r, \Omega_1)$ and $A_2 = (L_2, \Gamma_2, L_0_2, \Delta_2 = \Delta_2^c \cup \Delta_2^l \cup \Delta_2^r, \Omega_2)$ defined over $\Sigma = \langle \Sigma_c, \Sigma_r, \Sigma_l \rangle$ getting dsVPA for union is straightforward, and is obtained by taking disjoint union of the locations, initial locations, final locations(in the case of finite words) and transitions.

The dsVPA $A$ for intersection of $A_1$ and $A_2$ is obtained by as follows: locations(initial, final(in the case of finite words)) of $A$ are product of locations(initial, final) $A_1$ and $A_2$. Stack alphabet $\Gamma = \Gamma_1 \times \Gamma_2$. The priority of a location $(l_1, l_2) \in L_1 \times L_2$ of machine $A$ is defined as $\Omega(l_1, l_2) = \max \{ \Omega_1(l_1), \Omega_2(l_2) \}$. When a call symbol $a$ is the current input symbol then $(a, \beta)$ symbol is pushed on the stack with age 0. And, when a return symbol $b$ is the current input symbol then $(a, \beta)$ symbol is popped from the stack if their age lies in the $I_1 \cap I_2 = \emptyset$, if the corresponding return transitions for the individual machines are $(l_1, b, I_1, \alpha, l_1') \in \Delta_1^r$ and $(l_2, b, I_1, \alpha, l_2') \in \Delta_2^r$.