Binary Reachability Analysis of Pushdown Timed Automata with Dense Clocks

Zhe Dang

School of Electrical Engineering and Computer Science
Washington State University, Pullman, WA 99164, USA
zdang@eecs.wsu.edu

Abstract. We consider pushdown timed automata (PTAs) that are timed automata (with dense clocks) augmented with a pushdown stack. A configuration of a PTA includes a control state, dense clock values and a stack word. By using the pattern technique, we give a decidable characterization of the binary reachability (i.e., the set of all pairs of configurations such that one can reach the other) of a PTA. Since a timed automaton can be treated as a PTA without the pushdown stack, we can show that the binary reachability of a timed automaton is definable in the additive theory of reals and integers. The results can be used to verify a class of properties containing linear relations over both dense variables and unbounded discrete variables. The properties previously could not be verified using the classic region technique nor expressed by timed temporal logics for timed automata and CTL* for pushdown systems.

1 Introduction

A timed automaton [3] can be considered as a finite automaton augmented with a number of dense (either real or rational) clocks. Due to their ability to model and analyze a wide range of real-time systems, timed automata have been extensively studied in recent years (see [1,29] for recent surveys). In particular, by using the standard region technique, it has been shown that region reachability for timed automata is decidable [3]. This fundamental result and the technique help researchers, both theoretically and practically, in formulating various timed temporal logics [2,4,5,6,22,25,26,27] and developing verification tools [21,28,24].

Region reachability is useful but has intrinsic limitations. In many real-world applications [11], we might also want to know whether a timed automaton satisfies a non-region (e.g., Presburger) property. Recently, Comon and Jurski [13] have shown that the binary reachability of a timed automaton is definable in the additive theory of reals, by flattening a timed automaton into a real-valued counter machine without nested cycles [12]. The result immediately paves the way for automatic verification of a class of non-region properties that previously were not possible using the region technique.

In this paper, inspired by Comon and Jurski's result [13], we consider *pushdown timed automata* (PTAs) that are obtained by augmenting timed automata with a pushdown stack. The main result in this paper gives a decidable binary reachability characterization for PTAs such that a class of non-region properties can be verified. A possible way to show this result to look at the flattening technique of Comon and Jurski's to see whether the technique can be adapted by adding a pushdown stack. However, this approach has an inherent difficulty: the flattening technique, as pointed out in their paper, destroys the structure of the original timed automaton, and thus, the sequences of stack operations can not be maintained after flattening.

In this paper, we introduce a new technique, called the pattern technique, by separating a dense clock into an integral part and a fractional part. For a pair $(\boldsymbol{v}_0, \boldsymbol{v}_1)$ of two tuples of clock values, we define an ordering, called the pattern of $(\boldsymbol{v}_0, \boldsymbol{v}_1)$, on the fractional parts of $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$. An equivalent relation "$\approx$" is defined such that $(\boldsymbol{v}_0, \boldsymbol{v}_1) \approx (\boldsymbol{v}'_0, \boldsymbol{v}'_1)$ iff $\boldsymbol{v}_0$ and $\boldsymbol{v}'_0$ ($\boldsymbol{v}_1$ and $\boldsymbol{v}'_1$ will also) have the same integral parts, and both $(\boldsymbol{v}_0, \boldsymbol{v}_1)$ and $(\boldsymbol{v}'_0, \boldsymbol{v}'_1)$ have the same pattern. "$\approx$" preserves the binary reachability: $\boldsymbol{v}_0$ can reach $\boldsymbol{v}_1$ by a sequence of transitions iff $\boldsymbol{v}'_0$ can reach $\boldsymbol{v}'_1$ by the (almost) same sequence of transitions. Therefore, by preserving the (almost) same control structure, a PTA can be transformed into a discrete transition system (called the pattern graph) containing discrete clocks (for the integral parts of the dense clocks) and a finite variable over patterns. The pattern graph can be further reduced to a discrete PTA, whose binary reachability is decidable and can be accepted by a nondeterministic pushdown automaton augmented with reversal-bounded counters (NPCM) [15]. By translating a pattern back to a relation over the fractional parts of the clocks, the decidable binary reachability characterization (namely, $(\mathbf{D} + \mathbf{NPCM})$-definable) for PTAs can be derived. Given this characterization, it can be shown that the particular class of safety properties that contain mixed linear relations over both dense variables (e.g., clock values) and discrete variables (e.g., word counts) can be automatically verified for PTAs. In this extended abstract, all the proofs are omitted. For a complete exposition see [14].

2 Preliminaries

A nondeterministic multicounter machine is a nondeterministic machine with a finite number of states, a one-way input tape, and a finite number of integer counters. Each counter can be incremented by 1, decremented by 1, or stay unchanged. Besides, a counter can be tested against 0. A *reversal-bounded nondeterministic multicounter machine (NCM)* is a nondeterministic multicounter machine in which each counter is reversal-bounded (i.e., it changes mode between nondecreasing and nonincreasing for some bounded number of times). A *reversal-bounded nondeterministic pushdown multicounter machine (NPCM)* is an NCM augmented with a pushdown stack. It is known that the emptiness problem for NPCMs (and hence NCMs) is decidable [23].

Let $\mathbf{N}$ be integers, $\mathbf{D} = \mathbf{Q}$ (rationals) or $\mathbf{R}$ (reals), Γ be an alphabet. We use $\mathbf{N}^+$ and $\mathbf{D}^+$ to denote non-negative values in $\mathbf{N}$ and $\mathbf{D}$, respectively. Each value $v \in \mathbf{D}$ can be uniquely expressed as the sum of $\lceil v \rceil + \lfloor v \rfloor$, where $\lceil v \rceil \in \mathbf{N}$ is the integral part of v, and $0 \le \lfloor v \rfloor < 1$ is the fractional part of v. Given $m \ge 1$. Let x_i, y_i, and w_i be a dense variable over $\mathbf{D}$, an integer variable over $\mathbf{N}$, and a word variable Γ^*, for each $1 \le i \le m$, respectively. We use $\#_a(w_i)$ to denote a *count variable* representing the number of symbol $a \in \Gamma$ in w_i. A *linear term* t is defined as follows: $t ::= n \mid x_i \mid y_i \mid \#_a(w_i) \mid t - t \mid t + t$, where $n \in \mathbf{N}, a \in \Gamma$. A *mixed linear relation* l is defined as follows: $l ::= t > 0 \mid t = 0 \mid t_{discr}\ mod\ n = 0 \mid \neg l \mid l \wedge l$, where $0 \ne n \in \mathbf{N}$ and t_{discr} is a linear term not containing dense variables. A *dense* *linear relation* is a linear relation that contains dense variables only. A *discrete linear relation* is a linear relation that does not contain dense variables.

A tuple of integers and words can be encoded as a string by concatenating the unary representations of each integer and each of the words, with a separator $\# \notin \Gamma$. The domain of H, a predicate over integer variables and word variables, is the set of tuples of integers and words that satisfy H. H is an *NPCM predicate* (or simply NPCM) if there is an NPCM accepting the domain (encoded as a set of strings, i.e., a language) of H. A $(\mathbf{D}+\mathbf{NPCM})$*-formula* f is defined as follows: $f ::= l_{dense} \wedge H \mid l_{dense} \vee H \mid f \vee f$, where l_{dense} is a dense linear relation and H is an NPCM predicate. Given $p, q, r \geq 0$. A predicate A on tuples in $\mathbf{D}^p \times \mathbf{N}^q \times (\Gamma^*)^r$ is $(\mathbf{D}+\mathbf{NPCM})$*-definable* if there is a $(\mathbf{D}+\mathbf{NPCM})$-formula f with p dense variables, $p+q$ integer variables, and r word variables, such that, for all $x_1, \cdots, x_p \in \mathbf{D}$, $y_1, \cdots, y_q \in \mathbf{N}$, and $w_1, \cdots, w_r \in \Gamma^*$, $(x_1, \cdots, x_p, y_1, \cdots, y_q, w_1, \cdots, w_r) \in A$ iff $f(\lfloor x_1 \rfloor, \cdots, \lfloor x_p \rfloor, \lceil x_1 \rceil, \cdots, \lceil x_p \rceil, y_1, \cdots, y_q, w_1, \cdots, w_r)$ holds.

Lemma 1. *(1). Both $l_{discrete} \wedge H$ and $l_{discrete} \vee H$ are NPCM predicates, if $l_{discrete}$ is a discrete linear relation and H is an NPCM predicate. (2). NPCM predicates are closed under existential quantifications (over integer variables and word variables). (3). If A is $(\mathbf{D}+\mathbf{NPCM})$-definable and l is a mixed linear relation, then both $l \wedge A$ and $l \vee A$ are $(\mathbf{D}+\mathbf{NPCM})$-definable. (4). The emptiness (satisfiability) problem for $(\mathbf{D}+\mathbf{NPCM})$-definable predicates is decidable.*

3 Clock Patterns and Their Changes

A dense clock is simply a dense variable on $\mathbf{D}^+$. Fix a $k > 0$ and consider $k+1$ clocks $\boldsymbol{x} = x_0, \cdots, x_k$. For technical reasons, x_0 is an auxiliary clock indicating the current time now. Denote $K = \{0, \cdots, k\}$ and $K^+ = \{1, \cdots, k\}$. A subset K' of K is abused as a set of clocks; i.e., we say $x_i \in K'$ if $i \in K'$. A *(clock) valuation* $\boldsymbol{v}$ is a function $K \to \mathbf{D}^+$ that assigns a value in $\mathbf{D}^+$ to each clock in K. A *discrete (clock) valuation* $\boldsymbol{u}$ is a function $K \to \mathbf{N}^+$ that assigns a value in $\mathbf{N}^+$ to each clock in K. For each valuation $\boldsymbol{v}$ and $\delta \in \mathbf{D}^+$, $\lceil \boldsymbol{v} \rceil$, $\lfloor \boldsymbol{v} \rfloor$ and $\boldsymbol{v}+\delta$ are valuations satisfying $\lceil \boldsymbol{v} \rceil(i) = \lceil \boldsymbol{v}(i) \rceil$, $\lfloor \boldsymbol{v} \rfloor(i) = \lfloor \boldsymbol{v}(i) \rfloor$ and $(\boldsymbol{v}+\delta)(i) = \boldsymbol{v}(i)+\delta$ for each $i \in K$. The *relative representation* $\widehat{\boldsymbol{v}}$ of a valuation $\boldsymbol{v}$ is a valuation satisfying: (1). $\lceil \widehat{\boldsymbol{v}} \rceil = \lceil \boldsymbol{v} \rceil$, (2). $\lfloor \widehat{\boldsymbol{v}} \rfloor(0) = \lfloor 1 - \lfloor \boldsymbol{v} \rfloor(0) \rfloor$, (3). $\lfloor \widehat{\boldsymbol{v}} \rfloor(i) = \lfloor \lfloor \boldsymbol{v} \rfloor(i) + \lfloor \widehat{\boldsymbol{v}} \rfloor(0) \rfloor$, for each $i \in K^+$. A valuation $\boldsymbol{v}_0$ is *initial* if clock x_0 has value 0, i.e., $\boldsymbol{v}_0(0) = 0$.

We distinguish two disjoint sets, $K^0 = \{0^0, \cdots, k^0\}$ and $K^1 = \{0^1, \cdots, k^1\}$, of indices. A *pattern* η is a sequence $p_0, \cdots, p_n$, for some $0 \leq n < 2(k+1)$, of nonempty and disjoint subsets of $K^0 \cup K^1$ such that $0^0 \in p_0$ and $\cup_{0 \leq i \leq n} p_i = K^0 \cup K^1$. p_i is called the *i-position*. A pair of valuations $(\boldsymbol{v}_0, \boldsymbol{v}_1)$ is *initialized* if $\boldsymbol{v}_0$ is initial. An initialized pair $(\boldsymbol{v}_0, \boldsymbol{v}_1)$ *has pattern* $\eta = p_0, \cdots, p_n$, written $(\boldsymbol{v}_0, \boldsymbol{v}_1) \in \eta$, if, for each $0 \leq m, m' \leq n$, each $b, b' \in \{0, 1\}$, and each $i, i' \in K$, $i^b \in p_m$ and ${i'}^{b'} \in p_{m'}$ imply that

$$\lfloor \widehat{\boldsymbol{v}_b} \rfloor(i) = \lfloor \widehat{\boldsymbol{v}_{b'}} \rfloor(i') \text{ (resp. } <) \text{ iff } m = m' \text{ (resp. } m < m').$$

Φ denotes the set of all the patterns ($|\Phi| \leq 2^{6(k+1)^2}$). The *now-position* of η is p_i, for some i, with $0^1 \in p_i$. A pattern is *regulated* if the now-position of η is p_0. A pattern is *initial* if it is the pattern of $(\boldsymbol{v}_0, \boldsymbol{v}_0)$ for some initial valuation $\boldsymbol{v}_0$. If η is the pattern of $(\boldsymbol{v}_0, \boldsymbol{v}_1)$, we use $init(\eta)$ to denote the pattern of $(\boldsymbol{v}_0, \boldsymbol{v}_0)$. $init(\eta)$ is unique for each η. A pattern is a *merge-pattern* if the now-position is a singleton set (i.e., 0^1 is the only element). A pattern is a *split-pattern* if it is not a merge-pattern, i.e., the now-position contains more than one element. A valuation $\boldsymbol{v}_1$ *has pattern* η if η is the pattern of $(\boldsymbol{v}_0, \boldsymbol{v}_1)$ for some $\boldsymbol{v}_0$. A pattern of $\boldsymbol{v}_1$ tells the fractional orderings between $\lfloor \boldsymbol{v}_1 \rfloor(i)$ and $\lfloor \boldsymbol{v}_1 \rfloor(j)$ and between $\lfloor \boldsymbol{v}_1 \rfloor(i)$ and 0, for all $i, j \in K^+$. Given two initialized pairs $(\boldsymbol{v}_0^1, \boldsymbol{v}_1)$ and $(\boldsymbol{v}_0^2, \boldsymbol{v}_2)$, we write $(\boldsymbol{v}_0^1, \boldsymbol{v}_1) \approx (\boldsymbol{v}_0^2, \boldsymbol{v}_2)$, if $(\boldsymbol{v}_0^1, \boldsymbol{v}_1)$ and $(\boldsymbol{v}_0^2, \boldsymbol{v}_2)$ have the same pattern, and have the same integral parts (i.e., $\lceil \boldsymbol{v}_0^1 \rceil = \lceil \boldsymbol{v}_0^2 \rceil, \lceil \boldsymbol{v}_1 \rceil = \lceil \boldsymbol{v}_2 \rceil$).

Example 1. Let $\boldsymbol{v}_0 = (0_{0^0}, 5.5_{1^0}, 2.3_{2^0})$ and $\boldsymbol{v}_1 = (1.6_{0^1}, 2.9_{1^1}, 3.1_{2^1})$, where subscripts are indices. Note that $\widehat{\boldsymbol{v}_0} = (0_{0^0}, 5.5_{1^0}, 2.3_{2^0})$ and $\widehat{\boldsymbol{v}_1} = (1.4_{0^1}, 2.3_{1^1}, 3.5_{2^1})$. The pattern η of $(\boldsymbol{v}_0, \boldsymbol{v}_1)$ can be drawn by collecting the fractional parts in $\widehat{\boldsymbol{v}_0}$ and $\widehat{\boldsymbol{v}_1}$ from small to large while writing down the indices; i.e., $\{0^0\}, \{2^0, 1^1\}, \{0^1\}, \{1^0, 2^1\}$. η is a merge-pattern. Take $\boldsymbol{v}_2 = \boldsymbol{v}_1 + .1$ and compute $\widehat{\boldsymbol{v}_2} = (1.3_{0^1}, 3.3_{1^1}, 3.5_{2^1})$. Observe that the fractional parts (except for the first component) are the same in $\widehat{\boldsymbol{v}_2}$ and $\widehat{\boldsymbol{v}_1}$. The pattern η' of $(\boldsymbol{v}_0, \boldsymbol{v}_2)$ can be drawn similarly: $\{0^0\}, \{2^0, 1^1, 0^1\}, \{1^0, 2^1\}$, which is the result of merging 0^1 to its previous position in η. η' is a split-pattern. Take $\boldsymbol{v}_3 = \boldsymbol{v}_2 + .05$. We can verify the pattern of $(\boldsymbol{v}_0, \boldsymbol{v}_3)$ is $\{0^0\}, \{0^1\}, \{2^0, 1^1\}, \{1^0, 2^1\}$, which is the result of splitting 0^1 from the now-position of η'. This procedure can go on while incrementing $\boldsymbol{v}_3$: merge 0^1 to the 0-position $\{0^0\}$, and then split 0^1 from it (by appending $\{0^1\}$ at the end), and so on. Eventually, the pattern will repeat when 0^1 returns to the original position in η (e.g., after a total increment of 1 from $\boldsymbol{v}_1$). ∎

For each $0 < \delta \in \mathbf{D}^+$, $\boldsymbol{v} + \delta$ is the result of a clock progress from $\boldsymbol{v}$ by an amount of δ. Function $next : \Phi \times (\mathbf{N}^+)^{k+1} \to \Phi \times (\mathbf{N}^+)^{k+1}$ describes how a pattern changes after a clock progress. Given any discrete valuation $\boldsymbol{u}$ and pattern $\eta = p_0, \cdots, p_n$ with the now-position being p_i for some i, $next(\eta, \boldsymbol{u})$ is defined to be $(\eta', \boldsymbol{u}')$ such that,

– (the case when η is a merge-pattern) if $i > 0$ and $|p_i| = 1$ (that is, $p_i = \{0^1\}$), then η' is $p_0, \cdots, p_{i-1} \cup \{0^1\}, p_{i+1}, \cdots, p_n$ (that is, η' is the result of merging the now-position to the previous position), and for each $j \in K^+$, if $j^1 \in p_{i-1}$, then $\boldsymbol{u}'(j) = \boldsymbol{u}(j) + 1$ else $\boldsymbol{u}'(j) = \boldsymbol{u}(j)$. Besides, if $i = 1$ (i.e., the now-position is merged to p_0; in this case, η' is a regulated pattern), then $\boldsymbol{u}'(0) = \boldsymbol{u}(0) + 1$ else $\boldsymbol{u}'(0) = \boldsymbol{u}(0)$,
– (the case when η is a split pattern) if $i \geq 0$ and $|p_i| > 1$, then η' is the result of splitting 0^1 from the now-position. That is, if $i > 0$, η' is $p_0, \cdots, p_{i-1}, \{0^1\}, p_i - \{0^1\}, p_{i+1}, \cdots, p_n$. However, if $i = 0$, η' is $p_0 - \{0^1\}, p_1, \cdots, p_n, \{0^1\}$. In either case, $\boldsymbol{u}' = \boldsymbol{u}$.

If $next(\eta, \boldsymbol{u}) = (\eta', \boldsymbol{u}')$, η' is called *the next pattern of* η, written $Next(\eta)$.

According to Example 1, we visualize a pattern η as a circle. Applications of $Next$ can be regarded as moving 0^1 along the circle, by performing merge-operations and split-operations alternatively. After enough number of applications of $Next$, 0^1 will return to the original now-position after moving through the entire circle. That is, for each pattern η, $Next^m(\eta) = \eta$, where $m = 2n$ (resp. $m = 2(n+1)$) if η is a merge-pattern (resp. split-pattern). The sequence $\eta, Next(\eta), \cdots, Next^m(\eta)$ is called a *pattern ring*.

Notice that $next^m(\eta, \boldsymbol{u}) = (\eta, \boldsymbol{u}+1)$ for each $\boldsymbol{u}$. On a pattern ring, merge-patterns and split-patterns appear alternately.

Beside clock progresses, clock resets are the other form of clock behaviors.

Example 2. Take $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ as in Example 1. Consider $\boldsymbol{v}'_1 = (1.6_{0^1}, 0_{1^1}, 3.1_{2^1})$ that is the result of resetting x_1 in $\boldsymbol{v}_1$. The pattern of $(\boldsymbol{v}_0, \boldsymbol{v}'_1)$ is $\{0^0\}, \{2^0\}, \{0^1, 1^1\}, \{1^0, 2^1\}$, which is the result of moving 1^1 (the index of x_1 in $\boldsymbol{v}_1$) into the now-position $\{0^1\}$ of the pattern η of $(\boldsymbol{v}_0, \boldsymbol{v}_1)$ (see Example 1). ∎

Let $r \subseteq K^+$ be (a set of) *clock resets*. Denote $\boldsymbol{v} \downarrow_r$ to be the result of resetting each clock $x_i \in r$ (i.e., $i \in r$). That is, for each $i \in K$, if $i \in r$, then $(\boldsymbol{v} \downarrow_r)(i) = 0$ else $(\boldsymbol{v} \downarrow_r)(i) = \boldsymbol{v}(i)$. Functions $reset_r : \Phi \times (\mathbf{N}^+)^{k+1} \to \Phi \times (\mathbf{N}^+)^{k+1}$ for $r \subseteq K^+$ describe how a pattern changes after clock resets. Given any discrete valuation $\boldsymbol{u}$ and any pattern $\eta = p_0, \cdots, p_n$ with the now-position being p_i for some i, $reset_r(\eta, \boldsymbol{u})$ is defined to be $(\eta', \boldsymbol{u}')$ such that,

- η' is $p_0 - r^1, \cdots, p_{i-1} - r^1, p_i \cup r^1, p_{i+1} - r^1, \cdots, p_n - r^1$, where $r^1 = \{j^1 : j \in r\} \subseteq K^1$. Therefore, η' is the result of bringing every index in r^1 into the now-position. Notice that some of positions $p_m - r^1$ may be empty after moving indices in r^1 out of p_m, for $m \neq i$. In this case, these positions are removed from η' (to guarantee that η' is well defined.),
- for each $j \in K$, if $j \in r$, then $\boldsymbol{u}'(j) = 0$ else $\boldsymbol{u}'(j) = \boldsymbol{u}(j)$.

If $reset_r(\eta, \boldsymbol{u}) = (\eta', \boldsymbol{u}')$, η' is written as $Reset_r(\eta)$.

Given an initialized pair $(\boldsymbol{v}_0, \boldsymbol{v})$ and $0 < \delta \in \mathbf{D}^+$. Assume the patterns of $(\boldsymbol{v}_0, \boldsymbol{v})$ and $(\boldsymbol{v}_0, \boldsymbol{v} + \delta)$ are η and η', respectively. We say $\boldsymbol{v}$ *has no pattern change for* δ if, for all $0 \leq \delta' \leq \delta$, $(\boldsymbol{v}_0, \boldsymbol{v} + \delta')$ has the same pattern. We say $\boldsymbol{v}$ *has one pattern change for* δ if $Next(\eta) = \eta'$ (recall $Next(\eta) \neq \eta$) and, for all $0 < \delta' < \delta$, $(\boldsymbol{v}_0, \boldsymbol{v} + \delta')$ has pattern η, or, for all $0 < \delta' < \delta$, $(\boldsymbol{v}_0, \boldsymbol{v} + \delta')$ has pattern η'. We say $\boldsymbol{v}$ *has n pattern changes for* δ with $n \geq 1$, if there are positive $\delta_1, \cdots, \delta_n$ in $\mathbf{D}^+$ with $\Sigma_{1 \leq i \leq n} \delta_i = \delta$ such that $\boldsymbol{v} + \Sigma_{1 \leq i \leq j} \delta_i$ has one pattern change for δ_{j+1}, for each $j = 0, \cdots, n-1$. The following lemma states that both $next$ and $reset_r$ are "correct".

Lemma 2. *For all patterns η and η', for all $r \subseteq K^+$, and for all discrete valuations $\boldsymbol{u}$ and $\boldsymbol{u}'$, the following (1) and (2) hold:*

(1). (correctness of $next$) $next(\eta, \boldsymbol{u}) = (\eta', \boldsymbol{u}')$ iff there exist an initialized pair $(\boldsymbol{v}_0, \boldsymbol{v})$ and $0 < \delta \in \mathbf{D}^+$ such that

(1.1). η is the pattern of $(\boldsymbol{v}_0, \boldsymbol{v})$ and η' is the pattern of $(\boldsymbol{v}_0, \boldsymbol{v} + \delta)$,

(1.2). $\boldsymbol{u} = \lceil \boldsymbol{v} \rceil$ and $\boldsymbol{u}' = \lceil \boldsymbol{v} + \delta \rceil$,

(1.3). $\boldsymbol{v}$ has one pattern change for δ. In particular, if η is a merge-pattern, then for all $0 \leq \delta' < \delta$, η is the pattern of $(\boldsymbol{v}_0, \boldsymbol{v} + \delta')$. If, however, η is a split-pattern, then for all $0 < \delta' \leq \delta$, η' is the pattern of $(\boldsymbol{v}_0, \boldsymbol{v} + \delta')$,

(2). (correctness of $reset_r$) $reset_r(\eta, \boldsymbol{u}) = (\eta', \boldsymbol{u}')$ iff there exist an initialized pair $(\boldsymbol{v}_0, \boldsymbol{v})$ such that

(2.1). η is the pattern of $(\boldsymbol{v}_0, \boldsymbol{v})$ and η' is the pattern of $(\boldsymbol{v}_0, \boldsymbol{v} \downarrow_r)$,

(2.2). $\boldsymbol{u} = \lceil \boldsymbol{v} \rceil$ and $\boldsymbol{u}' = \lceil \boldsymbol{v} \downarrow_r \rceil$.

(3). For any fixed initialized pair $(\boldsymbol{v}_0, \boldsymbol{v})$ and fixed $0 < \delta \in \mathbf{D}^+$, there is a unique finite number n such that $\boldsymbol{v}$ has n pattern changes for δ. In particular, when $\delta = 1$, the number n is exactly the length of the pattern ring starting from the pattern of $(\boldsymbol{v}_0, \boldsymbol{v})$.

(4). The number n in (3) can be uniformly bounded for each δ. That is, for any fixed $\delta \in \mathbf{D}^+$, there is a finite number m such that, for any initialized pair $(\boldsymbol{v}_0, \boldsymbol{v})$, $\boldsymbol{v}$ has at most m pattern changes for δ.

(5). For any fixed initialized pair $(\boldsymbol{v}_0, \boldsymbol{v})$, the pattern of $(\boldsymbol{v}_0, \boldsymbol{v})$ is a merge-pattern iff there is a $0 < \delta \in \mathbf{D}^+$ such that $\boldsymbol{v}$ has no pattern change for δ.

4 Clock Constraints and Patterns

An *atomic clock constraint* (over clocks $x_1, \cdots, x_k$, excluding x_0) is a formula in the form of $x_i - x_j \# d$ or $x_i \# d$ where $0 \le d \in \mathbf{N}^+$ and $\#$ stands for $<, >, \le, \ge, =$. A *clock constraint* c is a Boolean combination of atomic clock constraints. Denote $\mathcal{C}$ to be the set of all clock constraint (over clocks $x_1, \cdots, x_k$). We say $\boldsymbol{v} \in c$ if clock valuation $\boldsymbol{v}$ (for $x_0, \cdots, x_k$) satisfies clock constraint c.

Any clock constraint c can be written as a Boolean combination $I(c)$ of clock constraints over discrete clocks $\lceil x_1 \rceil, \cdots, \lceil x_k \rceil$ and fractional orderings $\lfloor x_i \rfloor \# \lfloor x_j \rfloor$ and $\lfloor x_i \rfloor \# 0$. Therefore, testing $\boldsymbol{v} \in c$ is equivalent to testing $\lceil \boldsymbol{v} \rceil$ and the fractional orderings on $\lfloor \boldsymbol{v} \rfloor$ satisfying $I(c)$.

Assume $\boldsymbol{v}$ has pattern η. We use c^η to denote the result of replacing fractional orderings in $I(c)$ by the truth values given by η. c^η is a clock constraint (over discrete clocks). The following lemma can be observed.

Lemma 3. *(1). For any initialized pair $(\boldsymbol{v}_0, \boldsymbol{v})$, any pattern $\eta \in \Phi$, if $(\boldsymbol{v}_0, \boldsymbol{v})$ has pattern η, then, for any clock constraint $c \in \mathcal{C}$, $\boldsymbol{v} \in c$ iff $\lceil \boldsymbol{v} \rceil \in c^\eta$. (2). For any initialized pair $(\boldsymbol{v}_0, \boldsymbol{v})$ and any $0 < \delta \in \mathbf{D}^+$, if $\boldsymbol{v}$ has at most one pattern change for δ, then, for any clock constraint $c \in \mathcal{C}$, $\forall 0 \le \delta' \le \delta(\boldsymbol{v} + \delta' \in c)$ iff $\boldsymbol{v} \in c$ and $\boldsymbol{v} + \delta \in c$. (3). For any initialized pairs $(\boldsymbol{v}_0^1, \boldsymbol{v}_1)$ and $(\boldsymbol{v}_0^2, \boldsymbol{v}_2)$, if $(\boldsymbol{v}_0^1, \boldsymbol{v}_1) \approx (\boldsymbol{v}_0^2, \boldsymbol{v}_2)$, then, for any $c \in \mathcal{C}$, $\boldsymbol{v}_1 \in c$ iff $\boldsymbol{v}_2 \in c$.*

Consider two initialized pairs $(\boldsymbol{v}_0^1, \boldsymbol{v}_1)$ and $(\boldsymbol{v}_0^2, \boldsymbol{v}_2)$ such that $(\boldsymbol{v}_0^1, \boldsymbol{v}_1) \approx (\boldsymbol{v}_0^2, \boldsymbol{v}_2)$. ¿From Lemma 3(3), any test $c \in \mathcal{C}$ will not tell the difference between $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Assume $\boldsymbol{v}_1$ can be reached from a valuation $\boldsymbol{v}^1$ via a clock progress by an amount of δ_1, i.e., $\boldsymbol{v}^1 + \delta_1 = \boldsymbol{v}_1$. We would like to know whether $\boldsymbol{v}_2$ can be reached from some valuation $\boldsymbol{v}^2$ also via a clock progress but probably by a slightly different amount of δ_2 such that $(\boldsymbol{v}_0^1, \boldsymbol{v}^1)$ and $(\boldsymbol{v}_0^2, \boldsymbol{v}^2)$ are still equivalent($\approx$). We also expect that for any test c, if during the progress of $\boldsymbol{v}^1$, c is consistently satisfied, then so is c for the progress of $\boldsymbol{v}^2$. The following lemma concludes that these, as well as the parallel case for clock resets, can be done. This result can be used later to show that if $\boldsymbol{v}_1$ is reached from $\boldsymbol{v}_0^1$ by a sequence of transitions that repeatedly perform clock progresses and clock resets, then $\boldsymbol{v}_2$ can be also reached from $\boldsymbol{v}_0^2$ via a very similar sequence such that no test c can tell the difference on the two sequences.

Lemma 4. *For any initialized pairs $(\boldsymbol{v}_0^1, \boldsymbol{v}_1)$ and $(\boldsymbol{v}_0^2, \boldsymbol{v}_2)$ with $(\boldsymbol{v}_0^1, \boldsymbol{v}_1) \approx (\boldsymbol{v}_0^2, \boldsymbol{v}_2)$,*

(1). for any $0 \le \delta_1 \in \mathbf{D}^+$, for any clock valuation $\boldsymbol{v}^1$, if $\boldsymbol{v}^1 + \delta_1 = \boldsymbol{v}_1$, then there exist $0 \le \delta_2 \in \mathbf{D}^+$ and clock valuation $\boldsymbol{v}^2$ such that (1.1). $\boldsymbol{v}^2 + \delta_2 = \boldsymbol{v}_2$ and

$(\boldsymbol{v}_0^1, \boldsymbol{v}^1) \approx (\boldsymbol{v}_0^2, \boldsymbol{v}^2)$, (1.2). $\boldsymbol{v}^1$ is initial iff $\boldsymbol{v}^2$ is initial, $\boldsymbol{v}^1 = \boldsymbol{v}_0^1$ iff $\boldsymbol{v}^2 = \boldsymbol{v}_0^2$, and for any $c \in \mathcal{C}$, $\boldsymbol{v}^1 \in c$ (resp. $\boldsymbol{v}_1 \in c$) iff $\boldsymbol{v}^2 \in c$ (resp. $\boldsymbol{v}_2 \in c$), (1.3). for any clock constraint $c \in \mathcal{C}$, $\forall 0 \le \delta' \le \delta_1 (\boldsymbol{v}^1 + \delta \in c)$ iff $\forall 0 \le \delta' \le \delta_2 (\boldsymbol{v}^2 + \delta \in c)$.

(2). for any $r \subseteq K^+$, for any clock valuation $\boldsymbol{v}^1$, if $\boldsymbol{v}^1 \downarrow_r = \boldsymbol{v}_1$, then there exists a valuation $\boldsymbol{v}^2$ such that (2.1). $\boldsymbol{v}^2 \downarrow_r = \boldsymbol{v}_2$ and $(\boldsymbol{v}_0^1, \boldsymbol{v}^1) \approx (\boldsymbol{v}_0^2, \boldsymbol{v}^2)$, (2.2). same as (1.2).

5 Pushdown Timed Automata

A *pushdown timed automaton* (PTA) $\mathcal{A}$ is a tuple $\langle S, \{x_1, \cdots, x_k\}, Inv, R, \Gamma, PD \rangle$, where S is a finite set of *states*, $x_1, \cdots, x_k$ are (dense) clocks. $Inv : S \to \mathcal{C}$ assigns a clock constraint over clocks $x_1, \cdots, x_k$, called an *invariant*, to each state. $R : S \times S \to \mathcal{C} \times 2^{\{x_1, \cdots, x_k\}}$ assigns a clock constraint over clocks $x_1, \cdots, x_k$, called a *reset condition*, and a subset of clocks, called clock resets, to a (directed) edge in $S \times S$. Γ is the *stack alphabet*. $PD : S \times S \to \Gamma \times \Gamma^*$ assigns a pair (a, γ) with $a \in \Gamma$ and $\gamma \in \Gamma^*$, called a *stack operation*, to each edge in $S \times S$. A stack operation (a, γ) replaces the top symbol a of the stack with a string (possibly empty) in Γ^*. A *timed automaton* is a PTA without the pushdown stack.

The semantics of $\mathcal{A}$ is defined as follows. A *configuration* is a triple $(s, \boldsymbol{v}, w)$ of a state s, a clock valuation $\boldsymbol{v}$ on $x_0, \cdots, x_k$ (where x_0 is the auxiliary clock), and a stack word $w \in \Gamma^*$. $(s_1, \boldsymbol{v}_1, w_1) \to_{\mathcal{A}} (s_2, \boldsymbol{v}_2, w_2)$ denotes a *one-step transition* of $\mathcal{A}$ if one of the following conditions is satisfied:

- (*a progress transition*) $s_1 = s_2$, $w_1 = w_2$, and $\exists 0 < \delta \in \mathbf{D}^+$, $\boldsymbol{v}_2 = \boldsymbol{v}_1 + \delta$ and for all δ' satisfying $0 \le \delta' \le \delta$, $\boldsymbol{v}_1 + \delta' \in Inv(s_1)$. That is, a progress transition makes all the clocks synchronously progress by amount $\delta > 0$, during which the invariant is consistently satisfied, while the state and the stack content remain unchanged.
- (*a reset transition*) $\boldsymbol{v}_1 \in Inv(s_1) \wedge c$, $\boldsymbol{v}_1 \downarrow_r = \boldsymbol{v}_2 \in Inv(s_2)$, and $w_1 = aw, w_2 = \gamma w$ for some $w \in \Gamma^*$, where $R(s_1, s_2) = (c, r)$ for some clock constraint c and clock resets r, and $PD(s_1, s_2) = (a, \gamma)$ for some stack symbol $a \in \Gamma$ and string $\gamma \in \Gamma^*$. That is, a reset transition, by moving from state s_1 to state s_2, resets every clock in r to 0 and keeps all the other clocks unchanged. The stack content is modified according to the stack operation (a, γ) given on edge (s_1, s_2). Clock values before the transition satisfy the invariant $Inv(s_1)$ and the reset condition c; clock values after the transition satisfy the invariant $Inv(s_2)$.

We write $\to_{\mathcal{A}}^*$ to be the transitive closure of $\to_{\mathcal{A}}$. Given two valuations $\boldsymbol{v}_0^1$ and $\boldsymbol{v}_1$, two states s_0 and s_1, and two stack words w_0 and w_1, assume the auxiliary clock x_0 starts from 0, i.e., $\boldsymbol{v}_0^1$ is initial. The following result is surprising. It states that, for **any** initialized pair $(\boldsymbol{v}_0^2, \boldsymbol{v}_2)$ with $(\boldsymbol{v}_0^1, \boldsymbol{v}_1) \approx (\boldsymbol{v}_0^2, \boldsymbol{v}_2)$, $(s_0, \boldsymbol{v}_0^1, w_0) \to_{\mathcal{A}}^* (s_1, \boldsymbol{v}_1, w_1)$ if and only if $(s_0, \boldsymbol{v}_0^2, w_0) \to_{\mathcal{A}}^* (s_1, \boldsymbol{v}_2, w_1)$. This result implies that, from the definition of $\approx$, for any fixed s_0, s_1, w_0 and w_1, the pattern of $(\lfloor \boldsymbol{v}_0^1 \rfloor, \lfloor \boldsymbol{v}_1 \rfloor)$ (instead of the actual values of $\lfloor \boldsymbol{v}_0^1 \rfloor$ and $\lfloor \boldsymbol{v}_1 \rfloor$), the integral values $\lceil \boldsymbol{v}_0^1 \rceil$, and the integral values $\lceil \boldsymbol{v}_1 \rceil$ are sufficient to determine whether $(s_0, \boldsymbol{v}_0^1, w_0)$ can reach $(s_1, \boldsymbol{v}_1, w_1)$ in $\mathcal{A}$. The proof is an induction on the length of $(s_0, \boldsymbol{v}_0^1, w_0) \to_{\mathcal{A}}^* (s_1, \boldsymbol{v}_1, w_1)$ using Lemma 4 and Lemma 3.

Lemma 5. *Let $\mathcal{A}$ be a PTA. For any states s_0 and s_1, any two initial clock valuations $\boldsymbol{v}_0^1$ and $\boldsymbol{v}_0^2$, any two clock valuations $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, and any two stack words w_0*

and w_1*, if* $(\boldsymbol{v}_0^1, \boldsymbol{v}_1) \approx (\boldsymbol{v}_0^2, \boldsymbol{v}_2)$*, then,* $(s_0, \boldsymbol{v}_0^1, w_0) \rightarrow_{\mathcal{A}}^* (s_1, \boldsymbol{v}_1, w_1)$ *iff* $(s_0, \boldsymbol{v}_0^2, w_0) \rightarrow_{\mathcal{A}}^* (s_1, \boldsymbol{v}_2, w_1)$.

Example 3. It is the time to show an example to convince the reader that Lemma 5 indeed works. Consider a timed automaton $\mathcal{A}$ shown in Figure 1. Let $\boldsymbol{v}_0^1 = (0, 4.98, 2.52)$,

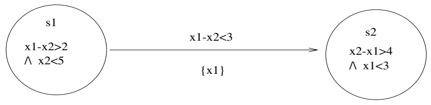

Fig. 1. An Example Timed Automaton $\mathcal{A}$.

$\boldsymbol{v}_3^1 = (5.36, 2.89, 7.88)$. $(s_1, \boldsymbol{v}_0^1) \rightarrow_{\mathcal{A}}^* (s_2, \boldsymbol{v}_3^1)$ is witnessed by: $(s_1, \boldsymbol{v}_0^1) \rightarrow_{\mathcal{A}}$ (progress by 2.47 at s_1) $(s_1, \boldsymbol{v}_1^1) \rightarrow_{\mathcal{A}}$ (reset x_1 and transit to s_2) $(s_2, \boldsymbol{v}_2^1) \rightarrow_{\mathcal{A}}$ (progress by 2.89 at s_2) $(s_2, \boldsymbol{v}_3^1)$. Take a new pair $\boldsymbol{v}_0^2 = (0, 4.89, 2.11)$, $\boldsymbol{v}_3^2 = (5.28, 2.77, 7.39)$. It is easy to check $(\boldsymbol{v}_0^1, \boldsymbol{v}_3^1) \approx (\boldsymbol{v}_0^2, \boldsymbol{v}_3^2)$. ¿From Lemma 5, $(s_1, \boldsymbol{v}_0^2) \rightarrow_{\mathcal{A}}^* (s_2, \boldsymbol{v}_3^2)$. Indeed, this is witnessed by $(s_1, \boldsymbol{v}_0^2) \rightarrow_{\mathcal{A}}$ (progress by 2.51 at s_1) $(s_1, \boldsymbol{v}_1^2) \rightarrow_{\mathcal{A}}$ (reset x_1 and transit to s_2) $(s_2, \boldsymbol{v}_2^2) \rightarrow_{\mathcal{A}}$ (progress by 2.77 at s_2) $(s_2, \boldsymbol{v}_3^2)$. These two witnesses differ slightly (2.47 and 2.89, vs. 2.51 and 2.77). We choose 2.77 and 2.51 by looking at the first witness backwardly. That is, $\boldsymbol{v}_2^2$ is picked such that $(\boldsymbol{v}_0^2, \boldsymbol{v}_2^2) \approx (\boldsymbol{v}_0^1, \boldsymbol{v}_2^1)$. Then, $\boldsymbol{v}_1^2$ is picked such that $(\boldsymbol{v}_0^2, \boldsymbol{v}_1^2) \approx (\boldsymbol{v}_0^1, \boldsymbol{v}_1^1)$. The existence of $\boldsymbol{v}_2^2$ and $\boldsymbol{v}_1^2$ is guaranteed by Lemma 4. Finally, according to Lemma 4 again, $\boldsymbol{v}_1^2$ is able to go back to $\boldsymbol{v}_0^2$. This is because $\boldsymbol{v}_1^1$ goes back to $\boldsymbol{v}_0^1$ through a one-step transition and $\boldsymbol{v}_0^1$ is initial. ∎

Now, we express $\rightarrow_{\mathcal{A}}^*$ in a form that treating the integral parts and the fractional parts of clock values separately. Given a pattern $\eta \in \Phi$, for any discrete valuations $\boldsymbol{u}_0$ and $\boldsymbol{u}_1$, and any stack words w_0 and w_1, define $(s_0, \boldsymbol{u}_0, w_0) \rightarrow_{\mathcal{A},\eta}^* (s_1, \boldsymbol{u}_1, w_1)$ to be $\exists \boldsymbol{v}_0 \exists \boldsymbol{v}_1 (\boldsymbol{v}_0(0) = 0 \wedge \lceil \boldsymbol{v}_0 \rceil = \boldsymbol{u}_0 \wedge \lceil \boldsymbol{v}_1 \rceil = \boldsymbol{u}_1 \wedge (\boldsymbol{v}_0, \boldsymbol{v}_1) \in \eta \wedge (s_0, \boldsymbol{v}_0, w_0) \rightarrow_{\mathcal{A}}^* (s_1, \boldsymbol{v}_1, w_1))$.

Lemma 6. *Let* $\mathcal{A}$ *be a PTA. For any states* s_0 *and* s_1*, any initialized pair* $(\boldsymbol{v}_0, \boldsymbol{v}_1)$*, and any stack words* w_0 *and* w_1*,* $(s_0, \boldsymbol{v}_0, w_0) \rightarrow_{\mathcal{A}}^* (s_1, \boldsymbol{v}_1, w_1)$ *iff* $\vee_{\eta \in \Phi} (\boldsymbol{v}_0(0) = 0 \wedge (\lfloor \boldsymbol{v}_0 \rfloor, \lfloor \boldsymbol{v}_1 \rfloor) \in \eta \wedge (s_0, \lceil \boldsymbol{v}_0 \rceil, w_0) \rightarrow_{\mathcal{A},\eta}^* (s_1, \lceil \boldsymbol{v}_1 \rceil, w_1))$.

Once we give a characterization of $\rightarrow_{\mathcal{A},\eta}^*$, Lemma 6 immediately gives a characterization for $\rightarrow_{\mathcal{A}}^*$. A decidable characterization of $\rightarrow_{\mathcal{A},\eta}^*$ is shown in the next section.

6 The Pattern Graph of a Timed Pushdown Automaton

Let $\mathcal{A} = \langle S, \{x_1, \cdots, x_k\}, Inv, R, \Gamma, PD \rangle$ be a PTA specified in the previous section. The *pattern graph* G of $\mathcal{A}$ is a tuple $\langle S \times \Phi, \{y_0, \cdots, y_k\}, E, \Gamma \rangle$ where S is the states in $\mathcal{A}$, Φ is the set of all patterns. A *node* is an element in $S \times \Phi$. *Discrete clocks* $y_0, \cdots, y_k$ are the integral parts of the clocks $x_0, \cdots, x_k$ in $\mathcal{A}$. E is a finite set of (directed) *edges* that connect pairs of nodes. An edge can be a *progress* edge, a *stay* edge, or a *reset* edge.

A progress edge corresponds to progress transitions in $\mathcal{A}$ that cause one pattern change. A stay edge corresponds to progress transitions in $\mathcal{A}$ that cause no pattern change. Since a progress transition can cause no pattern change only from a merge-pattern, a stay edge connects a merge-pattern to itself. A reset edge corresponds to a reset transition in $\mathcal{A}$. Formally, a progress edge $e_{s,\eta,\eta'}$ that connects node (s,η) to node (s,η') is in the form of $\langle(s,\eta),c,(s,\eta')\rangle$ such that $c = Inv(s)$, $\eta' = Next(\eta)$ (thus $\eta \neq \eta'$). A stay edge $e_{s,\eta,\eta}$, with η being a merge-pattern, that connects node (s,η) to itself is in the form of $\langle(s,\eta),c,(s,\eta)\rangle$ such that $c = Inv(s)$. A reset edge $e_{s,s',r,(a,\gamma)}$ that connects node (s,η) to node (s',η') is in the form of $\langle(s,\eta),c,r,a,\gamma,(s',\eta')\rangle$ where $R(s,s') = (c,r)$ and $PD(s,s') = (a,\gamma)$. E is the set of all progress edges, stay edges, and reset edges wrt $\mathcal{A}$. Obviously, E is finite.

A configuration of G is a tuple $(s,\eta,\boldsymbol{u},w)$ of state $s \in S$, pattern $\eta \in \Phi$, discrete valuation $\boldsymbol{u} \in (\mathbf{N}^+)^{k+1}$ and stack word $w \in \Gamma^*$. $(s,\eta,\boldsymbol{u},w) \to^e (s',\eta',\boldsymbol{u}',w')$ denotes a *one-step transition* through edge e of G if the following conditions are satisfied:

- if e is a progress edge, then e takes the form $\langle(s,\eta),c,(s,\eta')\rangle$ and $s' = s$, $\boldsymbol{u} \in c^\eta$, $\boldsymbol{u}' \in c^{\eta'}$, $next(\eta,\boldsymbol{u}) = (\eta',\boldsymbol{u}')$ and $w = w'$. Here c^η and $c^{\eta'}$ are called the *pre-* and the *post- (progress) tests* on edge e, respectively.
- if e is a stay edge, then e takes the form $\langle(s,\eta),c,(s,\eta)\rangle$ and $s = s'$, $\boldsymbol{u} \in c^\eta$, $\boldsymbol{u} = \boldsymbol{u}'$, $\eta = \eta'$ and $w = w'$. Here c^η is called the *pre-* and the *post- (stay) tests* on edge e.
- if e is a reset edge, then e takes the form $\langle(s,\eta),c,r,a,\gamma,(s',\eta')\rangle$ and $\boldsymbol{u} \in (c \wedge Inv(s))^\eta$, $\boldsymbol{u}' \in Inv(s')^{\eta'}$, $reset_r(\eta,\boldsymbol{u}) = (\eta',\boldsymbol{u}')$ and $w = aw'', w' = \gamma w''$ for some $w'' \in \Gamma^*$ (i.e., w changes to w' according to the stack operation). Here $(c \wedge Inv(s))^\eta$ and $Inv(s')^{\eta'}$ are called the *pre-* and the *post- (reset) tests* on edge e, respectively.

We write $(s,\eta,\boldsymbol{u},w) \to_G (s',\eta',\boldsymbol{u}',w')$ if $(s,\eta,\boldsymbol{u},w) \to^e (s',\eta',\boldsymbol{u}',w')$ for some e. The binary reachability $\to_G^*$ of G is the transitive closure of $\to_G$.

The pattern graph G simulates $\mathcal{A}$ in a way that the integral parts of the dense clocks are kept but the fractional parts are abstracted as a pattern. Edges in G indicates how the pattern and the discrete clocks change when a clock progress or a clock reset occur in $\mathcal{A}$. However, a progress transition in $\mathcal{A}$ could cause more than one pattern change. In this case, this big progress transition is treated as a sequence of small progress transitions such that each causes one pattern change (and therefore, each small progress transition in $\mathcal{A}$ can be simulated by a progress edge in G). We first show that the binary reachability $\to_G^*$ of G is NPCM. Observe that discrete clocks $y_0,\cdots,y_k$ are the integral values of dense clocks $x_0,\cdots,x_k$. Even though the dense clocks progress synchronously, the discrete clocks may not be synchronous (i.e., that one discrete clock is incremented by 1 does not necessarily cause **all** the other discrete clocks incremented by the same amount.). The proof has two parts. In the first part of the proof, a technique is used to translate $y_0,\cdots,y_k$ into another array of discrete clocks that are synchronous. In the second part of the proof, G can be treated as a discrete PTA [15] by replacing $y_0,\cdots,y_k$ with the synchronous discrete clocks. Therefore, Lemma 7 follows by the fact [15] that the binary reachability of discrete PTA is NPCM.

Lemma 7. *For any PTA $\mathcal{A}$, the binary reachability $\to_G^*$ of the pattern graph G of $\mathcal{A}$ is NPCM. In particular, if $\mathcal{A}$ is a timed automaton, then the binary reachability $\to_G^*$ is Presburger.*

The following lemma states that G faithfully simulates $\mathcal{A}$ when the fractional parts of dense clocks are abstracted away by a pattern. The if-part of the lemma uses Lemma 2. The only-if-part of the lemma is based upon the argument that a one-step transition of $\mathcal{A}$, when the pattern abstraction is used, can be simulated by a sequence of transitions of G.

Lemma 8. *Let $\mathcal{A}$ be a PTA with pattern graph G. For any $s_0, s_1 \in S$, $\eta \in \Phi$, $w_0, w_1 \in \Gamma^*$, and $(\boldsymbol{u}_0, \boldsymbol{u}_1)$ with $\boldsymbol{u}_0(0) = 0$, $(s_0, \boldsymbol{u}_0, w_0) \to_{\mathcal{A},\eta}^* (s_1, \boldsymbol{u}_1, w_1)$ iff $(s_0, init(\eta), \boldsymbol{u}_0, w_0) \to_G^* (s_1, \eta, \boldsymbol{u}_1, w_1)$.*

Now, we conclude this section by claiming that $\to_{\mathcal{A},\eta}^*$ is NPCM by combining Lemma 7 and Lemma 8.

Lemma 9. *For any PTA $\mathcal{A}$ and any fixed pattern $\eta \in \Phi$, $\to_{\mathcal{A},\eta}^*$ is NPCM. In particular, if $\mathcal{A}$ is a timed automaton, then $\to_{\mathcal{A},\eta}^*$ is Presburger.*

7 A Decidable Binary Reachability Characterization and Automatic Verification

Recall that PTA $\mathcal{A}$ actually has clocks $x_1, \cdots, x_k$. x_0 is the auxiliary clock. The *binary reachability* $\leadsto_{\mathcal{A}}^{*\mathbf{B}}$ of $\mathcal{A}$ is the set of tuples $\langle s, v_1, \cdots, v_k, w, s', v_1', \cdots, v_k', w' \rangle$ such that there exist $v_0 = 0, v_0' \in \mathbf{D}^+$ satisfying $(s, v_0, \cdots, v_k, w) \leadsto_{\mathcal{A}}^* (s', v_0', \cdots, v_k', w')$. The main theorem of this paper gives a decidable characterization for the binary reachability as follows. The proof uses Lemma 6 and Lemma 9.

Theorem 1. *The binary reachability $\leadsto_{\mathcal{A}}^{*\mathbf{B}}$ of a PTA $\mathcal{A}$ is $(\mathbf{D} + \mathbf{NPCM})$-definable. In particular, if $\mathcal{A}$ is a timed automaton, then the binary reachability $\leadsto_{\mathcal{A}}^{*\mathbf{B}}$ can be expressed in the additive theory of reals (or rationals) and integers.*

The importance of the above characterization for $\leadsto_{\mathcal{A}}^{*\mathbf{B}}$ is that, from Lemma 1, the emptiness of $(\mathbf{D} + \mathbf{NPCM})$-definable predicates is decidable. ¿From Theorem 1 and Lemma 1 (3)(4), we have,

Theorem 2. *The emptiness of $l \cap \leadsto_{\mathcal{A}}^{*\mathbf{B}}$ with respect to a PTA $\mathcal{A}$ for any mixed linear relation l is decidable.*

The emptiness of $l \cap \leadsto_{\mathcal{A}}^{*\mathbf{B}}$ is called a *mixed linear property* of $\mathcal{A}$. Many interesting safety properties (or their negations) for PTAs can be expressed as a mixed linear property. For instance, consider the following property of a PTA $\mathcal{A}$:

"for any two configurations α and β with $\alpha \leadsto_{\mathcal{A}}^{*\mathbf{B}} \beta$, if the difference between β_{x_3} (the value of clock x_3 in β) and $\alpha_{x_1} + \alpha_{x_2}$ (the sum of clocks x_1 and x_2 in α) is greater than the difference between $\#_a(\alpha_{\mathbf{w}})$ (the number of symbol a appearing in the stack word in α) and $\#_b(\beta_{\mathbf{w}})$ (the number of symbol b appearing in the stack word in β), then $\#_a(\alpha_{\mathbf{w}}) - 2\#_b(\beta_{\mathbf{w}})$ is greater than 5."

The negation of this property can be expressed in the form required by Theorem 2. Thus, this property can be automatically verified. Notice that this property can not be verified by using results in [8] and (even when clocks are ignored) in [7,18]. When $\mathcal{A}$ is a timed automaton, by Theorem 1, the binary reachability $\leadsto^{*\mathbf{B}}_{\mathcal{A}}$ can be expressed in the additive theory of reals (or rationals) and integers. Notice that this characterization is essentially equivalent to the one given by Comon and Jurski [13] in which $\leadsto^{*\mathbf{B}}_{\mathcal{A}}$ can be expressed in the additive theory of reals augmented with a predicate telling whether a term is an integer. Because the additive theory of reals and integers is decidable (see [10] for a procedure), we have,

Theorem 3. *The truth value for any closed formula expressible in the (first-order) additive theory of reals (or rationals) augmented with a predicate* $\leadsto^{*\mathbf{B}}_{\mathcal{A}}$ *for a timed automaton* $\mathcal{A}$ *is decidable. (also shown in [13])*

8 Conclusions

In this paper, we consider PTAs that are timed automata augmented with a pushdown stack. By introducing the concept of a clock pattern and using an automata-theoretic approach, we give a decidable characterization of the binary reachability of a PTA. The results can be used to verify a class of safety properties containing linear relations over both dense variables and unbounded discrete variables.

The results in this paper can be extended to PTAs augmented with reversal-bounded counters. A future research issue is to investigate whether the liveness results in [17] and the approximation techniques in [16] can be extended to dense clocks. Another issue is on the complexity analysis of the decision procedure presented in this paper. However, the complexity for the emptiness problem of NPCMs is still unknown, though it is believed that it can be derived along Gurari and Ibarra [19]. The results in this paper can be used to implement a model-checker for a subset of the real-time specification language ASTRAL [11] as well as for a class of real-time programming language with procedure calls (such as a timed version of Boolean programs [9]).

Acknowledgment

The author would like to thank H. Comon and O. H. Ibarra for discussions on the topic of dense timed pushdown automata during CAV'00 in Chicago, B. Boigelot, P. San Pietro and J. Su for recent discussions on [10], and F. Sheldon for reading an earlier draft of this paper. Thanks also go to anonymous reviewers for many useful suggestions.

References

1. R. Alur, *"Timed automata"*, *CAV'99*, LNCS 1633, pp. 8-22
2. R. Alur, C. Courcoibetis, and D. Dill, *"Model-checking in dense real time,"* *Information and Computation*, **104** (1993) 2-34
3. R. Alur and D. Dill, *"A theory of timed automata,"* *Theoretical Computer Science*, **126** (1994) 183-236

4. R. Alur, T. Feder, and T. A. Henzinger, *"The benefits of relaxing punctuality," J. ACM*, **43** (1996) 116-146
5. R. Alur, T. A. Henzinger, *"Real-time logics: complexity and expressiveness," Information and Computation*, **104** (1993) 35-77
6. R. Alur, T. A. Henzinger, *"A really temporal logic," J. ACM*, **41** (1994) 181-204
7. A. Bouajjani, J. Esparza, and O. Maler, *"Reachability Analysis of Pushdown Automata: Application to Model-Checking,", CONCUR'97*, LNCS 1243, pp. 135-150
8. A. Bouajjani, R. Echahed, and R. Robbana, *"On the Automatic Verification of Systems with Continuous Variables and Unbounded Discrete Data Structures," Hybrid System II*, LNCS 999, 1995, pp. 64-85
9. T. Ball and S. K. Rajamani, *"Bebop: A Symbolic Model-checker for Boolean Programs," Spin Workshop'00*, LNCS 1885, pp. 113-130.
10. B. Boigelot, S. Rassart and P. Wolper, *"On the expressiveness of real and integer arithmetic automata," ICALP'98*, LNCS 1443, pp. 152-163
11. A. Coen-Porisini, C. Ghezzi and R. Kemmerer, *"Specification of real-time systems using ASTRAL," IEEE Transactions on Software Engineering*, **23** (1997) 572-598
12. H. Comon and Y. Jurski, *"Multiple counters automata, safety analysis and Presburger arithmetic," CAV'98*, LNCS 1427, pp. 268-279.
13. H. Comon and Y. Jurski, *"Timed Automata and the Theory of Real Numbers," CONCUR'99*, LNCS 1664, pp. 242-257
14. Z. Dang, `http://www.eecs.wsu.edu/~zdang`, the full version of this paper
15. Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su, *"Binary reachability analysis of discrete pushdown timed automata," CAV'00*, LNCS 1855, pp. 69-84
16. Z. Dang, O. H. Ibarra and R. A. Kemmerer, *"Decidable Approximations on Generalized and Parameterized Discrete Timed Automata," COCOON'01*, LNCS (to appear)
17. Z. Dang, P. San Pietro and R. A. Kemmerer, *"On Presburger Liveness of Discrete Timed Automata," STACS'01*, LNCS 2010, pp. 132-143
18. A. Finkel, B. Willems and P. Wolper, *"A direct symbolic approach to model checking pushdown systems," INFINITY'97.*
19. E. Gurari and O. Ibarra, *"The Complexity of Decision Problems for Finite-Turn Multicounter Machines," J. Computer and System Sciences*, **22** (1981) 220-229
20. T. A. Henzinger, Z. Manna, and A. Pnueli, *"What good are digital clocks?," ICALP'92*, LNCS 623, pp. 545-558
21. T. A. Henzinger and Pei-Hsin Ho, *"HyTech: the Cornell hybrid technology tool," Hybrid Systems II*, LNCS 999, pp. 265-294
22. T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. *"Symbolic Model Checking for Real-time Systems," Information and Computation*, **111** (1994) 193-244
23. O. H. Ibarra, *"Reversal-bounded multicounter machines and their decision problems," J. ACM*, **25** (1978) 116-133
24. K. G. Larsen, P. Pattersson, and W. Yi, *"UPPAAL in a nutshell," International Journal on Software Tools for Technology Transfer*, **1** (1997): 134-152
25. F. Laroussinie, K. G. Larsen, and C. Weise, *"From timed automata to logic - and back," MFCS'95*, LNCS 969, pp. 529-539
26. J. Raskin and P. Schobben, *"State clock logic: a decidable real-time logic," HART'97*, LNCS 1201, pp. 33-47
27. T. Wilke, *"Specifying timed state sequences in powerful decidable logics and timed automata,"* LNCS 863, pp. 694-715, 1994
28. S. Yovine, *"A verification tool for real-time systems," International Journal on Software Tools for Technology Transfer*, **1** (1997): 123-133
29. S. Yovine, *"Model checking timed automata," Embedded Systems'98*, LNCS 1494, pp. 114-152