Timed Pushdown Automata: Expressiveness and Reachability

著者	Uezato Yuya
year	2018
その他のタイトル	時間プッシュダウンオートマトンの表現力と到達可能性問題
学位授与大学	筑波大学 (University of Tsukuba)
学位授与年度	2017
報告番号	12102甲第8516号
URL	http://doi.org/10.15068/00152389

Timed Pushdown Automata: Expressiveness and Reachability

March 2018

Yuya Uezato

Timed Pushdown Automata: Expressiveness and Reachability

Graduate School of Systems and Information Engineering

University of Tsukuba

March 2018

Yuya Uezato

Abstract

Timed automata were introduced by Alur et al. in the early 1990's as finite automata augmented with finitely many (global) clocks. Adding clocks to finite automata enables us to model continuous time behaviors of real-time systems. The reachability problem of timed automata is decidable (PSPACE-complete) and it is key to automatically verifying temporal properties of real-time systems.

Several classes of pushdown extensions of timed automata have been introduced and studied. Bouajjani et al. introduced pushdown timed automata (PTA) as timed automata augmented with a single stack (alternatively, pushdown automata augmented with finitely many global clocks). Abdulla et al. introduced dense-timed pushdown automata (DTPDA) as timed automata with a timed stack whose stack element is a pair of a stack symbol and a single local clock. A DTPDA has unboundedly many local clocks during a computation; thus DTPDA clearly extend PTA. On the other hand, Clemente and Lasota showed PTA and DTPDA are equally expressive.

In this thesis, we introduce two new classes of timed pushdown automata: timed pushdown automata with multiple local clocks (MTPDA) and synchronized recursive timed automata (SRTA). MTPDA and SRTA can be seen as extensions of DTPDA with multipe local clocks and several new operations. We study the expressiveness and show the decidability of the reachability problem of the two classes.

About the expressiveness, on MTPDA, we show that MTPDA are as expressive as PTA in spite of having multiple local clocks and new operations. We show that SRTA are strictly more expressive than MTPDA owing to the presence of new constraints called fractional constraints that inspect the fractional parts of local clocks.

About the decidability of the reachability problem, we show that the reachability problems of MTPDA and SRTA are EXPTIME-complete. We also show that the reachability problem of PTA is already EXPTIME-complete even though PTA are the simplest form of timed pushdown automata.

Acknowledgments

I would like to sincerely thank my thesis advisor Yasuhiko Minamide who guided me to the world of the theory of automata. This dissertation could not be completed without his thoughful guidance. I am indebted to my supervisor Yukiyoshi Kameyama who supported my research by providing a great work environment. I am also indebted to Prof. Kazuhiko Kato, Prof. Kazuo Misue, Prof. Mitsuharu Yamamoto, and Dr. Hiroshi Unno who have offered me many comments about this work. I thank my colleagues and researchers who discussed me about my research and papers. They gave me many sophisticated lectures in various field of theoretical computer science.

Contents

Chapter 1

Introduction

The most important application of the theory of automata is model checking of systems and programs. We use automata to formally represent behaviors of systems and programs and verify their properties by results of the theory of automata. There is no perfect class of automata for all situations due to an unavoidable trade-off between expressiveness and decidability. The higher expressiveness a class of automata has, the more programs the class can precisely represent but the less properties on the class can be solved algorithmically. Therefore, we should use or develop an adequate class of automata for each situation. To this end, on each class of automata, it is important to clarify its expressiveness and decidability of typical decision problems.

Expressiveness. We compare the expressiveness of a model with those of existing similar models to understand the model. For this direction, the Chomsky hierarchy, **REG** $\subsetneq$ **CFL** $\subsetneq$ **CSL** $\subsetneq$ **REC** [Cho59], plays a central role where **REG** is the language class of finite automata (regular languages), **CFL** is the language class of pushdown automata (context-free languages), **CSL** is the language class of linear bounded automata (context-sensitive languages), and **REC** is the language class of Turing machines (recursively enumerable languages). Since Chomsky proposed these four classes, many important classes have been introduced (indexed languages [Aho68, Aho69], higher-order indexed languages [Mas74, Mas76, Dam82, Eng91], multiple context-free languages [SMFK91, Den16], tree-adjoining languages [JLT75, Vij87], etc). Chomsky's four classes, however, are still invaluable due to the simple formalization of the corresponding automata and are bases to develop a new class of automata.

Decidability. As the Chomsky hierarchy, the expressiveness is a good measure to study automata; however, studying solvability of typical decision problems is also a good measure. Customary decision problems studied in the theory of automata are the membership, emptiness, and equivalence problems. Let A be an automaton of some class.

Membership The membership problem decides whether the automaton accepts a given word w, $w \in_? L(A)$.

Emptiness The emptiness problem decides whether the language of the automaton is empty, $L(A) =_? \emptyset$.

Equivalence The equivalence problem decides whether the languages are the same for a given automaton B of the considered class, $L(A) =_? L(B)$.

Though these problems seem to be theoretical, we can use these problems for formal verification of systems and programs. Especially, the emptiness problem is important on formal verification. On many classes of automata (at least, Chomsky's four classes), the emptiness problem equals to a decision problem called reachability problem. This problem decides whether there is a valid computation from a given configuration $\boldsymbol{c}_{\mathrm{start}}$ to another one $\boldsymbol{c}_{\mathrm{goal}}$ ($\boldsymbol{c}_{\mathrm{start}} \Rightarrow_?^* \boldsymbol{c}_{\mathrm{goal}}$). On formal verification, we can use this problem for safety analysis, which decides whether there is a computation that reaches an unexpected (or unsafe) configuration from the entry point of a program. The emptiness problem is decidable for the classes **REG** and **CFL**, and not for **CSL** and **REC** [HU79].

The languages that are conventionally considered in the theory of formal languages are sets of words of finite alphabet. Formally, a language is a subset of Σ^* where Σ is a finite set of symbols. Towards formal verification of real-time systems, Alur et al. introduced timed automata by adding the notion of time to finite automata [AD90, ACD93, AD94]. On timed automata, words $w \in \Sigma^*$ and languages $L \subseteq \Sigma^*$ are extended to timed words $w^t \in (\Sigma \times \mathbb{R}_{\geq 0})^*$ and timed languages $L^t \subseteq (\Sigma \times \mathbb{R}_{\geq 0})^*$ where $\mathbb{R}_{\geq 0} = \{r \in \mathbb{R} : r \geq 0\}$ is the set of non-negative real numbers. Timed automata are finite automata augmented with finitely many clocks (variables over $\mathbb{R}_{\geq 0}$) and the formalization of timed automata is not so different from that of finite automata. However, from the viewpoint of decidability, the two models are different; the language equivalence problem of finite automata is decidable but that of timed automata is undecidable. Fortunately, the emptiness problem of timed automata remains decidable and thus we can use timed automata for formal verification of real-time systems; for example, timed automata has been used to verify temporal properties of communication and security protocols [BGK$^+$02, AJKO97, DKN04, NP05].

It is expected that adding clocks to an existing non-timed model of computation makes a new interesting model of computation as timed automata. Especially towards formal verification, we would like to design a model that is more expressive than timed automata and whose reachability problem remains decidable (the language equivalence problem is already undecidable for timed automata). We consider pushdown automata are an adequate candidate to be extended with clocks. The first reason is that it is located next to the class of finite automata in Chomsky's hierarchy and its reachability problem is decidable. The second reason is that pushdown automata have already been extended for many directions and the reachability problems of some interesting extensions of them are decidable (higher-order pushdown automata [HO08, HMOS08, HMOS17], pushdown vector addition automata [LST15], restricted classes of multi-stack pushdown automata [LMP07, MP11], etc.): therefore, we expect the reachability problems of timed-extensions of pushdown automata also remain decidable. Bouajjani, Echahed, and Robbana introduced a timed-extension of pushdown automata called pushdown timed automata (PTA) [BER94]. This model is a simple combination of timed automata and pushdown automata and its reachability problem is decidable. Abdulla, Atig, and Stenman extended pushdown timed automata and introduced dense-timed pushdown automata (DTPDA) [AAS12a]. Although the latter model is more complex and seems strictly more powerful than the former model, PTA and DTPDA are equally expressive [CL15a].

In this thesis, we extend existing timed-extensions of pushdown automata to enlarge their expressiveness while preserving the decidability of the reachability problem. We introduce two new timed-extensions of pushdown automata called timed pushdown automata with multiple local clocks (MTPDA) and synchronized recursive timed automata (SRTA). Our result on expressiveness is the following:

- MTPDA and PTA are equally expressive: **MTPDA** = **PTA** [Uez18].
- SRTA is more expressive than PTA: **PTA** $\subsetneq$ **SRTA** [UM18, UM15].

Our result on decidability is the following:

- both the reachability problems of MTPDA and SRTA are decidable and ExpTime-complete [Uez18, UM18, UM15].

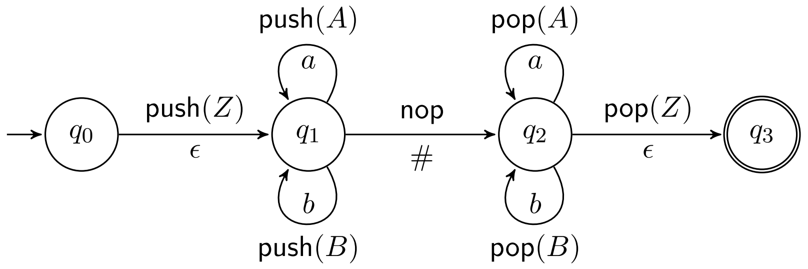
Hereafter we briefly review pushdown automata, timed automata, and DTPDA. Then, we see our two classes of timed pushdown automata and the differences from DTPDA.

Pushdown automata Pushdown automata (PDA) are finite automata augmented with a single stack [Eve63, Sch63, HU79]; therefore, a PDA is a 6-tuple $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \Delta)$ where Q is a finite set of locations, $q_{\text{init}} \in Q$ is the initial location, $F \subseteq Q$ is a set of accepting locations, Σ is a finite input alphabet, Γ is a finite stack alphabet, and Δ is a finite set of transition rules. A configuration of pushdown automata $\langle q, w\rangle$ consists of a control location q and a stack $w \in \Gamma^*$. Each transition rule is of the form $p \xrightarrow[\alpha]{\tau} q \in \Delta$ where $p, q \in Q$, $\alpha \in \Sigma \cup \{\epsilon\}$, $\tau \in \{\mathsf{nop}\} \cup \{\mathsf{push}(\gamma), \mathsf{pop}(\gamma) : \gamma \in \Gamma\}$. [1]

The operational semantics of a PDA is given by a labeled transition system $(Q\times\Gamma^*, \rightarrow)$ where $Q \times \Gamma^*$ is the set of states and $\rightarrow$ is the set of labeled transitions defined as follows:

$$\frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta}{\langle p, w\rangle \xrightarrow{\alpha} \langle q, w\gamma\rangle}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta}{\langle p, w\gamma\rangle \xrightarrow{\alpha} \langle q, w\rangle}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta}{\langle p, w\rangle \xrightarrow{\alpha} \langle q, w\rangle}.$$

The following is an example of PDA where $\Sigma = \{a, b\}$ and $\Gamma = \{A, B, Z\}$:



where q_0 is the initial location and q_3 is an accepting location. This PDA accepts the following language (here we omit the definition of the language of PDA):

$$L_1 = \left\{w\#w^R : w \in \{a, b\}^*, w^R \text{ is the reverse word of } w\right\}.$$

Pushdown automata play an important role in program verification. Since we can use a stack to implement the call-and-return mechanism of recursive programs, we can use pushdown automata as an abstract model of recursive programs. Furthermore, the reachability problem of pushdown automata is decidable and in PTIME [BEM97]. For verification of recursive programs, we use this decidability to solve the safety analysis which decides whether or not a given recursive program enters an undesirable state or exceptional state.

Timed automata Timed automata are finite automata augmented with finite clocks, which are used to measure the time elapsed between events [AD90, ACD93, AD94]. Formally, a timed automaton A is a 6-tuple $A = (Q, q_{\text{init}}, F, \Sigma, \mathcal{X}, \Delta)$ where Q, q_{init}, F, Σ, and Δ are the same as pushdown automata. The component $\mathcal{X}$ is a finite set of clocks. A configuration of timed automata $\langle q, \nu\rangle$ consists of a control location q and a valuation over finite clocks $\nu : \mathcal{X} \rightarrow \mathbb{R}_{\geq 0}$. On timed automata, we consider a timed word which belongs to the set $(\Sigma\times\mathbb{R}_{\geq 0})^*$ instead of a normal word on Σ^*. A timed word $(\sigma_1, r_1)(\sigma_2, r_2)\ldots(\sigma_n, r_n)$ means that a symbol σ_1 occurs in r_1 seconds after a computation started and also σ_2 occurs in r_2 seconds after the computation started (alternatively, σ_2 occurs in $r_2 - r_1$ seconds after the symbol σ_1 appeared), and so on. On timed automata, there are three types of transition rules: $p \xrightarrow[\alpha]{\mathsf{nop}} q$, $p \xrightarrow[\alpha]{\mathsf{reset}(x)} q$, and $p \xrightarrow[\alpha]{x \in_? I} q$ where x is a clock and I is an interval. These types of transition rules are called *discrete transition* rules. The operational semantics of a timed automaton is given by a labeled transition system $(Q \times (\mathcal{X} \rightarrow \mathbb{R}_{\geq 0}), \rightarrow, \rightsquigarrow)$

[1] For the sake of simplicity, we use a formalization of PDA that differs from the most standard one [HU79] but is equivalent to that.

where $Q \times (\mathcal{X} \to \mathbb{R}_{\geq 0})$ is the set of states and $\to$ is the set of labeled transitions over the set of states defined as follows:

$$\frac{p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta}{\langle p, \nu\rangle \xrightarrow[\alpha]{\mathsf{nop}} \langle q, \nu\rangle}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta}{\langle p, \nu\rangle \xrightarrow[\alpha]{\mathsf{reset}(x)} \langle q, \nu[x := 0]\rangle}, \qquad \frac{p \xrightarrow[\alpha]{x \in_? I} q \in \Delta \quad \nu(x) \in I}{\langle p, \nu\rangle \xrightarrow[\alpha]{x \in_? I} \langle q, \nu\rangle},$$

where $\nu[x := 0]$ is the reset valuation of ν for x by the real 0.0 defined as follows:

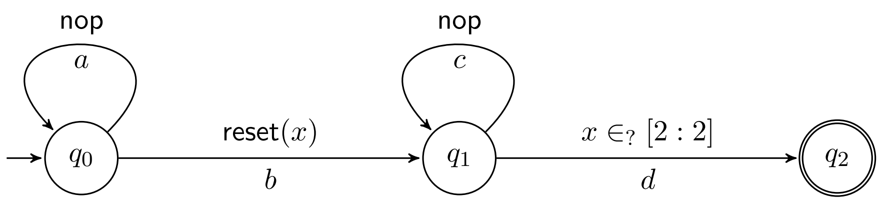
$$(\nu[x := 0])(y) \triangleq \begin{cases} 0.0 & \text{if } x = y, \\ \nu(y) & \text{otherwise.} \end{cases}$$

We also have another type of transition called *timed transitions*:

$$\frac{\delta \in \mathbb{R}_{\geq 0}}{\langle p, \nu\rangle \overset{\delta}{\rightsquigarrow} \langle p, \nu + \delta\rangle}$$

where $\nu + \delta$ is the evolved valuation of ν by δ defined as $(\nu + \delta)(x) \triangleq \nu(x) + \delta$.

The following is an example of a timed automaton where $\mathcal{X} = \{x\}$ and $\Sigma = \{a, b, c, d\}$:



This timed automaton represents the following timed language (here we omit the definition of the language of a timed automaton):

$$L_2 = \{\ (a, r_1) \dots (a, r_n)(b, r_{n+1})(c, r_{n+2}) \dots (c, r_{n+m})(d, r_{n+m+1}) : r_{n+m+1} - r_{n+1} = 2\}\,.$$

The timed language is the set of timed words of the form $aa \dots abcc \dots cd$ where d appears in two seconds after b appears.

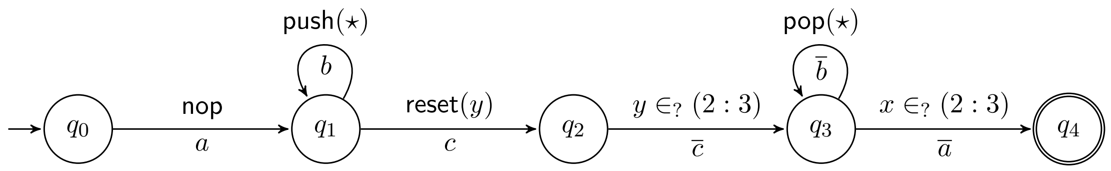
Alur et al. introduced timed automata and proved that the reachability problem of timed automata is decidable and PSPACE-complete with respect to the size of a timed automaton. In order to show the decidability of the reachability problem of timed automata, they developed a notion called region abstraction; today, it is an important and fundamental tool to study timed automata.

Timed pushdown automata It is natural to consider a hybrid model of pushdown automata and timed automata since both the reachability problems of these models are decidable. Bouajjani, Echahed, and Robbana considered the simplest combination of those two models, pushdown timed automata [BER94, Dan03]. Pushdown timed automata (PTA) are finite automata augmented with finitely many clocks and a single stack. Formally, a PTA is a 7-tuple $A = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ where each component has the same meaning as that of pushdown automata and timed automata. A configuration of PTA is a triple $\langle q, \nu, w\rangle$ of a location q, valuation $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$, and stack $w \in \Gamma^*$. Each transition rule is of the form $p \xrightarrow[\alpha]{\tau} q \in \Delta$ where $p, q \in Q$, $\alpha \in \Sigma \cup \{\epsilon\}$, and

$$\tau \in \{\mathsf{nop}\} \cup \{\mathsf{reset}(x), x \in_? I : x \in \mathcal{X}, I \text{ is an interval}\} \cup \{\mathsf{push}(\gamma), \mathsf{pop}(\gamma) : \gamma \in \Gamma\}\,.$$

The operational semantics of a PTA is given by a labeled transition system $(Q \times (\mathcal{X} \to \mathbb{R}_{\geq 0}) \times \Gamma^*, \to, \rightsquigarrow)$ where $Q \times (\mathcal{X} \to \mathbb{R}_{\geq 0}) \times \Gamma^*$ is the set of states, $\to$ is the set of labeled transitions for discrete transitions, and $\rightsquigarrow$ is the set of labeled transitions for timed transitions. The sets $\rightarrow$ and $\rightsquigarrow$ are defined in the same way as pushdown automata and timed automata.

Although this model is a simple combination of pushdown automata and timed automata, it strictly enlarges the language class of timed automata. The following is an example of PTA where $\mathcal{X} = \{x, y\}$, $\Sigma = \{a, b, c, \overline{a}, \overline{b}, \overline{c}\}$, and $\Gamma = \{\star\}$:



The above PTA accepts the following timed language by its accepting location q_4 and the empty stack:

$$\begin{aligned} L_3 = \{ & \\ & (a, r_1)(b, r_2) \dots (b, r_n)(c, r_{n+1})(\overline{c}, r'_{n+1})(\overline{b}, r'_n) \dots (\overline{b}, r'_2)(\overline{a}, r'_1) : \\ & 2 < r'_{n+1} - r_{n+1} < 3,\ 2 < r'_1 - r_1 < 3 \\ \}. & \end{aligned}$$

On the other hand, this language cannot be accepted by any timed automaton. In order to accept this language (more precisely, to check whether or not the number of b is the same as that of $\overline{b}$), we need a stack rather than clocks.

The reachability problem of PTA is decidable. This decidability was easily shown:

1. We translate a given PTA to the corresponding PDA that preserves the reachability by the technique of the region abstraction of timed automata;

2. We use the reachability analysis of pushdown automata.

Therefore, we can solve the reachability problem of PTA by using existing model checkers for pushdown automata. Applying the region abstraction technique to a given PTA yields the corresponding PDA where the number of control locations of the PDA is exponential with respect to the number of clocks of the PTA. Since the reachability problem of pushdown automata is in polynomial-time, we obtain an exponential-time algorithm of the reachability problem of PTA [BER94]. In this thesis, we newly show the exponential blowup is unavoidable, i.e., the reachability problem of PTA is EXPTime-complete (Corollary 3.1). On timed automata, the reachability problem is PSPACE-complete although there is a similar exponential blowup caused by applying the region abstraction [AD94]. If the complexity class PSPACE is strictly included in the class EXPTime (PSPACE $\subsetneq$ EXPTime), the coexistence of clocks and a stack makes that PTA essentially differ from timed automata and pushdown automata.

Abdulla, Atig, and Stenman introduced another timed-extension of pushdown automata, dense-timed pushdown automata (DTPDA) [AAS12a, AAS14b, AAS14a]. Formally, a DTPDA is a 7-tuple $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ as with PTA. A configuration of DTPDA $\langle q, \nu, \xi \rangle$ is a triple of a location q, valuation $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$, and timed stack $\xi \in (\Gamma \times \mathbb{R}_{\geq 0})^*$. A timed stack $\langle \gamma_1, k_1 \rangle \langle \gamma_2, k_2 \rangle \dots \langle \gamma_n, k_n \rangle$ is a stack whose each element $\langle \gamma_i, k_i \rangle$ is a pair of a stack symbol γ_i and value k_i of an accompanying clock. We call a clock of $\mathcal{X}$ global clock and an accompanying clock in a stack local clock. Hence, unboundedly many clocks appear in a timed stack as a timed stack unboundedly grows. Each transition rule is of the form $p \xrightarrow[\alpha]{\tau} q \in \Delta$ where $p, q \in Q$, $\alpha \in \Sigma \cup \{\epsilon\}$, and

$$\begin{aligned} \tau \in\ & \{\mathsf{nop}\} \cup \{\mathsf{reset}(x), x \in_? I : x \in \mathcal{X}, I \text{ is an interval}\} \\ \cup\ & \{\mathsf{push}(\gamma), \mathsf{pop}(\gamma, I) : \gamma \in \Gamma, I \text{ is an interval}\}. \end{aligned}$$

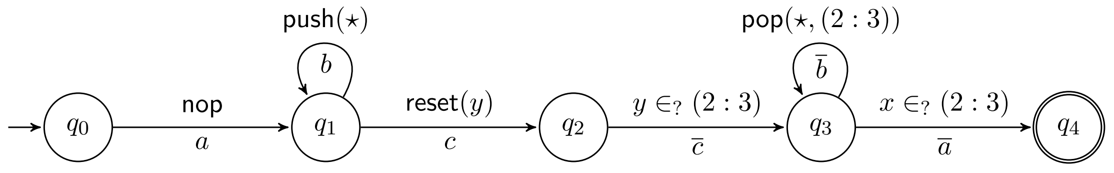
The operational semantics of a DTPDA is given by a labeled transition system $(Q \times (\mathcal{X} \to \mathbb{R}_{\geq 0}) \times (\Gamma \times \mathbb{R}_{\geq 0})^*, \to, \rightsquigarrow)$ where $Q \times (\mathcal{X} \to \mathbb{R}_{\geq 0}) \times (\Gamma \times \mathbb{R}_{\geq 0})^*$ is the set of states, $\to$ is the set of labeled transitions for discrete transitions, and $\rightsquigarrow$ is the set of labeled transitions for timed transitions. The transitions for $\xrightarrow[\alpha]{\mathsf{nop}}$, $\xrightarrow[\alpha]{\mathsf{reset}(x)}$, and $\xrightarrow[\alpha]{x \in_? I}$ are defined in the same way as PTA. We define the cases of $\xrightarrow[\alpha]{\mathsf{push}(\gamma)}$, $\xrightarrow[\alpha]{\mathsf{pop}(\gamma, I)}$, and $\overset{\delta}{\rightsquigarrow}$:

$$\frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta}{\langle p, \nu, w \rangle \xrightarrow[\alpha]{\mathsf{push}(\gamma)} \langle q, \nu, w \, \langle \gamma, 0.0 \rangle \rangle}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma, I)} q \in \Delta \quad k \in I}{\langle p, \nu, w \, \langle \gamma, k \rangle \rangle \xrightarrow[\alpha]{\mathsf{pop}(\gamma, I)} \langle q, \nu, w \rangle},$$

$$\frac{\delta \in \mathbb{R}_{\geq 0}}{\langle p, \nu, \langle \gamma_1, k_1 \rangle \dots \langle \gamma_n, k_n \rangle \rangle \overset{\delta}{\rightsquigarrow} \langle p, \nu + \delta, \langle \gamma_1, k_1 + \delta \rangle \dots \langle \gamma_n, k_n + \delta \rangle \rangle}.$$

Although PDA only modify the top part of a stack by the push and pop operations, DTPDA simultaneously evolve all the local clocks in their stack to reflect time-elapsing. It should also be noted that we can check the value of a local clock only when we pop the stack frame that the local clock belongs to. This means that we cannot check local clocks more than once and reset local clocks.

The following is an example of DTPDA where $\mathcal{X} = \{x, y\}$, $\Sigma = \{a, b, c, \overline{a}, \overline{b}, \overline{c}\}$, and $\Gamma = \{\star\}$:



This DTPDA represents the following timed language by the accepting loction q_4 and the empty stack:

$$L_4 = \left\{ \ (a, r_1)(b, r_2) \dots (b, r_n)(c, r_{n+1})(\overline{c}, r'_{n+1})(\overline{b}, r'_n) \dots (\overline{b}, r'_2)(\overline{a}, r'_1) : 2 < r'_i - r_i < 3 \ \right\}.$$

In spite of the unboundedness of the number of local clocks in a timed stack of DTPDA, its reachability problem remains decidable (and ExpTime-complete) [AAS12a]. Due to the presence of local clocks, we cannot use the same construction of the reachability problem of PTA for DTPDA. Indeed, Abdulla et al. adapted the classical region abstraction of timed automata for DTPDA and the decidability proof of the reachability problem for DTPDA significantly differs from that for PTA.

On the viewpoint from the theory of formal languages, Clemente and Lasota showed the untiming theorem of timed pushdown automata [CL15a]. The theorem says that we can translate a DTPDA to the corresponding PTA while preserving its language. This implies that PTA and DPTDA are equally expressive. Indeed, the above timed language L_4 can be accepted by a PTA with two clocks because the two languages L_3 and L_4 are the same. Following the untiming theorem of Clemente and Lasota, we obtain another proof of the decidability of the reachability problem of DTPDA.

Contribution

In this thesis, we introduce two classes of timed pushdown automata: timed pushdown automata with multiple local clocks (MTPDA) [Uez18] and synchronized recursive timed automata (SRTA) [UM18, UM15].

Timed pushdown automata with multiple local clocks (MTPDA). The class of MTPDA is an extension of timed pushdown automata of Clemente and Lasota with multiple local clocks. Formally, MTPDA (K-MTPDA) is an 8-tuple $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \{z_1, z_2, \dots, z_K\}, \Delta)$ as with DTPDA except a set of K-local clocks $\{z_1, z_2, \dots, z_K\}$. A configuration of MTPDA is a triple $\langle q, \nu, \Upsilon \rangle$ of a control location q, clock valuation ν, and timed stack $\Upsilon \in \left(\Gamma \times (\{z_1, z_2, \dots, z_K\} \to \mathbb{R}_{\geq 0})\right)^*$. Although DTPDA cannot reset and check local clocks of a timed stack, MTPDA allow those operations by transition rules of the form $p \xrightarrow[\alpha]{\mathsf{reset}(z_i)} q$ and $p \xrightarrow[\alpha]{z_i \in_? I} q$ where z_i is a local clock rather than a global clock and I is an interval.

Our result on MTPDA is the following:

Expressiveness MTPDA and PTA are equally expressive.

Decidability The reachability problem of MTPDA is ExpTime-complete.

In order to show these results, we prove the untiming theorem of MTPDA that generalizes the untiming theorem of Clemente and Lasota. The untiming theorem of MTPDA translates a given MTPDA to the corresponding PTA while preserving its language.

Synchronized Recursive Timed Automata (SRTA). The class of SRTA can be seen as an extension of MTPDA with a new kind of clock constraints called fractional constraints. Each fractional constraint $\{x\} =_? 0.0$ checks whether or not the fractional part of the value of a (global or local) clock x is 0.0. In other words, it checks whether or not the value of a clock is a natural number.

Our result on SRTA is the following:

Expressiveness SRTA is more expressive than PTA; therefore, **PTA** = **DTPDA** = **MTPDA** $\subsetneq$ **SRTA**.

Decidability The reachability problem of MTPDA is ExpTime-complete.

Although we can easily simulate a PTA while preserving its language by an SRTA, the following timed language L_{SRTA} cannot be accepted by any PTA because it requires unboundedly many clocks in a stack:

$$L_{\text{SRTA}} = \left\{ \quad (a, r_1)(a, r_2) \dots (a, r_n)(b, r'_n) \dots (b, r'_2)(b, r'_1) : r'_i - r_i \in \mathbb{N} \right\}.$$

This unrecognizability result means that we cannot untime stacks of SRTA and use the same argument as MTPDA to show the decidability of the reachability problem of SRTA. Alternatively, we use the region abstraction designed by Abdulla et al. that was introduced to show the decidability of the reachability problem of their DTPDA. Although our proof is based on the region abstraction of Abdulla et al., we clarify key structures of their elaborated proof and give a simpler decidability proof. Technically, our proof simplification comes from backward-simulation of SRTA by pushdown automata obtained by region abstraction. This does not only simplify our proof but also generalizes the decidability result of timed pushdown automata. On the previous work, the decidability of the location reachability problem of timed pushdown automata has been studied [BER94, AAS12a]. We generalize this to the decidability of the configuration reachability problem and show both the reachability problems of SRTA remain ExpTime-complete [UM18, UM15].

Overview of the Thesis

In Chapter 2, we will introduce basic notation and formalize timed automata. We review the classical decidability result of timed automata, the decidability of the location reachability problem of timed automata. In addition, through introducing collapsed valuations and the semantics of timed automata based on such valuations, we show the PSPACE-completeness of the configuration reachability problem of timed automata. As far as we know, this decidabiliy result is new.

In Chapter 3, as a preliminary to our new extensions of timed pushdown automata, we will review pushdown timed automata (PTA) of Bouajjani et al. [BER94], dense-timed pushdown automata (DTPDA) of Abdulla et al. [AAS12a], and timed pushdown automata (TPDA) of Clemente and Lasota [CL15a]. We also review the known result that PTA, DTPDA, and TPDA are equally expressive. As a new result for PTA, we show that the reachability problem of PTA is already PSPACE-complete. This PSPACE-hardness refines the PSPACE-hardness of DTPDA shown by Abdulla et al.

In Chapter 4, we will introduce a new extension of timed pushdown automata, timed pushdown automata with multiple local clocks (MTPDA). For this class, we show the untiming theorem which implies that MTPDA and PTA are equally expressive and that the reachability problem of MTPDA is EXPTIME-complete.

In Chapter 5, we will introduce a new extension of timed pushdown automata, synchronized recursive timed automata (SRTA). We will show that SRTA is more expressive than PTA and the reachability problem of SRTA remains EXPTIME-complete. Finally, we will conclude in Chapter 6.

Chapter 2

Timed Automata

In this chapter, we revisit timed automata introduced by Alur et al [AD94]. We define basic notation and then formalize timed automata. We also show the decidability of the important decision problem of timed automata called the location reachability problem along with introducing digital valuations that are useful tool to analyze timed automata. Our digital valuations and the region of timed automata introduced by Alur et al. are similar but slightly different. Although we can also use the region of timed automata to show the decidability of the location reachability problem, we will use digital valuations instead of the classical region in Chapter 5; therefore, we introduce them in this chapter as a preparation for that chapter. Furthermore, we show the decidability of the configuration reachability problem of timed automata through the technique called a backward simulation. As far as the author know, our decidability proof of the configuration reachability problem differs from existing approaches.

2.1 Basic Notation

We define basic notation to define timed automata.

The set of natural numbers and real numbers are written by $\mathbb{N}$ and $\mathbb{R}$, respectively. We use $\mathbb{R}_{\geq 0}$ to denote the set of non-negative real numbers: $\mathbb{R}_{\geq 0} \triangleq \{r \in \mathbb{R} : r \geq 0\}$. For a real number $r \in \mathbb{R}$, we write $\lfloor r \rfloor$ and $\mathit{frac}(r)$ to denote the integral part and fractional part of r, respectively. For example, $\lfloor 3.14 \rfloor = 3$ and $\mathit{frac}(3.14) = 0.14$.

We use intervals of the following form to denote a set of real numbers:

$$\begin{aligned}
(a : b) &\triangleq \{r \in \mathbb{R} : a < r < b\}\,, & (a : b] &\triangleq \{r \in \mathbb{R} : a < r \leq b\}\,, \\
[a : b) &\triangleq \{r \in \mathbb{R} : a \leq r < b\}\,, & [a : b] &\triangleq \{r \in \mathbb{R} : a \leq r \leq b\}\,, \\
(a : \omega) &\triangleq \{r \in \mathbb{R} : a < r\}\,, & [a : \omega) &\triangleq \{r \in \mathbb{R} : a \leq r\}\,.
\end{aligned}$$

where $a, b \in \mathbb{N}$.

We will use $\mathcal{X}$ to denote a set of clocks of a timed automaton and call a function $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$ from $\mathcal{X}$ to $\mathbb{R}_{\geq 0}$ *concrete valuation* or simply *valuation* on $\mathcal{X}$. The special valuation that assigns 0.0 to any clock of $\mathcal{X}$ is written as $\mathbf{0}_{\mathcal{X}}$: $\mathbf{0}_{\mathcal{X}}(x) = 0.0$ for any $x \in \mathcal{X}$. If $\mathcal{X}$ is clear from the context, we omit it and simply write $\mathbf{0}$.

Let $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be a valuation. We write $\nu[x \coloneqq r]$ to denote the valuation obtained by updating the value of x by r:

$$(\nu[x \coloneqq r])(y) \triangleq \begin{cases} r & \text{if } x = y, \\ \nu(y) & \text{otherwise.} \end{cases}$$

Let $\nu' : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be another valuation and $Y \subseteq \mathcal{X}$ be a subset of $\mathcal{X}$. We write $\nu[Y \coloneqq \nu']$

to denote the updated valuation of ν defined as follows:

$$\nu[Y := \nu'](x) \triangleq \begin{cases} \nu(x) & \text{if } x \notin Y, \\ \nu'(x) & \text{otherwise.} \end{cases}$$

For a subset $Y \subseteq \mathcal{X}$, we write $\nu \restriction Y : Y \to \mathbb{R}_{\geq 0}$ to denote the restriction of ν to Y. We write $\nu \models x \in I$ if, on the valuation ν, the value of x belongs to the interval I:

$$\nu \models x \in I \overset{\text{def}}{\iff} \nu(x) \in I.$$

For two valuations $\nu_1 : X \to \mathbb{R}_{\geq 0}$ and $\nu_2 : Y \to \mathbb{R}_{\geq 0}$ where their domains are disjoint, $X \cap Y = \emptyset$, we write $\nu_1 \cup \nu_2$ to denote the valuation on $X \cup Y$ that is defined as follows:

$$(\nu_1 \cup \nu_2)(z) \triangleq \begin{cases} \nu_1(z) & \text{if } z \in X, \\ \nu_2(z) & \text{otherwise.} \end{cases}$$

For two valuations $\nu_1, \nu_2 : \mathcal{X} \to \mathbb{R}_{\geq 0}$, we write $\nu_1 \leq \nu_2$ if $\nu_1(x) \leq \nu_2(x)$ for any $x \in \mathcal{X}$. It is clear that this relation $\leq$ forms an ordering.

2.2 Formalization of Timed Automata

A timed automaton A is a 6-tuple $A = (Q, q_{\text{init}}, F, \Sigma, \mathcal{X}, \Delta)$ where

- Q is a finite set of control locations, $q_{\text{init}} \in Q$ is the initial location, $F \subseteq Q$ is a finite set of accepting locations,
- Σ is a finite input alphabet,
- $\mathcal{X}$ is a finite set of clocks, and
- $\Delta \subseteq Q \times \Sigma_\epsilon \times \mathit{Act} \times Q$ is a finite set of transition rules.
  - We write Σ_ϵ to denote the set $\Sigma \cup \{\epsilon\}$.
  - *Act* is the set of *actions* of timed automata defined as the following grammar:

    $$\mathit{Act} ::= \mathsf{reset}(x) \mid x \in_? I$$

    where $x \in \mathcal{X}$ and I is an interval.

In the present chapter, we fix a timed automaton $A = (Q, q_{\text{init}}, F, \Sigma, \mathcal{X}, \Delta)$ to define some notation.

A configuration of the timed automaton A is a pair $\langle q, \nu \rangle$ of a location $q \in Q$ and clock valuation $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$. Especially, we call the configuration $\langle q_{\text{init}}, \mathbf{0} \rangle$ *initial configuration.* We write $p \xrightarrow[\alpha]{\tau} q$ to denote a transition rule $\langle p, \alpha, \tau, q \rangle \in \Delta$.

The operational semantics of the timed automaton A is defined as an infinite labeled transition system $T_A = (Q \times (\mathcal{X} \to \mathbb{R}_{\geq 0}), \to, \rightsquigarrow)$. First, we define *discrete transitions* $\to$ as follows:

$$\frac{p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta}{\langle p, \nu \rangle \xrightarrow[\alpha]{\mathsf{reset}(x)} \langle q, \nu[x := 0] \rangle}, \qquad \frac{p \xrightarrow[\alpha]{x \in_? I} q \in \Delta \quad \nu \models x \in I}{\langle p, \nu \rangle \xrightarrow[\alpha]{x \in_? I} \langle q, \nu \rangle}.$$

The transition rule $p \xrightarrow[\alpha]{\mathsf{reset}(x)} q$ resets the designated clock x by assigning 0.0 to x in a valuation. The transition rule $p \xrightarrow[\alpha]{x \in_? I} q$ checks whether or not the designated clock x is in the interval I in a valuation.

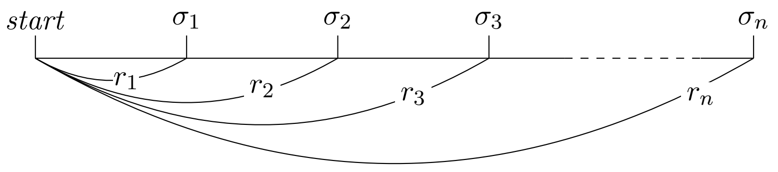
Next, we define *timed transitions* $\rightsquigarrow$ that reflects time-elapsing as follows:

$$\frac{\delta \in \mathbb{R}_{\geq 0}}{\langle p, \nu \rangle \overset{\delta}{\rightsquigarrow} \langle p, \nu + \delta \rangle.}$$

The timed transition $\boldsymbol{c}_1 \overset{\delta}{\rightsquigarrow} \boldsymbol{c}_2$ denotes the time-elapsing of δ time units. It is worth noting that timed transitions only affect valuations and do not change locations.

On timed automata, we consider *timed words* and *timed languages* over $(\Sigma \times \mathbb{R}_{\geq 0})^*$ instead of (untimed) words and language over Σ^* of ordinary finite automata. A timed word $w \in (\Sigma \times \mathbb{R}_{\geq 0})^*$ is a finite sequence of pairs of Σ and $\mathbb{R}_{\geq 0}$. Intuitively, a timed word $w = (\sigma_1, r_1)(\sigma_2, r_2) \dots (\sigma_n, r_n)$ means that

- a symbol σ_1 appears in r_1 seconds after the computation started
- a symbol σ_2 appears in r_2 seconds after the computation started, and so on:



More formally, timed words are weakly monotonic sequences on $(\Sigma \times \mathbb{R}_{\geq 0})^*$; therefore, the set of timed words $\mathbf{TW}(\Sigma)$ is defined as follows:

$$\mathbf{TW}(\Sigma) \triangleq \{(\sigma_1, r_1)(\sigma_2, r_2) \dots (\sigma_n, r_n) : \sigma_i \in \Sigma, r_i \in \mathbb{R}_{\geq 0}, r_i \leq r_{i+1}\}.$$

A timed language $L \subseteq \mathbf{TW}(\Sigma)$ is a set of timed words.

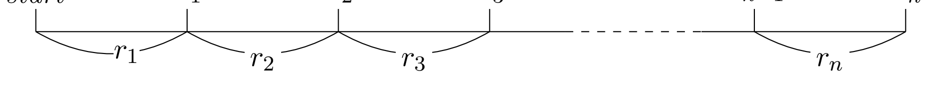
2.2.1 Language of Timed Automata

In order to define the timed language of a timed automaton, we need some notation. A *computation* π of a timed automaton is a finite alternating sequence of timed and discrete transitions that starts from the initial configuration $\langle q_{\text{init}}, \mathbf{0} \rangle$ as follows:

$$\pi \equiv \langle q_{\text{init}}, \mathbf{0} \rangle \overset{\delta_1}{\rightsquigarrow} \boldsymbol{c}_1 \xrightarrow[\alpha_1]{\tau_1} \boldsymbol{c}'_1 \overset{\delta_2}{\rightsquigarrow} \boldsymbol{c}_2 \xrightarrow[\alpha_2]{\tau_2} \boldsymbol{c}'_2 \overset{\delta_3}{\rightsquigarrow} \cdots \xrightarrow[\alpha_{n-1}]{\tau_{n-1}} \boldsymbol{c}'_{n-1} \overset{\delta_n}{\rightsquigarrow} \boldsymbol{c}_n \xrightarrow[\alpha_n]{\tau_n} \boldsymbol{c}'_n.$$

Let $\pi = \langle q_{\text{init}}, \mathbf{0} \rangle \overset{\delta_1}{\rightsquigarrow}\xrightarrow[\alpha_1]{\tau_1} \boldsymbol{c}'_1 \overset{\delta_2}{\rightsquigarrow}\xrightarrow[\alpha_2]{\tau_2} \cdots \overset{\delta_n}{\rightsquigarrow}\xrightarrow[\alpha_n]{\tau_n} \boldsymbol{c}'_n$ be a computation. Intuitively, this computation means that

- a symbol α_1 (maybe $\alpha_1 = \epsilon$) appears in r_1 seconds after the computation started
- a symbol α_2 (maybe $\alpha_2 = \epsilon$) appears in r_2 seconds after α_1 appeared, and so on.



Following this intuition, we define the timed trace $\texttt{tt}(\pi) \in (\Sigma_\epsilon \times \mathbb{R}_{\geq 0})^*$ for the computation π as follows:

$$\texttt{tt}(\pi) \equiv \langle \alpha_1, \delta_1 \rangle \langle \alpha_2, \delta_1 + \delta_2 \rangle \dots \langle \alpha_n, \sum_{i=1}^{n} \delta_i \rangle.$$

Excluding silent transitions from timed traces, we define the timed word $\mathtt{tw}(\pi) \in (\Sigma \times \mathbb{R}_{\geq 0})^*$ for the computation: $\mathtt{tw}(\pi) \triangleq \Psi(\mathtt{tt}(\pi))$ where Ψ is the homomorphism from $(\Sigma_\epsilon \times \mathbb{R}_{\geq 0})^* \to (\Sigma \times \mathbb{R}_{\geq 0})^*$ defined as follows:

$$\Psi(\langle \alpha, r\rangle) = \begin{cases} \langle \alpha, r\rangle & \text{if } \alpha \in \Sigma, \\ \epsilon & \text{otherwise.} \end{cases}$$

We define the language of a timed automaton as the set of timed words obtained from the computations that reach a configuration whose location is an accepting location:

$$L(A) \triangleq \left\{ \mathtt{tw}(\pi) : \pi = \langle q_{\text{init}}, \mathbf{0}\rangle \overset{\delta_1}{\rightsquigarrow} \xrightarrow[\alpha_1]{\tau_1} \cdots \overset{\delta_n}{\rightsquigarrow} \xrightarrow[\alpha_n]{\tau_n} \langle q_F, \nu\rangle,\ q_F \in F \right\}.$$

Normalizing Intervals. We can normalize intervals that appears in a given timed automaton while preserving its language.

Proposition 2.1. Let A be a timed automaton. There is a timed automaton B such that $L(A) = L(B)$ and if a rule $p \xrightarrow[\alpha]{x \in_? I} q \in B$, then $I = (a : a+1)$, $I = [a : a]$, or $I = (a : \omega)$.

Proof. This is immediately shown by the following simple equation:

$$\begin{aligned}
&(a : a+k) = (a : a+1) \cup [a+1 : a+1] \cup (a+1 : a+2) \cup \cdots \cup (a+(k-1) : a+k), \\
&(a : a+k] = (a : a+k) \cup [a+k : a+k], \\
&[a : a+k) = [a : a] \cup (a : a+k), \\
&[a : a+k] = [a : a] \cup (a : a+k], \\
&[a : \omega) = [a : a] \cup (a : \omega).
\end{aligned}$$

If we have a transition $p \xrightarrow[\alpha]{\mathsf{reset}(x)} q$ in A, then we add it to B.

If we have a transition $p \xrightarrow[\alpha]{x \in_? I} q$ in A, then we add transitions obtained by decomposing I to B. For example, if $p \xrightarrow[\alpha]{x \in_? (1:3]} q$ in A, then add the following transitions:

$$p \xrightarrow[\alpha]{x \in_? (1:2)} q, \quad p \xrightarrow[\alpha]{x \in_? [2:2]} q, \quad p \xrightarrow[\alpha]{x \in_? (2:3)} q, \quad p \xrightarrow[\alpha]{x \in_? [3:3]} q.$$

□

Atomic Operations. We consider a new action $\tau_1 ⨟ \tau_2$ that performs two actions τ_1 and τ_2 sequentially at a single transition without time-elapsings. Therefore, the semantics of a transition rule $p \xrightarrow[\alpha]{\tau_1 ⨟ \tau_2} q$ is defined as follows:

$$\frac{\langle p, \nu\rangle \xrightarrow[\epsilon]{\tau_1} \langle p', \nu'\rangle \quad \langle p', \nu'\rangle \xrightarrow[\epsilon]{\tau_2} \langle q, \nu''\rangle}{\langle p, \nu\rangle \xrightarrow[\alpha]{\tau_1 ⨟ \tau_2} \langle q, \nu''\rangle}$$

We call this transition rule *atomic* transition rule. Adding atomic transition rules does not enlarge the expressiveness of timed automata.

Proposition 2.2. Let A be a timed automaton with atomic transition rules. There is a timed automaton B such that $L(A) = L(B)$.

Proof. Let $A = (Q, q_{\text{init}}, F, \Sigma, \mathcal{X}, \Delta)$ and $p \xrightarrow[\alpha]{\tau_1 ⨟ \tau_2} q$ be an atomic transition rule of A. We remove this transition rule by constructing the following timed automaton:

$$B_1 = (Q \cup \{p_0, p_1, p_2\}, q_{\text{init}}, F, \Sigma, \mathcal{X} \cup \{\mathtt{C}\}, \Delta')$$

where p_0, p_1, p_2 are fresh locations, $\mathtt{C}$ is a fresh clock, and Δ' is defined as follows:

$$\begin{aligned}\Delta &= \Delta' \setminus \left\{ p \xrightarrow[\alpha]{\tau_1 ⨟ \tau_2} q \right\} \\ &\cup \left\{ p \xrightarrow[\alpha]{\mathsf{reset}(\mathtt{C})} p_0, \quad p_0 \xrightarrow[\epsilon]{\tau_1} p_1, \quad p_1 \xrightarrow[\epsilon]{\tau_2} p_2, \quad p_2 \xrightarrow[\epsilon]{\mathtt{C} \in [0:0]} q \right\}.\end{aligned}$$

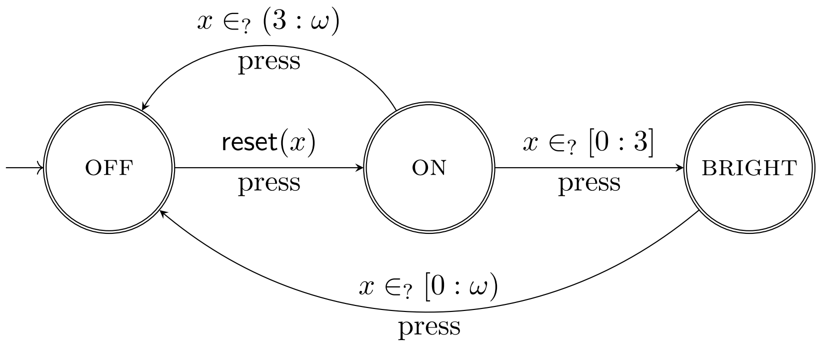
We use the fresh clock $\mathtt{C}$ to ensure that there are no time-elapsings among moving p to q through p_1, p_2, and p_3. It can be easily verified $L(A) = L(B_1)$. Furthermore, we can remove all the atomic transition rules of B_1 by repeatedly applying the same construction. Finally, we obtain a timed automaton B without atomic transition rules such that $L(A) = L(B)$. □

We also use a more general form $p \xrightarrow[\alpha]{\tau_1 ⨟ \tau_2 ⨟ \cdots ⨟ \tau_n} q$. Adding such atomic transition rules also does not enlarge the expressiveness of timed automata; indeed, on the basis of the same argument of Proposition 2.2, we can remove such atomic transition rules.

Example of Timed Automaton As an example of timed automata, we model a time-dependent light-switch that behaves as follows:

- As the initial status of the light-switch, the light is off.
- If we push the switch, the light becomes on.
- Furthermore, if we again push the switch within 3 seconds after the light becomes on, then the light becomes bright.
- Otherwise, the light becomes off when we push the switch.

The following timed automaton $A = (Q = \{\text{OFF}, \text{ON}, \text{BRIGHT}\}, \text{OFF}, Q, \{\text{press}\}, \{x\}, \Delta)$ models the above behavior:



Let us consider the following two computations:

$$\begin{aligned}\pi_1 = \langle \text{OFF}, \{x \mapsto 0.0\}\rangle &\overset{0.4}{\rightsquigarrow} \langle \text{OFF}, \{x \mapsto 0.4\}\rangle \xrightarrow[\text{press}]{\mathsf{reset}(x)} \\ \langle \text{ON}, \{x \mapsto 0.0\}\rangle &\overset{6.3}{\rightsquigarrow} \langle \text{ON}, \{x \mapsto 6.3\}\rangle \xrightarrow[\text{press}]{x \in_? (3:\omega)} \langle \text{OFF}, \{x \mapsto 6.3\}\rangle.\end{aligned}$$

$$\begin{aligned}\pi_2 = \langle \text{OFF}, \{x \mapsto 0.0\}\rangle &\overset{0.4}{\rightsquigarrow} \langle \text{OFF}, \{x \mapsto 0.4\}\rangle \xrightarrow[\text{press}]{\mathsf{reset}(x)} \\ \langle \text{ON}, \{x \mapsto 0.0\}\rangle &\overset{2.1}{\rightsquigarrow} \langle \text{ON}, \{x \mapsto 2.1\}\rangle \xrightarrow[\text{press}]{x \in_? [0:3]} \\ \langle \text{BRIGHT}, \{x \mapsto 2.1\}\rangle &\overset{1.9}{\rightsquigarrow} \langle \text{BRIGHT}, \{x \mapsto 4.0\}\rangle \xrightarrow[\text{press}]{x \in_? [0:\omega)} \langle \text{OFF}, \{x \mapsto 4.0\}\rangle.\end{aligned}$$

From each computation, we obtain the following timed words:

$$\mathtt{tw}(\pi_1) = \langle \text{press}, 0.4\rangle\langle \text{press}, 6.7\rangle, \qquad \mathtt{tw}(\pi_2) = \langle \text{press}, 0.4\rangle\langle \text{press}, 2.5\rangle\langle \text{press}, 4.4\rangle.$$

2.3 Reachability Problem and Digital Automata

In the present section, we show the classical decidability problem of timed automata called the *location reachability problem* or simply the *reachability problem.* For a given location q of a timed automaton, the location reachability problem of timed automata $\langle q_{\mathrm{init}}, \mathbf{0} \rangle \Rightarrow^*_? \langle q, {}^\exists\nu \rangle$ decides whether or not there is a computation that starts from the initial configuration reaches a configuration $\langle q, \nu \rangle$ where q is the designated location and ν is some valuation.

The decidability (and PSPACE-completeness) of this problem was shown using the important technique of timed automata called *region abstraction* [AD94]. They used the region abstraction to remove two infiniteness in timed automata: 1) the unboundedness of real numbers and 2) the denseness of real numbers. We introduce *digital valuations* instead of the classical regions of [AD94] and construct *digital automata* from timed automata to show the decidability of the reachability problem of timed automata. Although the classical region suffices to show the decidability of the reachability problem of timed automata, we introduce digital valuations in advance as a preliminary for Chapter 5 where we need digital valuations rathar than the classical regions to establish an important lemma.

To define digital valuations and digital automata, we fix a timed automaton $A = (Q, q_{\mathrm{init}}, F, \Sigma, \mathcal{X}, \Delta)$ whose intervals are normalized by following Proposition 2.1.

2.3.1 Digital Valuations

We write M_A to denote a sufficiently large natural number with respect to the constants appearing in Δ. Formally, we define M_A as follows:

$$\mathsf{M}_A \triangleq \max\{\, i, j \in \mathbb{N} : (i:j),\ [i:j],\text{ or } (i:\omega) \text{ appears in } \Delta\} + 1.$$

We simply write M by omitting A from M_A when a timed automaton A is clear from the context. We cannot distinguish large real numbers r_1 and r_2 such that $r_1, r_2 \geq \mathsf{M}$ with any transition rule $p \xrightarrow[\alpha]{x \in_? I} q \in \Delta$.

Proposition 2.3. Let r_1 and r_2 be real numbers such that $r_1, r_2 \geq \mathsf{M}$.
If $p \xrightarrow[\alpha]{x \in_? I} q$ is a transition rule of the timed automaton A, then the following holds:

$$\{x \mapsto r_1\} \models x \in I \iff \{x \mapsto r_2\} \models x \in I.$$

Proof. This is trivial from the definition of M. □

On the basis of this proposition, we can forget values of a clock valuation that are beyond M. This is useful to deal with the first infiniteness of timed automata, the unboundedness of real numbers.

To deal with the second infiniteness of timed automata (the denseness of real numbers), we forget the fractional parts of a clock valuation but keep the ordering of the fractional parts of the clock valuation. Indeed, we cannot check whether or not the value of a clock x is a real constant with any clock constraint $x \in_? I$. (However, by some constraint such as $x \in [n:n]$, we can check if the fractional part of a clock x is 0.0.)

Following these intuitions, we define digital valuations.

Definition 2.1 (Digital Valuation)**.** A sequence of sets $\boldsymbol{d} = d_0\, d_1 \ldots d_n$, where $d_i \subseteq \mathcal{X} \times \{0, 1, \ldots, \mathsf{M}-1, \infty\}$, is a *digital valuation* on $\mathcal{X}$ if $\boldsymbol{d}$ satisfies the following conditions:

- Every clock in $\mathcal{X}$ appears in $\boldsymbol{d}$ exactly once.
- Except d_0, all the sets d_i are not empty: $d_i \neq \emptyset$ for all $i \in \{1, 2, \ldots, n\}$.

We write $(x, k) \in \boldsymbol{d}$ if $(x, k) \in d_i$ for some $i \in \{0, 1, \dots, n\}$.

We use $\mathbb{D}(\mathcal{X}, \mathsf{M})$ to denote the set of digital valuations on a finite clock set $\mathcal{X}$ and constant M. We simply write $\mathbb{D}$ if $\mathcal{X}$ and M are clear from the context. ■

We define a realization relation between concrete valuations and digital valuations.

Definition 2.2 (Realization)**.** Let ν be a valuation on $\mathcal{X}$, and $\boldsymbol{d} = d_0 d_1 \dots d_n$ be a digital valuation on $\mathcal{X}$. We write $\nu \models \boldsymbol{d}$ if the following hold:

- For all $x \in \mathcal{X}$, $\nu(x) \geq \mathsf{M}$ iff $(x, \infty) \in \boldsymbol{d}$.
- If $\nu(x) < \mathsf{M}$, then $(x, \lfloor \nu(x) \rfloor) \in \boldsymbol{d}$.
- If $(x, k) \in \boldsymbol{d}$ with some $k \in \{0, 1, \dots, \mathsf{M} - 1\}$, then $\lfloor \nu(x) \rfloor = k$.
- For all $x \in \mathcal{X}$, $\mathit{frac}(\nu(x)) = 0.0$ iff $x \in d_0$.
- $\mathit{frac}(\nu(x)) < \mathit{frac}(\nu(y))$ iff $x \in d_i$ and $y \in d_j$ for some $i < j$.

■

Example. Let $\mathsf{M} = 4$.

$$\{x \mapsto 2.0;\ y \mapsto 4.3\} \models \{(x,2)\}_0 \, \{(y,\infty)\} \, ,$$
$$\{x \mapsto 0.8;\ y \mapsto 1.5;\ z \mapsto 3.8\} \models \{\}_0 \, \{(y,1)\} \, \{(x,0),(z,3)\} \, ,$$
$$\{x \mapsto 4.0;\ y \mapsto 2.5;\ z \mapsto 5.5\} \models \{(x,\infty)\}_0 \, \{(y,1),(z,\infty)\} \, .$$

Remark: For the special sets d_0 that contain clocks whose fractional parts are 0.0, we use the notation $\{\dots\}_0$ as above.

We can easily construct a digital valuation from a valuation by collapsing the integral parts of the valuation to $\{0, 1, \dots, \mathsf{M} - 1, \infty\}$ and sorting fractional parts of the valuation. Therefore, the following simple property holds.

Proposition 2.4. The realization relation $\models$ is *functional*, i.e., for a valuation ν, there exists the unique digital valuation $\mathcal{D}(\nu)$ such that $\nu \models \mathcal{D}(\nu)$.

On the other hand, the realization relation $\models$ is not injective: for example, $\{x \mapsto 0.3\} \models \{\}_0 \, \{(x,0)\}$ and $\{x \mapsto 0.9\} \models \{\}_0 \, \{(x,0)\}$.

We define operations, clock checking $\boldsymbol{d} \models x \in I$ and clock resetting $\boldsymbol{d}[x := 0]$, for digital valuations. To this end, we need the following properties.

Proposition 2.5. Let ν_1 and ν_2 be valuations on $\mathcal{X}$. If $\mathcal{D}(\nu_1) = \mathcal{D}(\nu_2)$,

Checking. $\nu_1 \models x \in_? I$ iff $\nu_2 \models x \in_? I$ for any clock constraint $x \in_? I$ in Δ of the timed automaton A.

Resetting. $\mathcal{D}(\nu_1[x := 0]) = \mathcal{D}(\nu_2[x := 0])$ for any clock $x \in \mathcal{X}$.

On the basis of this proposition, we define $\boldsymbol{d} \models x \in I$ and $\boldsymbol{d}[x := 0]$ for digital valuations.

Definition 2.3. Let $\boldsymbol{d}$ be a digital valuation and ν be a valuation such that $\nu \models \boldsymbol{d}$.

- For a constraint $x \in_? I$, $\boldsymbol{d} \models x \in I$ if $\nu \models x \in I$.
- For a clock $x \in \mathcal{X}$, $\boldsymbol{d}[x := 0] \triangleq \mathcal{D}(\nu[x := 0])$.

■

Remark: The definition for a digital valuation $\boldsymbol{d}$ uses a valuation ν that satisfies $\nu \models \boldsymbol{d}$. But the definition is well-defined and does not depend on the choice of such a valuation because of Proposition 2.5.

The definition immediately leads to the following proposition.

Proposition 2.6. Let ν be a valuation and $\boldsymbol{d}$ be a digital valuation. If we have $\nu \models \boldsymbol{d}$,

- $\nu \models x \in I \iff \boldsymbol{d} \models x \in I$ for any clock constraint in Δ.
- $\nu[x := 0] \models \boldsymbol{d}[x := 0]$ for any clock $x \in \mathcal{X}$.

We define the successor relation $\boldsymbol{d} \vdash \boldsymbol{d'}$ that corresponds to time elapsing on valuations.

Definition 2.4 (Successor)**.** Let $\boldsymbol{d}$ and $\boldsymbol{d'}$ be digital valuations. The valuation $\boldsymbol{d'}$ is the unique successor of $\boldsymbol{d}$ ($\boldsymbol{d} \vdash \boldsymbol{d'}$) if one of the following holds:

Case $\boldsymbol{d} = d_0 d_1 \dots d_n$ and $d_0 \neq \emptyset$:

$$d_0\, d_1 \dots d_n \vdash \emptyset\, d_0\, d_1 \dots d_n.$$

Case $\boldsymbol{d} = \emptyset\, d_1 \dots d_{n-1} d_n$:

$$\emptyset\, d_1 \dots d_{n-1}\, d_n \vdash d'_n\, d_1 \dots d_{n-1},$$

where d'_n satisfies the following: if $(x, k) \in d_n$,

$$\begin{cases} (x, k+1) \in d'_n & \text{if } k < \mathsf{M} - 1, \\ (x, \infty) \quad\;\; \in d'_n & \text{if } k = \mathsf{M} - 1 \text{ or } k = \infty. \end{cases}$$

We use $\vdash^*$ to denote the reflexive transitive closure of $\vdash$. ■

Example. The following is an example of the successor relation $\vdash$ with $\mathsf{M} = 2$.

$$\begin{array}{ccccccc}
\{x \mapsto 0.0;\ y \mapsto 1.3\} & \leq & \{x \mapsto 0.5;\ y \mapsto 1.8\} & \leq & \{x \mapsto 0.7;\ y \mapsto 2.0\} & \leq & \{x \mapsto 0.9;\ y \mapsto 2.2\} \\
\rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} \\
\{(x,0)\}_0\,\{(y,1)\} & \vdash & \{\}_0\,\{(x,0)\}\,\{(y,1)\} & \vdash & \{(y,\infty)\}_0\,\{(x,0)\} & \vdash & \{\}_0\,\{(y,\infty)\}\,\{(x,0)\}
\end{array}$$

We show the following diagrams that state the relation $\vdash^*$ reflects time-elapsings on concrete valuations:

$$\begin{array}{ccccccc}
\nu & \leq & \nu' & & \nu & \leq & \nu' \\
\rotatebox{-90}{$\models$} & & & \Longrightarrow & \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} \\
\boldsymbol{d} & & & & \boldsymbol{d} & \vdash^* & {}^{\exists}\boldsymbol{d'},
\end{array}
\qquad
\begin{array}{ccccccc}
\nu & & & & \nu & \leq & {}^{\exists}\nu' \\
\rotatebox{-90}{$\models$} & & & \Longrightarrow & \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} \\
\boldsymbol{d} & \vdash^* & \boldsymbol{d'} & & \boldsymbol{d} & \vdash^* & \boldsymbol{d'}.
\end{array}$$

To this end, we define the three auxiliary time elapsing relation $<_1$, $<_2$, and $<_3$ on clock valuations. We need the notation $\mathit{max\_fract}(\nu)$ defined as follows to denote the maximal fractional part of a valuation ν:

$$\mathit{max\_fract}(\nu) \triangleq \max\left\{\mathit{frac}(\nu(x)) : x \in \mathcal{X}\right\}.$$

Let ν and ν' be clock valuations.

$$\nu <_1 \nu' \quad \overset{\text{def}}{\iff} \quad \begin{cases} \text{there is no clock } x \text{ such that } \mathit{frac}(\nu(x)) = 0.0 \text{ and} \\ \nu' = \nu + (1.0 - \mathit{max\_fract}(\nu)). \end{cases}$$

$$\nu <_2 \nu' \quad \overset{\text{def}}{\iff} \quad \begin{cases} \text{there is no clock } x \text{ such that } \mathit{frac}(\nu(x)) = 0.0 \text{ and} \\ \nu' < \nu + (1.0 - \mathit{max\_fract}(\nu)). \end{cases}$$

$$\nu <_3 \nu' \quad \overset{\text{def}}{\iff} \quad \begin{cases} \text{there is a clock } x \text{ such that } \mathit{frac}(\nu(x)) = 0.0 \text{ and} \\ \nu' < \nu + (1.0 - \mathit{max\_fract}(\nu)). \end{cases}$$

For example,

$$\begin{aligned}
&\{x \mapsto 0.2;\ y \mapsto 1.6;\ z \mapsto 2.9\} <_1 \{x \mapsto 0.3;\ y \mapsto 1.7;\ z \mapsto 3.0\}\,, \\
&\{x \mapsto 0.2;\ y \mapsto 1.6;\ z \mapsto 2.9\} <_2 \{x \mapsto 0.28;\ y \mapsto 1.68;\ z \mapsto 2.98\}\,, \\
&\{x \mapsto 0.3;\ y \mapsto 1.7;\ z \mapsto 3.0\} <_3 \{x \mapsto 0.4;\ y \mapsto 1.8;\ z \mapsto 3.1\}\,.
\end{aligned}$$

Proposition 2.7. Let ν and ν' be valuations and $\boldsymbol{d}$ be a digital valuation. The following diagram holds for $i = \{1, 2, 3\}$:

$$\begin{array}{ccc} \nu & <_i & \nu' \\ ⫪ & & \\ \boldsymbol{d} & & \end{array} \quad \Longrightarrow \quad \begin{array}{ccc} \nu & <_i & \nu' \\ ⫪ & & ⫪ \\ \boldsymbol{d} & \vdash^* & \boldsymbol{d'}. \end{array}$$

Proof. All the cases are trivial from the definition of $<_i$ and $\vdash$. ☐

Let ν and ν' be valuations such that $\nu < \nu'$. We can decompose $\nu < \nu'$ by using the relations $<_1$, $<_2$, and $<_3$. Let us consider the following example:

$$(\nu =)\, \{x \mapsto 0.2;\ y \mapsto 1.6;\ z \mapsto 2.9\} < \{x \mapsto 1.0;\ y \mapsto 2.4;\ z \mapsto 3.7\}\, (= \nu').$$

We have the following decomposition:

$$\begin{aligned}
\nu &= \{x \mapsto 0.2;\ y \mapsto 1.6;\ z \mapsto 2.9\} <_1 \\
\nu_1 &= \{x \mapsto 0.3;\ y \mapsto 1.7;\ z \mapsto 3.0\} <_3 \\
\nu_2 &= \{x \mapsto 0.4;\ y \mapsto 1.8;\ z \mapsto 3.1\} <_1 \\
\nu_3 &= \{x \mapsto 0.6;\ y \mapsto 2.0;\ z \mapsto 3.3\} <_3 \\
\nu_4 &= \{x \mapsto 0.9;\ y \mapsto 2.3;\ z \mapsto 3.6\} <_1 \\
\nu' &= \{x \mapsto 1.0;\ y \mapsto 2.4;\ z \mapsto 3.7\}\,.
\end{aligned}$$

Formally, the following property holds.

Proposition 2.8. Let ν and ν' be valuations such that $\nu < \nu'$. We can decompose $\nu < \nu'$ as follows:

$$\nu <_{i_1} \nu_1 <_{i_2} \nu_2 <_{i_3} \cdots <_{i_n} \nu'$$

where $i_1, i_2, \ldots, i_n \in \{1, 2, 3\}$.

Proof. It suffices to consider the case $\nu' = \nu + \delta$ where $0 < \delta < 1$.

We use the following function.

$$\mathit{Measure}(\nu, \delta) \triangleq |\, \{x \in \mathcal{X} : \lfloor \nu(x) \rfloor \neq \lfloor \nu(x) + \delta \rfloor\} \,|.$$

The following properties for *Measure* can be easily verified:

- $\nu <_1 \nu + \delta$ or $\nu <_1 {}^{\exists}\nu' < \nu + \delta$ if and only if $\mathit{Measure}(\nu, \delta) > 0$.
- $\nu <_2 \nu + \delta$ or $\nu <_3 \nu + \delta$ if and only if $\mathit{Measure}(\nu, \delta) = 0$.
- If $0 \leq \delta_1 + \delta_2 < 1$, then $\mathit{Measure}(\nu, \delta_1 + \delta_2) = \mathit{Measure}(\nu, \delta_1) + \mathit{Measure}(\nu + \delta_1, \delta_2)$.

We proceed by induction on $\mathit{Measure}(\nu, \nu' - \nu)$.

Base Case. We consider the case $\mathit{Measure}(\nu, \nu' - \nu) = 0$. Since there are two subcases $\nu <_2 \nu'$ or $\nu <_3 \nu'$, this case is finished.

Induction Case. We consider the case $\mathit{Measure}(\nu, \nu' - \nu) > 0$. For this case, we have the following two subcases:

$$\nu <_1 \nu', \quad \exists \nu''.\ \nu <_1 \nu'' < \nu'.$$

Now we consider the latter case. Since

- $\mathit{Measure}(\nu, \nu' - \nu) = \mathit{Measure}(\nu, \nu'' - \nu) + \mathit{Measure}(\nu'', \nu' - \nu'')$; and
- $\mathit{Measure}(\nu, \nu'' - \nu) > 0$,

we have $\mathit{Measure}(\nu', \nu' - \nu'') < \mathit{Measure}(\nu, \nu' - \nu)$. By the induction hypothesis, we have the following:

$$\nu'' <_{j_1} \nu_1 <_{j_2} < \nu_2 <_{j_3} \cdots <_{j_m} \nu'.$$

Therefore, $\nu <_1 \nu'' <_{j_1} \nu_1 <_{j_2} < \nu_2 <_{j_3} \cdots <_{j_m} \nu'$ and it finishes the proof. □

We can easily show the following proposition from Proposition 2.7 and 2.8.

Proposition 2.9. Let ν be a valuation and $\boldsymbol{d}$ be a digital valuation.

$$\begin{array}{ccc} \nu & \leq & \nu' \\ \rotatebox[origin=c]{-90}{$\models$} & & \\ \boldsymbol{d} & & \end{array} \quad \Longrightarrow \quad \begin{array}{ccc} \nu & \leq & \nu' \\ \rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} \\ \boldsymbol{d} & \vdash^* & \exists \boldsymbol{d'}. \end{array}$$

Proof. If $\nu = \nu'$, then it suffices to take $\boldsymbol{d'} = \boldsymbol{d}$. Otherwise, we decompose $\nu < \nu'$ by Proposition 2.8 and then repeatedly apply Proposition 2.7. □

The following similar propositions also hold.

Proposition 2.10. Let ν be a valuation and $\boldsymbol{d}$ be a digital valuation.

$$\begin{array}{ccc} \nu & & \\ \rotatebox[origin=c]{-90}{$\models$} & & \\ \boldsymbol{d} & \vdash^* & \boldsymbol{d'} \end{array} \quad \Longrightarrow \quad \begin{array}{ccc} \nu & \leq & \exists \nu' \\ \rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} \\ \boldsymbol{d} & \vdash^* & \boldsymbol{d'}. \end{array}$$

Proof. It suffices to show the following 1-step diagram holds:

$$\begin{array}{ccc} \nu & & \\ \rotatebox[origin=c]{-90}{$\models$} & & \\ \boldsymbol{d} & \vdash & \boldsymbol{d'} \end{array} \quad \Longrightarrow \quad \begin{array}{ccc} \nu & \leq & \exists \nu' \\ \rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} \\ \boldsymbol{d} & \vdash & \boldsymbol{d'}. \end{array}$$

In order to show this diagram, we do case analysis on $\boldsymbol{d} \vdash \boldsymbol{d'}$.

If $\boldsymbol{d} = \{\cdots\}_0\, d_1 \ldots d_n \vdash \boldsymbol{d'} = \{\}_0\ d_0\, d_1 \ldots d_n$:

Since this corresponds to $<_1$, there is a valuation ν' such that $\nu <_1 \nu'$. It is clear that $\nu' \models \boldsymbol{d'}$.

If $\boldsymbol{d} = \{\}_0\, d_1 \ldots d_{n-1} d_n \vdash \boldsymbol{d'} = \{\cdots\}_0\, d_1 \ldots d_{n-1}$: Since this corresponds to $<_3$, we take ν' such that $\nu <_3 \nu'$. It is clear that $\nu' \models \boldsymbol{d'}$.

□

Proposition 2.6, 2.9, and 2.10 mean that we can use digital valuations to simulate clock valuations. In the following subsection, we will define digital automata to simulate timed automata and show the reachability problem of timed automata is decidable due to the finiteness of digital automata.

2.3.2 Digital Automata and Decidability of Reachability Problem

We define digital automata D_A from timed automata A and show the decidability of the location reachability problem of timed automata.

A digital automaton D_A defined from the timed automaton A is a 5-tuple:

$$D_A = (Q \times \mathbb{D}, \langle q_{\text{init}}, \mathcal{D}(\mathbf{0})\rangle, F \times \mathbb{D}, \Sigma, \partial).$$

where

- $Q \times \mathbb{D}$ is the set of states (each state $\langle q, \boldsymbol{d}\rangle$ is a pair of location $q \in Q$ and digital valuation $\boldsymbol{d}$ corresponding a clock valuation of A),
- $\langle q_{\text{init}}, \mathcal{D}(\mathbf{0})\rangle$ is the initial state that corresponds to the initial configuration $\langle q_{\text{init}}, \mathbf{0}\rangle$ of A,
- $F \times \mathbb{D}$ is accepting states that correspond to configuration of A whose location is in q_F,
- Σ is the input alphabet, and
- $\partial \subseteq (Q \times \mathbb{D}) \times \Sigma_\epsilon \times Act_{\text{digi}} \times (Q \times \mathbb{D})$ is a finite set of transitions where the actions Act_{digi} is defined as follows:

$$\tau \in Act_{\text{digi}} ::= \mathsf{reset}(x) \mid x \in_? I \mid \mathsf{evolve}$$

The set of transitions ∂ is defined as follows from Δ:

$$\frac{p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta}{\langle p, \boldsymbol{d}\rangle \xrightarrow[\alpha]{\mathsf{reset}(x)} \langle q, \boldsymbol{d}[x := 0]\rangle \in \partial,} \qquad \frac{p \xrightarrow[\alpha]{x \in_? I} q \in \Delta \quad \boldsymbol{d} \models x \in I}{\langle p, \boldsymbol{d}\rangle \xrightarrow[\alpha]{x \in_? I} \langle q, \boldsymbol{d}\rangle \in \partial,} \qquad \frac{q \in Q \quad \boldsymbol{d} \in \mathbb{D} \quad \boldsymbol{d} \vdash^* \boldsymbol{d'}}{\langle q, \boldsymbol{d}\rangle \xrightarrow[\epsilon]{\mathsf{evolve}} \langle q, \boldsymbol{d'}\rangle \in \partial.}$$

Since digital automata are finite automata, we can define the language $L(D_A) \subseteq \Sigma^*$ for the digital automaton D_A as follows:

$$L(D_A) \triangleq \left\{\alpha_1\alpha_2\cdots\alpha_n : \langle q_{\text{init}}, \mathcal{D}(\mathbf{0})\rangle \xrightarrow[\alpha_1]{\tau_1} s_1 \xrightarrow[\alpha_2]{\tau_2} \cdots \xrightarrow[\alpha_n]{\tau_n} \langle q_F, \boldsymbol{d}\rangle \in F \times \mathbb{D}\right\}.$$

In order to show that the digital automaton D_A simulates the timed automaton and vice versa, we define a correspondence relation between timed and digital automata.

Definition 2.5. Let A be a timed automaton and D_A be the corresponding digital automaton. For a configuration $\langle q, \nu\rangle$ of A and $\langle p, \boldsymbol{d}\rangle$ of D_A, we write $\langle q, \nu\rangle \sim \langle p, \boldsymbol{d}\rangle$ if the following holds:

$$q = p \qquad \text{and} \qquad \nu \models \boldsymbol{d}.$$

■

We show the digital automaton D_A simulates the timed automaton A. First, we consider the case of discrete transitions.

Lemma 2.1. Let $\langle p, \nu\rangle$ and $\langle p, \boldsymbol{d}\rangle$ be configurations such that $\langle p, \nu\rangle \sim \langle p, \boldsymbol{d}\rangle$:

$$\begin{array}{c} \langle p, \nu\rangle \xrightarrow[\alpha]{\tau} \langle q, \nu'\rangle \\ \wr \\ \langle p, \boldsymbol{d}\rangle \end{array} \quad \Longrightarrow \quad \begin{array}{ccc} \langle p, \nu\rangle & \xrightarrow[\alpha]{\tau} & \langle q, \nu'\rangle \\ \wr & & \wr \\ \langle p, \boldsymbol{d}\rangle & \xrightarrow[\alpha]{\tau'} & \exists\langle q, \boldsymbol{d'}\rangle. \end{array}$$

Proof. We proceed by case analysis on $\langle p, \nu\rangle \xrightarrow[\alpha]{\tau} \langle q, \nu'\rangle$.

Case $\langle p,\nu\rangle \xrightarrow[\alpha]{\mathsf{reset}(x)} \langle q,\nu[x := 0]\rangle$: This means that there is a rule $p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta$ and therefore we have a rule $\langle p,\boldsymbol{d}\rangle \xrightarrow[\alpha]{\mathsf{reset}(x)} \langle q,\boldsymbol{d}[x := 0]\rangle \in \partial$. Proposition 2.6 implies $\nu[x := 0] \models \boldsymbol{d}[x := 0]$ because $\nu \models \boldsymbol{d}$.

Case $\langle p,\nu\rangle \xrightarrow[\alpha]{x\in_? I} \langle q,\nu\rangle$: This means that there is a rule $p \xrightarrow[\alpha]{x\in_? I} q \in \Delta$. Since $\nu \models \boldsymbol{d}$ and Proposition 2.6 implies $\boldsymbol{d} \models x \in I$, we have a rule $\langle p,\boldsymbol{d}\rangle \xrightarrow[\alpha]{x\in_? I} \langle q,\boldsymbol{d}\rangle$. □

Next, we consider the case of timed transitions.

Lemma 2.2. Let $\langle p,\nu\rangle$ and $\langle p,\boldsymbol{d}\rangle$ be configurations such that $\langle p,\nu\rangle \sim \langle p,\boldsymbol{d}\rangle$:

$$\begin{tikzcd}
\langle p,\nu\rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,\nu'\rangle & & \langle p,\nu\rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle p,\boldsymbol{d}\rangle & \arrow[r, phantom, "\Longrightarrow"] & & \langle p,\boldsymbol{d}\rangle \arrow[r, "\mathsf{evolve}", "\epsilon"'] & \langle p,{}^{\exists}\boldsymbol{d'}\rangle.
\end{tikzcd}$$

Proof. Since $\nu \models \boldsymbol{d}$, Proposition 2.10 implies that there is a digital valuation $\boldsymbol{d'}$ such that $\boldsymbol{d} \vdash^* \boldsymbol{d'}$ and $\nu + \delta \models \boldsymbol{d'}$. By the definition of digital automata, we have $\langle p,\boldsymbol{d}\rangle \xrightarrow[\epsilon]{\mathsf{evolve}} \langle p,\boldsymbol{d'}\rangle \in \partial$. □

Conversely, we show the timed automaton A simulates the digital automaton D_A.

Lemma 2.3. Let $\langle q,\nu'\rangle$ and $\langle q,\boldsymbol{d'}\rangle$ be configurations such that $\langle q,\nu'\rangle \sim \langle q,\boldsymbol{d'}\rangle$.

$$(1):\quad \begin{tikzcd}
 & \langle q,\nu\rangle \arrow[d, phantom, "\wr"] & & \langle p,\nu\rangle \arrow[r, "\tau", "\alpha"'] \arrow[d, phantom, "\wr"] & \langle q,{}^{\exists}\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle p,\boldsymbol{d}\rangle \arrow[r, "\tau", "\alpha"'] & \langle q,\boldsymbol{d'}\rangle & \Longrightarrow & \langle p,\boldsymbol{d}\rangle \arrow[r, "\tau", "\alpha"'] & \langle q,\boldsymbol{d'}\rangle.
\end{tikzcd}$$

$$(2):\quad \begin{tikzcd}
\langle p,\nu\rangle \arrow[d, phantom, "\wr"] & & & \langle p,\nu\rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,{}^{\exists}\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle p,\boldsymbol{d}\rangle \arrow[r, "\mathsf{evolve}", "\epsilon"'] & \langle p,\boldsymbol{d'}\rangle & \Longrightarrow & \langle p,\boldsymbol{d}\rangle \arrow[r, "\mathsf{evolve}", "\epsilon"'] & \langle p,\boldsymbol{d'}\rangle.
\end{tikzcd}$$

Proof. This lemma is shown by the same argument of Lemma 2.1 and 2.2. □

Combining these lemmas, we can reduce the location reachability problem of a timed automaton to the reachability problem of the corresponding digital automaton.

Theorem 2.1. Let A be a timed automaton and q be a location of A. The following location reachability of the timed automaton A and the digital automaton D_A are equivalent:

- $\langle q_{\mathrm{init}},\mathbf{0}\rangle \Rightarrow^* \langle q,{}^{\exists}\nu\rangle$ where $\Rightarrow$ is a discrete transition or timed transition.
- $\langle q_{\mathrm{init}},\mathcal{D}(\mathbf{0})\rangle \to^* \langle q,{}^{\exists}\boldsymbol{d}\rangle$ where $\to$ is some transition.

Proof. First we show the direction ($\Rightarrow$) by induction on the length of the transition $\langle q_{\mathrm{init}},\mathbf{0}\rangle \Rightarrow^* \langle q,\nu\rangle$.
If $\langle q_{\mathrm{init}},\mathbf{0}\rangle = \langle q,\nu\rangle$, then the transition $\langle q_{\mathrm{init}},\mathcal{D}(\mathbf{0})\rangle \to^* \langle q_{\mathrm{init}},\mathcal{D}(\mathbf{0})\rangle$ suffices.
If $\langle q_{\mathrm{init}},\mathbf{0}\rangle \Rightarrow^* \langle p,\nu'\rangle \Rightarrow \langle q,\nu\rangle$, then we have $\langle q_{\mathrm{init}},\mathcal{D}(\mathbf{0})\rangle \Rightarrow^* \langle p,\boldsymbol{d'}\rangle$ by the induction hypothesis.

The diagram $\begin{tikzcd} \langle p,\nu'\rangle \arrow[r] \arrow[d, phantom, "\wr"] & \langle q,\nu\rangle \\ \langle p,\boldsymbol{d'}\rangle & \end{tikzcd}$, Lemma 2.1, and Lemma 2.2 imply the following one

$$\begin{array}{ccccc}\langle q_{\mathrm{init}}, \mathbf{0}\rangle & \Rightarrow^* & \langle p, \nu'\rangle & \Rightarrow & \langle q, \nu\rangle \\ \wr & & \wr & & \wr \\ \langle q_{\mathrm{init}}, \mathcal{D}(\mathbf{0})\rangle & \rightarrow^* & \langle p, \boldsymbol{d'}\rangle & \Rightarrow & \langle q, \exists \boldsymbol{d}\rangle.\end{array}$$

The direction ($\Leftarrow$) is also shown by induction on the length of $\langle q_{\mathrm{init}}, \mathcal{D}(\mathbf{0})\rangle \rightarrow^* \langle q, \boldsymbol{d}\rangle$ with Lemma 2.3. □

This theorem states that the location reachability problem of timed automata can be reduced to the state reachability problem of finite automata.

Corollary 2.1. Let A be a timed automaton and q be a location of A.
The location reachability problem $\langle q_{\mathrm{init}}, \mathbf{0}\rangle \Rightarrow^*_? \langle q, {}^{\exists}\nu\rangle$ is PSPACE-complete.

Proof. Although this result was well-known in the theory of timed automata [AD94], we show this because we will use the construction of this proof in Corollary 3.1 of Chapter 3.

The location reachability problem is in PSPACE. We focus on the size of the digital automaton D_A obtained from the timed automaton A. Since the size of digital valuations $\mathbb{D}(\mathcal{X}, \mathsf{M})$ of D_A is exponential with respect to the size of $\mathcal{X}$ and M, the size of D_A is exponential with respect to the size of A. The reachability problem of finite automata (or finite graphs) can be solved in log-space for the number of states (or nodes): hence, we can solve the corresponding reachability problem of D_A is in PSPACE for the size of A.

The location reachability problem is PSPACE-hard. We use the following PSPACE-complete problem [Koz77]:

> Let $M_1, \ldots, M_k$ be k deterministic finite automata (DFA) with a common input alphabet Σ, and let $L(M_i)$ be the language of the DFA M_i.
>
> The emptiness problem of the intersection language $\bigcap_{i=1}^{k} L(M_i) =_? \emptyset$ is PSPACE-complete.

We can translate each DFA M_i to a DFA N_i that satisfies the following conditions in polynomial time for the size of M_i.

- The numbers of states of $N_1, N_2, \ldots, N_k$ are the same.

  Therefore, there is some K such that $|Q_i| = K$ for any $1 \le i \le k$.

We denote N_i by $N_i = (Q_i, q^i_{\mathrm{init}}, F^i, \delta^i)$ where Q_i is a finite set of states, s^i_{init} is the initial state, F^i is the set of accepting states, and $\delta^i : Q_i \times \Sigma \rightarrow Q_i$ is the transition function. We assume $Q_i = \left\{q^i_1, q^i_2, \ldots, q^i_K\right\}$.

To simplify our explanation, we assume $k = 3$. However, the following argument can be easily generalized to any k. First, we consider the following (untimed) word:

$$\sigma_1\ s_1 \sharp\ t_1 \sharp'\ u_1 \quad \sigma_2\ s_2 \sharp\ t_2 \sharp'\ u_2 \quad \cdots \quad \sigma_n\ s_n \sharp\ t_n \sharp'\ u_n \quad \sigma_{n+1}\ s_{n+1} \sharp\ t_{n+1} \sharp'\ u_{n+1}$$

where it satisfies the following condition:

- $s_i \in Q_1$, $t_i \in Q_2$, and $u_i \in Q_3$ for $1 \le i \le n+1$;
- $\delta^1(q^1_{\mathrm{init}}, \sigma_1) = s_1$, $\delta^1(s_i, \sigma_{i+1}) = s_{i+1}$ for $i \in \{1, \ldots, n\}$, and $s_{n+1} \in F^1$.
- $\delta^2(q^2_{\mathrm{init}}, \sigma_1) = t_1$, $\delta^2(t_i, \sigma_{i+1}) = t_{i+1}$ for $i \in \{1, \ldots, n\}$, and $t_{n+1} \in F^2$.
- $\delta^3(q^3_{\mathrm{init}}, \sigma_1) = u_1$, $\delta^3(u_i, \sigma_{i+1}) = u_{i+1}$ for $i \in \{1, \ldots, n\}$, and $u_{n+1} \in F^3$.

Let L be the set of words that satisfy the above condition. It is clear that $L = \emptyset \iff L(N_1) \cap L(N_2) \cap L(N_3) = \emptyset$ and L is a regular language. We can construct a DFA M that accepts L by product construction in exponential time for the size of N_1, N_2, and N_3.

We construct a timed automaton A such that $\mathit{Untime}(L(A)) = L$ in polynomial time for the size of N_1, N_2, N_3 by avoiding product construction where the homomorphism $\mathit{Untime}\colon (\Sigma \times \mathbb{R}_{\geq 0})^* \to \Sigma^*$ is defined as follows:

$$\mathit{Untime}(\langle \sigma_1, r_1\rangle \langle \sigma_2, r_2\rangle \dots \langle \sigma_n, r_n\rangle) \triangleq \sigma_1 \sigma_2 \dots \sigma_n.$$

We require the following condition for each timed word $\tau \in L(A)$:

(1) $\mathit{Untime}(\tau) = \sigma_1\ s_1\ \sharp_1\ t_1\ \sharp'_1\ u_1 \quad \cdots \quad \sigma_n\ s_n\ \sharp_n\ t_n\ \sharp'_n\ u_n \in (\Sigma \cdot Q^1 \cdot \sharp \cdot Q^2 \cdot \sharp' \cdot Q^3)^*$.

- We do not require $\mathit{Untime}(\tau) \in L$.

We write $\tau(\alpha)$ to denote the time r in the timed word $\tau = \cdots \langle r, \alpha\rangle \cdots$.

(2) $\tau(\sharp_i) - \tau(\sigma_i) = K$, $\tau(\sharp'_i) - \tau(\sharp_i) = K$, and $\tau(\sigma_{i+1}) - \tau(\sharp'_i) = K$.

(3a) $\tau(s_i) - \tau(\sigma_i) = j$ if $Q^1 = \left\{q^1_1, q^1_2, \dots, q^1_K\right\}$ and $s_i = q^1_j$.

(3b) $\delta^1(q^1_{\text{init}}, \sigma_1) = s_1$, $\delta^1(s_i, \sigma_{i+1}) = s_{i+1}$ for $i \in \{1, \dots, n\}$, and $s_{n+1} \in F^1$.

(4a) $\tau(t_i) - \tau(\sharp_i) = j$ if $Q^2 = \left\{q^2_1, q^2_2, \dots, q^2_K\right\}$ and $t_i = q^2_j$.

(4b) $\delta^2(q^2_{\text{init}}, \sigma_1) = t_1$, $\delta^2(t_i, \sigma_{i+1}) = t_{i+1}$ for $i \in \{1, \dots, n\}$, and $t_{n+1} \in F^2$.

(5a) $\tau(u_i) - \tau(\sharp'_i) = j$ if $Q^2 = \left\{q^3_1, q^3_2, \dots, q^3_K\right\}$ and $u_i = q^3_j$.

(5b) $\delta^3(q^3_{\text{init}}, \sigma_1) = u_1$, $\delta^3(u_i, \sigma_{i+1}) = u_{i+1}$ for $i \in \{1, \dots, n\}$, and $u_{n+1} \in F^3$.

To check condition (1), for the general case k (k is the number of DFA), we need no clocks but $O(k)$-states and $O(\mathit{poly}(k, |\Sigma|, K))$-edges.

To check the condition (2), we use three clocks x_1, x_2, and x_3.

- When we read a symbol σ_i, we reset the clock x_1. After that, when we read the corresponding symbol $\sharp_i$, we check $x_1 \in [K : K]$.
- Similary, we use the clock x_2 for the pair $\sharp_i$ and $\sharp'_i$, and use the clock x_3 for the pair $\sharp'_i$ and σ_{i+1}.

Therefore, for the general case k, we need k clocks.

To see how we check the condition (3a) and (3b), let us consider the following timed word:

$$\tau = \ \cdots \quad \sigma_i\ s_i\ \sharp_i\ t_i\ \sharp'_i\ u_i \quad \sigma_{i+1}\ s_{i+1}\ \sharp_{i+1}\ t_{i+1}\ \sharp'_i\ u_{i+1} \cdots$$

where $\tau(\sharp_i) - \tau(\sigma_i) = K$, $\tau(\sharp'_i) - \tau(\sharp_i) = K$, and $\tau(\sigma_{i+1}) - \tau(\sharp'_i) = K$. We use a clock y to ensure $\tau(s_i) - \tau(\sigma_i) = j$ where $q^1_j = s_i$ and clock y' to restore s_i when we read the symbol σ_{i+1}. To this end, we carry out the following steps:

1. When we read the symbol σ_i, we reset a clock y.
2. When we read the symbol s_i,
   - we check whether or not $\tau(s_i) - \tau(\sigma_i) = j$ by $y \in_? [j : j]$; and
   - we reset the clock y'.
3. When we read the symbol σ_{i+1}, we restore the state s_i based on the following properties:

- The condition (2) implies the value of y' is in $\{2K, 2K+1, \ldots, 3K-1\}$.
- Furthermore, if $y' = 3K - \ell$, then $s_i = q^1_\ell$ holds.

Therefore, we can check whether or not $s_{i+1} = \delta^1(s_i, \sigma_{i+1})$.

For the general case k (k is the number of DFA), the construction to check the conditions corresponding to (3a)–(5b) is accomplished with $O(k)$-clocks, $O(\mathit{poly}(k, |\Sigma|, K))$-states, and $O(\mathit{poly}(k, |\Sigma|, K))$-edges.

Following the above argument to avoid an exponential blowup, we obtain a timed automaton A in polynomial time for the size of $N_1, N_2, \ldots, N_k$. Therefore, the location reachability problem of timed automata is PSPACE-complete. □

Recently, Fearnley and Jurdziński showed the location reachability problem of two-clocks timed automata is already PSPACE-complete [FJ15]. Laroussinie, Markey, and Schnoebelen showed that location reachability problem of one-clock timed automata is NLOGSPACE-complete [LMS04].

Regularity of Timed Automata Since the digital automaton D_A obtained from a timed automaton A faithfully simulates computations of A, we can show the regularity of timed automata. The regularity of timed automata is that if we forget the time-stamps from the language $L(A)$ of a timed automaton A, then the untimed language is regular. To state this formally, we consider the projection $\mathit{Untime} : (\Sigma \times \mathbb{R}_{\geq 0})^* \to \Sigma^*$ from timed words to (untimed) words. This function was already used in the proof of Corollary 2.1. The regularity of timed automata can be stated as: for any timed automaton A, $\mathit{Untime}(L(A))$ is regular. This is shown by the following theorem.

Theorem 2.2. *Let A be a timed automaton and D_A be the digital automaton corresponding to A. The untimed language $\mathit{Untime}(L(A))$ equals to the language of D_A: $\mathit{Untime}(L(A)) = L(D_A)$.*

Proof. This is shown by Lemma 2.1, 2.2, and 2.3 because we preserve labels in these lemmas.

First, we show that if $w^t \in L(A)$ then $\mathit{Untime}(w^t) \in L(D_A)$. We assume the following:

$$w^t = \mathsf{tw}\left(\langle q_{\mathrm{init}}, \mathbf{0}\rangle \overset{\delta_1}{\rightsquigarrow} \boldsymbol{c}_1 \xrightarrow[\alpha_1]{\tau_1} \boldsymbol{c}'_1 \overset{\delta_2}{\rightsquigarrow} \boldsymbol{c}_2 \xrightarrow[\alpha_2]{\tau_2} \boldsymbol{c}'_2 \overset{\delta_3}{\rightsquigarrow} \cdots \xrightarrow[\alpha_n]{\tau_n} \langle q_f, \nu\rangle\right).$$

It suffices to show $\mathit{Untime}(w^t) = \alpha_1\alpha_2\cdots\alpha_n \in L(D_A)$. Applying Lemma 2.1 and 2.2, we have the following computation in D_A:

$$\langle q_{\mathrm{init}}, \mathcal{D}(\mathbf{0})\rangle \xrightarrow[\epsilon]{\mathsf{evolve}} s_1 \xrightarrow[\alpha_1]{\tau_1} s'_1 \xrightarrow[\epsilon]{\mathsf{evolve}} s_2 \xrightarrow[\alpha_2]{\tau_2} s'_2 \xrightarrow[\epsilon]{\mathsf{evolve}} \cdots\cdots \xrightarrow[\alpha_n]{\tau_n} \langle q_F, \boldsymbol{d}\rangle$$

where $\boldsymbol{c}_i \models s_i$ and $\boldsymbol{c}'_i \models s'_i$. Since $q_F \in F$, we have $\alpha_1\alpha_2\cdots\alpha_n \in L(D_A)$.

Next, we show that if $w \in L(D_A)$ then there is a timed word w^t such that $w^t \in L(A)$ and $\mathit{Untime}(w^t) = w$. This is shown by the similar argument by using Lemma 2.3. □

2.4 Backward Simulation and Decidability of Configuration Reachability Problem

We consider the configuration reachability problem that decides whether or not a timed automaton A can reach a designated configuration $\langle q, \nu\rangle$ rather than a location:

$$\langle q_{\mathrm{init}}, \mathbf{0}\rangle \Rightarrow^*_? \langle q, \nu\rangle.$$

As with the decidability proof of the location reachability problem, we reduce the configuration reachability problem on timed automata to the configuration reachability problem on digital automata.

We can show the following property based on the previous section.

Lemma 2.4.

$$\langle q_{\text{init}}, \mathbf{0} \rangle \Rightarrow^{*} \langle q, \nu \rangle \implies \langle q_{\text{init}}, \mathcal{D}(\mathbf{0}) \rangle \rightarrow^{*} \langle q, \mathcal{D}(\nu) \rangle.$$

Proof. This can be shown in the same way as the ($\Longrightarrow$)-part of Theorem 2.1. $\square$

However, we cannot show the following directly:

$$\langle q_{\text{init}}, \mathcal{D}(\mathbf{0}) \rangle \rightarrow^{*} \langle q, \mathcal{D}(\nu) \rangle \implies \langle q_{\text{init}}, \mathbf{0} \rangle \Rightarrow^{*} \langle q, \nu \rangle.$$

We will see why this does not hold in the following two subsections.

2.4.1 Redefine the Constant M

For the configuration reachability problem $\langle q_{\text{init}}, \mathbf{0} \rangle \Rightarrow^{*} \langle q, \nu \rangle$, we need to redefine M by involving constants in the target configuration $\langle q, \nu \rangle$. Let us explain why we should redefine M by considering the following timed automata:

$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[r, "x \in_? [2:2]", "a"'] & q_1 \arrow[r, "\mathsf{reset}(x)", "b"'] & q_2 \arrow[r, "y \in_? (5:\omega)", "c"'] & q_3
\end{tikzcd}$$

We have $\mathsf{M} = 6$ by the current definition:

$$\mathsf{M} \triangleq \max\{\ i, j \in \mathbb{N} : (i : j),\ [i : j], \text{ or } (i : \omega) \text{ appears in } \Delta\} + 1.$$

Let us consider the following instance of the configuration reachability problem:

$$\langle q_0, \{x \mapsto 0.0;\ y \mapsto 0.0\} \rangle \Rightarrow^{*}_{?} \langle q_3, \{x \mapsto 10.0;\ y \mapsto 11.0\} \rangle.$$

This does not hold because, for any configuration $\langle q_3, \nu \rangle$ that is reachable from $\langle q_0, \{x \mapsto 0;\ y \mapsto 0\} \rangle$, $\nu(y) - \nu(x) \geq 2$ holds.

We translate the query to the following one of the digital automaton D_A:

$$\langle q_0, \{(x, 0),\ (y, 0)\}_0 \rangle \rightarrow^{*}_{?} \langle q_3, \{(x, \infty),\ (y, \infty)\}_0 \rangle.$$

Although we cannot reach $\langle q_3, \{x \mapsto 10.0;\ y \mapsto 11.0\} \rangle$, we can reach $\langle q_3, \{(x, \mathsf{M}),\ (y, \mathsf{M})\}_0 \rangle$ with the following witness:

$$\begin{aligned}
&\langle q_0,\ \{(x,0),\ (y,0)\}_0 \rangle \xrightarrow[\epsilon]{\mathsf{evolve}} \langle q_0,\ \{(x,2),\ (y,2)\}_0 \rangle \xrightarrow[a]{x \in_? [2:2]} \\
&\langle q_1,\ \{(x,2),\ (y,2)\}_0 \rangle \xrightarrow[\epsilon]{\mathsf{evolve}} \langle q_1,\ \{(x,2),\ (y,2)\}_0 \rangle \xrightarrow[b]{\mathsf{reset}(y)} \\
&\langle q_2,\ \{(x,0),\ (y,2)\}_0 \rangle \xrightarrow[\epsilon]{\mathsf{evolve}} \langle q_2,\ \{(x,\infty),\ (y,\infty)\}_0 \rangle \xrightarrow[c]{y \in_? (5:\omega)} \langle q_3,\ \{(x,\infty),\ (y,\infty)\}_0 \rangle.
\end{aligned}$$

To overcome this problem, it suffices to take a sufficiently large M with respect to a given instance of the configuration reachability problem. To formalize this, let us fix one instance $\langle q_{\text{init}}, \mathbf{0} \rangle \Rightarrow^{*}_{?} \langle q, \nu \rangle$. We use $\max(\nu)$ to denote the maximum value of the real numbers in a valuation ν defined as follows:

$$\max(\nu) \triangleq \max\{\nu(x) : x \in \mathcal{X}\}.$$

We take an upper-bound constant $\mathsf{M} \in \mathbb{N}$ so that it satisfies the following:

$$\mathsf{M} > \max\{\ i, j : (i : j),\ [i : j], \text{ or } (i : \omega) \text{ appears in } \Delta\} \quad \text{and} \quad \mathsf{M} > \max(\nu).$$

Now we have the following query and it does not hold on the digital automaton D_A:

$$\langle q_0, \{(x, 0),\ (y, 0)\}_0 \rangle \rightarrow^{*}_{?} \langle q_3, \{(x, 10),\ (y, 11)\}_0 \rangle.$$

2.4.2 Forward Simulation is not Enough

Again let us consider the following timed automaton:

$$\begin{tikzcd}
\arrow[r] & q_0 \arrow[r, "x \in_? [2:2]", "a"'] & q_1 \arrow[r, "\mathsf{reset}(x)", "b"'] & q_2 \arrow[r, "y \in_? (5:\omega)", "c"'] & q_3
\end{tikzcd}$$

and we consider the following reachability query:

$$\langle q_0, \{x \mapsto 0.0; y \mapsto 0.0\}\rangle \Rightarrow_?^* \langle q_3, \{x \mapsto 5.0; y \mapsto 7.5\}\rangle.$$

This holds due to the following witness:

$$\begin{aligned}
&\langle q_0, \{x \mapsto 0.0; y \mapsto 0.0\}\rangle \overset{2.0}{\rightsquigarrow} \langle q_0, \{x \mapsto 2.0; y \mapsto 2.0\}\rangle \xrightarrow[a]{x \in_? [2:2]} \\
&\langle q_1, \{x \mapsto 2.0; y \mapsto 2.0\}\rangle \overset{0.5}{\rightsquigarrow} \langle q_1, \{x \mapsto 2.5; y \mapsto 2.5\}\rangle \xrightarrow[b]{\mathsf{reset}(x)} \\
&\langle q_2, \{x \mapsto 0.0; y \mapsto 2.5\}\rangle \overset{5.0}{\rightsquigarrow} \langle q_2, \{x \mapsto 5.0; y \mapsto 7.5\}\rangle \xrightarrow[c]{y \in_? (5:\omega)} \\
&\langle q_3, \{x \mapsto 5.0; y \mapsto 7.5\}\rangle.
\end{aligned}$$

We take $\mathsf{M} = 8$ to satisfy the following:

$$\mathsf{M} > \max(A) = 5, \qquad \mathsf{M} > \max(\{x \mapsto 5.0; y \mapsto 7.5\}) = 7.5.$$

We consider the corresponding query on the digital automaton:

$$\langle q_0, \{(x,0),(y,0)\}_0\rangle \rightarrow_?^* \langle q_3, \{(x,5)\}_0 \{(y,7)\}\rangle.$$

It can be easily checked that the query holds. Therefore, applying Theorem 2.1, we have $\langle q_{\mathrm{init}}, \mathbf{0}\rangle \Rightarrow^* \langle q_3, \nu\rangle$ with $\nu \models \{(x,5)\}_0 \{(y,7)\}$ on the timed automaton. Unfortunately, this does not ensure $\nu = \{x \mapsto 5.0; y \mapsto 7.5\}$ and we cannot say the original query holds on the timed automaton.

To deal with this problem, we show the following backward simulation property in this section (Lemma 2.6):

$$\begin{array}{ccccccc}
 & & \langle q, \nu'\rangle & & \langle q_{\mathrm{init}}, \mathbf{0}\rangle & \Rightarrow^* & \langle q, \nu'\rangle \\
 & & \wr & \Longrightarrow & \wr & & \wr \\
\langle q_{\mathrm{init}}, \mathcal{D}(\mathbf{0})\rangle & \rightarrow^* & \langle q, \boldsymbol{d'}\rangle & & \langle q_{\mathrm{init}}, \mathcal{D}(\mathbf{0})\rangle & \rightarrow^* & \langle q, \boldsymbol{d'}\rangle.
\end{array}$$

It should be noted that there exists a unique valuation ν such that $\nu \models \mathcal{D}(\mathbf{0})$, i.e., $\nu = \mathbf{0}$. We would like to show the following diagram to prove this theorem:

$$\begin{array}{ccccccc}
 & & \nu' & & \exists\nu & \leq & \nu' \\
 & & \rotatebox{-90}{$\models$} & \Longrightarrow & \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} \\
\boldsymbol{d} & \vdash^* & \boldsymbol{d'} & & \boldsymbol{d} & \vdash^* & \boldsymbol{d'}.
\end{array} \qquad (\#)$$

Unfortunately again, this diagram does not hold. Let us consider the following example:

$$\begin{array}{ccc}
 & & \{x \mapsto 3.0; y \mapsto 1.7\} \\
 & & \rotatebox{-90}{$\models$} \\
\{\}_0 \{(y,1)\} \{(x,1)\} & \vdash & \{(x,\infty)\}_0 \{(y,1)\}
\end{array}$$

where $\mathsf{M} = 2$. There is no valuation ν that satisfies the conditions because

- if $\nu \models \{\}_0 \{(y,1)\} \{(x,1)\}$, then $0 < \nu(x) - \nu(y) < 1$;
- However, since $\nu \leq \nu'$, $\nu(x) - \nu(y) = 1.3$.

To avoid this problem and show Lemma 2.6, we introduce *collapsed valuations* that are another abstraction of clock valuations. On collapsed valuations, we only abstract (or collapse) the integral parts of clock valuations. Recall that, on digital valuations, we abstract both the integral and fractional parts of clock valuations. We will show that the above diagram (#) holds (Proposition 2.12) between collapsed and digital valuations and show the backward simulation property (Lemma 2.6) based on the proposition.

2.4.3 Collapsed Valuations

Let $A = (Q, q_{\text{init}}, F, \Sigma, \mathcal{X}, \Delta)$ be a timed automaton, $\langle q_{\text{init}}, \mathbf{0}\rangle \Rightarrow^* \langle q, \nu\rangle$ be a configuration reachability query, and M is the constant defined from A and ν.

Definition 2.6 (Collapsed domain)**.** The set of collapsed real numbers $\mathbb{C}$ is defined as follows:

$$\mathbb{C} \triangleq \big(\{0, 1, \ldots, \mathsf{M} - 1\} \cup \{\infty\}\big) \times [0 : 1).$$

The collapsing function $\mathcal{C} : \mathbb{R}_{\geq 0} \to \mathbb{C}$ is defined as follows:

$$\mathcal{C}(r) \triangleq \begin{cases} (\infty\ , \mathit{frac}(r)) & \text{if } r \geq \mathsf{M}, \\ (\lfloor r \rfloor, \mathit{frac}(r)) & \text{if } r < \mathsf{M}. \end{cases}$$

For a concrete valuation $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$, we define the *collapsed valuation of* ν, $\mathcal{C}(\nu)$ as follows:

$$\mathcal{C}(\nu)(x) \triangleq \mathcal{C}(\nu(x)).$$

■

We write $v.r$ to denote (v, r). For example, we write 2.6 and $\infty.3$ to denote the collapsed values $(2, 0.6)$ and $(\infty, 0.3)$, respectively. Moreover, $\lfloor v.r \rfloor$ and $\mathit{frac}(v.r)$ denote v and r, respectively. We use Greek letters $\lambda, \ldots$ to denote a collapsed valuation.

The following basic properties are immediately shown by the above definition.

Proposition 2.11. Let $\nu_1, \nu_2 : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be valuations. If $\mathcal{C}(\nu_1) = \mathcal{C}(\nu_2)$,

Checking. $\nu_1 \models x \in I$ iff $\nu_2 \models x \in I$ for any interval I of the timed automaton A.

Assigning. $\mathcal{C}(\nu_1[x := r]) = \mathcal{C}(\nu_2[x := r])$ for any $x \in \mathcal{X}$ and $r \in \mathbb{R}_{\geq 0}$.

Evolving. $\mathcal{C}(\nu_1 + \delta) = \mathcal{C}(\nu_2 + \delta)$ for any $\delta \in \mathbb{R}_{\geq 0}$.

On the basis of Proposition 5.5, we define operations for collapsed valuations as follows.

Definition 2.7. Let ν and λ be concrete and collapsed valuations on $\mathcal{X}$ such that $\mathcal{C}(\nu) = \lambda$.

- For a constraint $x \in_? I$, we write $\lambda \models x \in I$ if $\nu \models x \in I$.
- For a real number $r \in \mathbb{R}_{\geq 0}$, $\lambda[x := r] \triangleq \mathcal{C}(\nu[x := r])$.
- For a real number $\delta \in \mathbb{R}_{\geq 0}$, $\lambda + \delta \triangleq \mathcal{C}(\nu + \delta)$.

■

The above definition is well-defined because Proposition 5.5 ensures that the result does not depend on the choice of a witness ν for λ.

We define a (quasi) ordering $\lambda \preccurlyeq \lambda'$ for collapsed valuations that corresponds to the ordering $\leq$ on concrete valuations.

Definition 2.8. Let λ and λ' be collapsed valuations. We write $\lambda \preccurlyeq \lambda'$ if there are two concrete valuations ν and ν' such that $\nu \leq \nu'$, $\mathcal{C}(\nu) = \lambda$, and $\mathcal{C}(\nu') = \lambda'$. ■

This quasi-ordering is not antisymmetric because $\{x \mapsto \infty.0\} \preccurlyeq \{x \mapsto \infty.5\}$ and $\{x \mapsto \infty.5\} \preccurlyeq \{x \mapsto \infty.0\}$ but these are different valuations.

Collapsed Semantics. For our fixed timed automaton $A = (Q, q_{\text{init}}, F, \Sigma, \mathcal{X}, \Delta)$, we define the collapsed semantics of A as an infinite labeled transition system $(Q \times (\mathcal{X} \to \mathbb{C}), \to, \rightsquigarrow)$ where

$$\frac{p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta}{\langle p, \lambda \rangle \xrightarrow[\alpha]{\mathsf{reset}(x)} \langle q, \lambda[x := 0] \rangle,} \qquad \frac{p \xrightarrow[\alpha]{x \in_? I} q \in \Delta \quad \lambda \models x \in I}{\langle p, \lambda \rangle \xrightarrow[\alpha]{x \in_? I} \langle q, \lambda \rangle,} \qquad \frac{\delta \in \mathbb{R}_{\geq 0}}{\langle q, \lambda \rangle \overset{\delta}{\rightsquigarrow} \langle q, \lambda + \delta \rangle.}$$

Relating the standard semantics and the collapsed semantics, we define a relation between configurations of them as follows:

$$\langle q, \nu \rangle \sim \langle p, \lambda \rangle \overset{\text{def}}{\iff} q = p \quad \text{and} \quad \mathcal{C}(\nu) = \lambda.$$

The definitions of the collapsed semantics and the relation $\sim$ leads to the following property that states the operational semantics of timed automata is simulated by the collapsed semantics, and vice versa:

Lemma 2.5.

$$(1): \begin{tikzcd} \langle p, \nu \rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p, \nu' \rangle \\ \langle p, \lambda \rangle & \end{tikzcd} \Longrightarrow \begin{tikzcd} \langle p, \nu \rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p, \nu' \rangle \arrow[d, phantom, "\wr"] \\ \langle p, \lambda \rangle \arrow[r, squiggly, "\delta"] & \langle p, {}^{\exists}\lambda' \rangle, \end{tikzcd} \qquad \begin{tikzcd} \langle p, \nu \rangle \arrow[r, "\tau"', "\alpha"] \arrow[d, phantom, "\wr"] & \langle q, \nu' \rangle \\ \langle p, \lambda \rangle & \end{tikzcd} \Longrightarrow \begin{tikzcd} \langle p, \nu \rangle \arrow[r, "\tau", "\alpha"'] \arrow[d, phantom, "\wr"] & \langle q, \nu' \rangle \arrow[d, phantom, "\wr"] \\ \langle p, \lambda \rangle \arrow[r, "\tau", "\alpha"'] & \langle q, {}^{\exists}\lambda' \rangle. \end{tikzcd}$$

$$(2): \begin{tikzcd} \langle p, \nu \rangle \arrow[d, phantom, "\wr"] & \\ \langle p, \lambda \rangle \arrow[r, squiggly, "\delta"] & \langle p, \lambda' \rangle \end{tikzcd} \Longrightarrow \begin{tikzcd} \langle p, \nu \rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p, {}^{\exists}\nu' \rangle \arrow[d, phantom, "\wr"] \\ \langle p, \lambda \rangle \arrow[r, squiggly, "\delta"] & \langle p, \lambda' \rangle, \end{tikzcd} \qquad \begin{tikzcd} \langle p, \nu \rangle \arrow[d, phantom, "\wr"] & \\ \langle p, \lambda \rangle \arrow[r, "\tau", "\alpha"'] & \langle q, \lambda' \rangle \end{tikzcd} \Longrightarrow \begin{tikzcd} \langle p, \nu \rangle \arrow[r, "\tau", "\alpha"'] \arrow[d, phantom, "\wr"] & \langle q, {}^{\exists}\nu' \rangle \arrow[d, phantom, "\wr"] \\ \langle p, \lambda \rangle \arrow[r, "\tau", "\alpha"'] & \langle q, \lambda' \rangle. \end{tikzcd}$$

Proof. This is immediate from the definition of the collapsed semantics. □

We use the property between the collapsed semantics and digital automata that corresponds to the diagram (#) in Section 2.4.2. First, we define the realization relation between collapsed and digital valuations.

Definition 2.9. Let $\lambda : \mathcal{X} \to \mathbb{C}$ be a collapsed valuation on $\mathcal{X}$, and $\boldsymbol{d} = d_0 d_1 \ldots d_n$ be a digital valuation on $\mathcal{X}$. We write $\lambda \models \boldsymbol{d}$ if the following hold:

- For all $x \in \mathcal{X}$, $(x, \lfloor \lambda(x) \rfloor) \in \boldsymbol{d}$.
- For all $x \in \mathcal{X}$, $\mathit{frac}(\lambda(x)) = 0.0$ iff $x \in d_0$.
- $\mathit{frac}(\lambda(x)) < \mathit{frac}(\lambda(y))$ iff $x \in d_i$ and $y \in d_j$ for some $i < j$.

■

Proposition 2.12. Let λ be a collapsed valuation and $\boldsymbol{d}$ be a digital valuation.

$$\begin{array}{ccc} & & \lambda' \\ & & \rotatebox{90}{$\models$} \\ \boldsymbol{d} & \vdash^* & \boldsymbol{d'} \end{array} \Longrightarrow \begin{array}{ccc} \exists\lambda & \leq & \lambda' \\ \rotatebox{90}{$\models$} & & \rotatebox{90}{$\models$} \\ \boldsymbol{d} & \vdash^* & \boldsymbol{d'}. \end{array}$$

Proof. It suffices to show the following 1-step diagram holds:

$$\begin{array}{ccc} & & \lambda' \\ & & \rotatebox{90}{$\models$} \\ \boldsymbol{d} & \vdash & \boldsymbol{d'} \end{array} \Longrightarrow \begin{array}{ccc} \exists\lambda & \leq & \lambda' \\ \rotatebox{90}{$\models$} & & \rotatebox{90}{$\models$} \\ \boldsymbol{d} & \vdash & \boldsymbol{d'}. \end{array}$$

To prove this, we reuse the ordering $<_1$ and $<_3$ which were defined as follows:

$$\nu <_1 \nu' \quad \overset{\text{def}}{\iff} \quad \begin{cases} \text{there is no clock } x \text{ such that } \mathit{frac}(\nu(x)) = 0.0 \text{ and} \\ \nu' = \nu + (1.0 - \mathit{max\_fract}(\nu)), \end{cases}$$

$$\nu <_3 \nu' \quad \overset{\text{def}}{\iff} \quad \begin{cases} \text{there is a clock } x \text{ such that } \mathit{frac}(\nu(x)) = 0.0 \text{ and} \\ \nu' < \nu + (1.0 - \mathit{max\_fract}(\nu)) \end{cases}$$

where $\mathit{max\_fract}(\nu)$ denotes the maximal fractional part of a valuation ν:

$$\mathit{max\_fract}(\nu) \triangleq \max \{\mathit{frac}(\nu(x)) : x \in \mathcal{X}\}.$$

In order to show this diagram, we proceed by analysis on $\boldsymbol{d} \vdash \boldsymbol{d'}$.

Case $\boldsymbol{d} = \{\cdots\}_0\, d_1 \ldots d_n \vdash \boldsymbol{d'} = \{\}_0\; d_0\, d_1 \ldots d_n$:
There is a valuation ν' such that $\nu' \models \lambda' \models \boldsymbol{d'}$. Furthermore, there is a valuation ν such that $\nu <_3 \nu'$. It suffices to show $\nu \models \boldsymbol{d}$ and it is shown as follows:

- The following two condition is easily verified from $\nu' \models \boldsymbol{d'}$ and $\nu <_3 \nu'$:
  - For a clock $x \in \mathcal{X}$, $\mathit{frac}(\nu(x)) = 0.0$ iff $x \in d_0$.
  - $\mathit{frac}(\nu(x)) < \mathit{frac}(\nu(y))$ iff $x \in d_i$ and $y \in d_j$ for some $i < j$.
- Since $\lfloor \nu'(x) \rfloor = \lfloor \nu(x) \rfloor$ for any $x \in \mathcal{X}$ and $\nu' \models \boldsymbol{d'}$, the following conditions are easily verified:
  - $\nu(x) \geq \mathsf{M}$ iff $(x, \infty) \in \boldsymbol{d}$.
  - If $\nu(x) < \mathsf{M}$, then $(x, \lfloor \nu(x) \rfloor) \in \boldsymbol{d}$.
  - If $(x, k) \in \boldsymbol{d}$ with $k < \mathsf{M}$, then $\lfloor \nu(x) \rfloor = n$.

Case $\boldsymbol{d} = \{\}_0\, d_1 \ldots d_{n-1} d_n \vdash \boldsymbol{d'} = \{\cdots\}_0\, d_1 \ldots d_{n-1}$:
There is a valuation ν' such that $\nu' \models \lambda' \models \boldsymbol{d'}$. On the basis of ν', we define a valuation θ' as follows:

$$\theta'(x) \triangleq \begin{cases} \mathsf{M} & \text{if } (x, \mathsf{M} - 1) \in d_n, \\ \mathsf{M} + 1 & \text{if } (x, \infty) \in d_n, \\ \nu'(x) & \text{otherwise.} \end{cases}$$

It can be easily verified that $\theta' \models \lambda' \models \boldsymbol{d'}$ and there is a valuation θ such that $\theta <_1 \theta'$.
It suffices to show $\theta \models \boldsymbol{d}$.
The following condition is easily verified from $\theta' \models \boldsymbol{d}$ and $\theta <_1 \theta'$:

- $\mathit{frac}(\theta(x)) < \mathit{frac}(\theta(y))$ iff $x \in d_i$ and $y \in d_j$ for some $i < j$.
- $\mathit{frac}(\theta(x)) = 0.0$ iff $x \in \{\}_0$.

To show the other conditions, we proceed by case analysis for a clock $x \in \mathcal{X}$.

Case $x \notin \{\cdots\}_0$ of $\boldsymbol{d'}$: For this case, $\theta' \models \boldsymbol{d'}$ and $\theta <_1 \theta'$ imply $\lfloor \theta(x) \rfloor = \lfloor \theta'(x) \rfloor$. Therefore, the following condition hold from $\theta' \models \boldsymbol{d'}$:

- $\theta(x) \geq \mathsf{M}$ iff $(x, \infty) \in \boldsymbol{d}$.
- If $\theta(x) < \mathsf{M}$, then $(x, \lfloor \theta(x) \rfloor) \in \boldsymbol{d}$.
- If $(x, k) \in \boldsymbol{d}$ with $k < \mathsf{M}$, then $\lfloor \theta(x) \rfloor = k$.

Case $x \in \{\cdots\}_0$ of $\boldsymbol{d'}$ (equivalently, $x \in d_n$ for $\boldsymbol{d}$): For this case, $\boldsymbol{d} \vdash \boldsymbol{d'}$ implies

$$\begin{cases} k < \mathsf{M} - 1 \wedge (x,k) \in \boldsymbol{d} \implies (x,k+1) \in \boldsymbol{d'}, \\ (x, \mathsf{M}-1) \in \boldsymbol{d} \implies (x,\infty) \in \boldsymbol{d'}, \\ (x,\infty) \in \boldsymbol{d} \implies (x,\infty) \in \boldsymbol{d'}. \end{cases}$$

Furthermore, by $\theta' \models \boldsymbol{d'}$ and $\boldsymbol{d} \vdash \boldsymbol{d'}$, the definition of θ' leads to the following:

$$\theta'(x) = \begin{cases} k+1 & \text{if } (x,k) \in d_n \text{ with } k < \mathsf{M}-1, \\ \mathsf{M} & \text{if } (x,\mathsf{M}-1) \in d_n, \\ \mathsf{M}+1 & \text{if } (x,\infty) \in d_n. \end{cases}$$

We show that if $(x,k) \in \boldsymbol{d}$ with $k \in \{0,1,\ldots,\mathsf{M}-1\}$, then $\lfloor \theta(x) \rfloor = k$.

- If $(x,k) \in \boldsymbol{d}$ with $k < \mathsf{M}-1$, then $\theta'(x) = k+1$ and $\theta <_1 \theta'$ implies $\lfloor \theta(x) \rfloor = k$.
- If $(x,\mathsf{M}-1) \in \boldsymbol{d}$, then $\theta'(x) = \mathsf{M}$. We have $\lfloor \theta(x) \rfloor = \mathsf{M}-1$ from $\theta <_1 \theta'$.

We show that if $(x,\infty) \in \boldsymbol{d}$, then $\lfloor \theta(x) \rfloor \geq \mathsf{M}$.

- Since $\theta'(x) = \mathsf{M}+1$, $\theta <_1 \theta'$ implies $\lfloor \theta(x) \rfloor = \mathsf{M}$.

We show that if $\theta(x) \geq \mathsf{M}$, then $(x,\infty) \in \boldsymbol{d}$.

- $\theta <_1 \theta'$ implies $\theta' \geq \mathsf{M}+1$; thus the definition of θ' implies $(x,\infty) \in \boldsymbol{d}$.

We show that if $\theta(x) < \mathsf{M}$, then $(x, \lfloor \theta(x) \rfloor) \in \boldsymbol{d}$.

SubCase $\lfloor \theta(x) \rfloor = \mathsf{M} - 1$: $\theta <_1 \theta'$ implies $\theta'(x) = \mathsf{M}$ and thus $(x,\mathsf{M}-1) \in \boldsymbol{d}$.
SubCase $\lfloor \theta(x) \rfloor = k < \mathsf{M} - 1$: $\theta <_1 \theta'$ implies $\theta'(x) = k+1$ and thus $(x,k) \in \boldsymbol{d}$.

□

Relating the collapsed semantics and the digital automaton, we define the following relation:

$$\langle q, \lambda \rangle \sim \langle p, \boldsymbol{d} \rangle \overset{\text{def}}{\iff} q = p \quad \text{and} \quad \lambda \models \boldsymbol{d}.$$

Lemma 2.6.

$$\begin{tikzcd}[column sep=small] & \langle q, \lambda' \rangle \arrow[d, phantom, "\wr"] & & \exists \langle p, \lambda \rangle \arrow[r, "\tau"', "\alpha"] \arrow[d, phantom, "\wr"] & \langle q, \lambda' \rangle \arrow[d, phantom, "\wr"] & & \exists \langle p, \lambda \rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle q, \lambda' \rangle \arrow[d, phantom, "\wr"] \\ \langle p, \boldsymbol{d} \rangle \arrow[r, "\tau", "\alpha"'] & \langle q, \boldsymbol{d'} \rangle & \implies & \langle p, \boldsymbol{d} \rangle \arrow[r, "\tau", "\alpha"'] & \langle q, \boldsymbol{d'} \rangle & \text{or} & \langle p, \boldsymbol{d} \rangle \arrow[r, "\tau", "\alpha"'] & \langle q, \boldsymbol{d'} \rangle. \end{tikzcd}$$

Proof. We proceed by case analysis on $\langle p, \boldsymbol{d} \rangle \xrightarrow[\alpha]{\tau} \langle q, \boldsymbol{d'} \rangle$.

Case $\langle p, \boldsymbol{d} \rangle \xrightarrow[\alpha]{\mathsf{reset}(x)} \langle q, \boldsymbol{d}[x := 0] \rangle$: This means that there is a rule $p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta$.

It is not difficult to find $r \in \mathbb{R}_{\geq 0}$ such that $\lambda'[x := r] \models \boldsymbol{d}$ from $\lambda' \models \boldsymbol{d}[x := 0]$. Therefore, we have the following:

$$\begin{tikzcd} \langle p, \lambda'[x := r] \rangle \arrow[r, "\mathsf{reset}(x)", "\alpha"'] \arrow[d, phantom, "\wr"] & \langle q, \lambda' \rangle \arrow[d, phantom, "\wr"] \\ \langle p, \boldsymbol{d} \rangle \arrow[r, "\mathsf{reset}(x)", "\alpha"'] & \langle q, \boldsymbol{d}[x := 0] \rangle. \end{tikzcd}$$

Case $\langle p, \boldsymbol{d}\rangle \xrightarrow[\alpha]{x \in_? I} \langle q, \boldsymbol{d}\rangle$: This means that there is a rule $p \xrightarrow[\alpha]{x \in_? I} q \in \Delta$. Since $\lambda' \models \boldsymbol{d}$ and $\boldsymbol{d} \models x \in_? I$, Proposition 2.6 implies $\lambda' \models x \in_? I$ and we have a transition $\langle p, \lambda'\rangle \xrightarrow[\alpha]{x \in_? I} \langle q, \lambda'\rangle$.

Case $\langle p, \boldsymbol{d}\rangle \xrightarrow[\epsilon]{\mathsf{evolve}} \langle p, \boldsymbol{d'}\rangle$ where $\boldsymbol{d} \vdash^* \boldsymbol{d'}$:

$$\text{The diagram} \quad \begin{array}{ccc} & & \lambda' \\ & & \rotatebox{90}{$\models$} \\ \boldsymbol{d} & \vdash^* & \boldsymbol{d'} \end{array} \quad \text{and Proposition 2.12 imply} \quad \begin{array}{ccc} \exists\lambda & \leq & \lambda' \\ \rotatebox{90}{$\models$} & & \rotatebox{90}{$\models$} \\ \boldsymbol{d} & \vdash^* & \boldsymbol{d'}. \end{array}$$

Therefore, we have a transition $\langle p, \lambda\rangle \overset{\delta}{\rightsquigarrow} \langle p, \lambda'\rangle$ where $\lambda' = \lambda + \delta$.

□

2.4.4 Decidability of the Configuration Reachability Problem

Lemma 2.4, Lemma 2.5, and Lemma 2.6 immediately lead to the decidability of the configuration reachability problem.

Theorem 2.3. The following two reachability queries are equivalent:

- On the timed automaton, $\langle q_{\text{init}}, \mathbf{0}\rangle \Rightarrow^* \langle q, \nu\rangle$.
- On the digital automaton, $\langle q_{\text{init}}, \mathcal{D}(\mathbf{0})\rangle \rightarrow^* \langle q, \mathcal{D}(\nu)\rangle$.

This leads to the PSPACE-completenss of the configuration reachability problem of timed automata.

Corollary 2.2. The configuration reachability problem of timed automata is PSPACE-complete.

Proof. It can be easily shown that the configuration reachability problem of timed automata is in PSPACE because we can reduce the configuration reachability problem of a timed automaton to the reachability problem of the corresponding digital automaton by Theorem 2.3.

To show that the configuration reachability problem is PSPACE-hard, we show a linear-time reduction from the location reachability problem into the configuration reachability problem. Let $A = (Q, q_{\text{init}}, F, \Sigma, \mathcal{X}, \Delta)$ be a timed automaton and $\langle q_{\text{init}}, \mathbf{0}\rangle \Rightarrow^*_? \langle q, {}^{\exists}\nu\rangle$ be a query of the location reachability problem. We assume $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and construct the following timed automaton:

$$B = (Q \cup \{s_1, s_2, \ldots, s_n\}, q_{\text{init}}, F, \Sigma, \mathcal{X}, \Delta')$$

where

$$\Delta' = \Delta \cup \left\{ q \xrightarrow[\epsilon]{\mathsf{reset}(x_1)} s_1,\ s_1 \xrightarrow[\epsilon]{\mathsf{reset}(x_2)} s_2,\ \ldots,\ s_{n-1} \xrightarrow[\epsilon]{\mathsf{reset}(x_n)} s_n \right\}.$$

Now it is easily verified that:

$$(\exists\nu.\ \langle q_{\text{init}}, \mathbf{0}\rangle \Rightarrow^* \langle q, \nu\rangle) \iff \langle q_{\text{init}}, \mathbf{0}\rangle \Rightarrow^* \langle s_n, \mathbf{0}\rangle.$$

Therefore, the configuration reachability problem of timed automata is PSPACE-hard. □

It has been shown that the decidability of the configuration reachability problem in general form [CJ99, Dim02, QSW17]. They showed that the binary reachability relation of two locations q and q', $\{(\nu, \nu') : (q, \nu) \to^* (q', \nu')\}$, is effectively definable in the additive theory of real numbers [CJ99, QSW17] or representable by a class of automata called $2n$-automata [Dim02]. On the basis of those characterizations, they showed the decidability of the general configuration reachability problem of timed automata: we can decide if $\langle q, \nu\rangle \to^* \langle q', \nu'\rangle$ for given configurations $\langle q, \nu\rangle$ and $\langle q', \nu'\rangle$. The known time-complexity of the binary configuration reachability problem is EXPTIME-hard.

With respect to their result, our result states that the configuration reachability problem is a tractable case of the binary configuration reachability problem; indeed, the configuration reachability problem is PSPACE-complete with respect to the size of an input timed automaton and query. Especially, the backward simulation on timed automata is important to show that the configuration reachability problem is in PSPACE.

2.5 Extensions of Timed Automata

In this section, we review two extensions of timed automata, timed automata with *diagonal constraints* [BPDG98, BLR05] and *updatable timed automata* [BDFP00a, BDFP00b, BDFP04]. Each class is an extension of timed automata with new constraints and operations; however, these extensions are as expressive as timed automata. Furthermore, the location and configuration reachability problems of them are decidable. We also see an extension, updatable timed automata with diagonal constraints. Unlike the two extensions, this class is more expressive than timed automata and unfortunately its location and configuration reachability problems are undecidable.

2.5.1 Timed Automata with Diagonal Constraints

We extend timed automata by *diagonal constraints* that are constraints of the following form:

$$x - y \in_? I$$

where x and y are clocks and I is an interval. By the constraint $x - y \in_? I$, we can check whether or not the difference of the values of x and y is in I.

Let $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be a valuation, x and y are clocks, and I be an interval. We write $\nu \models x - y \in I$ if $\nu(x) - \nu(y) \in I$.

A timed automaton with diagonal constraints A is a 6-tuple $A = (Q, q_{\text{init}}, F, \Sigma, \mathcal{X}, \Delta)$ where all component except Δ is the same as timed automata and $\Delta \subseteq Q \times \Sigma_\epsilon \times Act_{diag} \times Q$. Act_{diag} is the set of actions of timed automata defined as the following grammar:

$$Act_{diag} ::= \mathsf{reset}(x) \mid x \in_? I \mid x - y \in_? I.$$

The operational semantics of timed automata with diagonal constraints is defined in the same way as timed automata except the transition rule $p \xrightarrow[\alpha]{x-y\in_? I} q$. The meaning of the rule $p \xrightarrow[\alpha]{x-y\in_? I} q$ is defined as follows:

$$\frac{p \xrightarrow[\alpha]{x-y\in_? I} q \in \Delta \quad \nu \models x - y \in I}{\langle p, \nu\rangle \xrightarrow[\alpha]{x-y\in_? I} \langle q, \nu\rangle.}$$

It is the well-known result that adding diagonal constraints to timed automata does not enlarge the language class.

Theorem 2.4 ([BPDG98])**.** Let A be a timed automaton with diagonal constraints. There is a timed automaton B without diagonal constraints that recognizes the same language as A: i.e., $L(A) = L(B)$.

Proof. Let $A = (Q, q_{\text{init}}, F, \Sigma, \mathcal{X}, \Delta)$ and $\mathtt{C}_1 - \mathtt{C}_2 \in_? J$ be a diagonal constraint in A. By removing transition rules with the diagonal constraint $\mathtt{C}_1 - \mathtt{C}_2 \in_? J$, we construct a timed automaton $B = (Q \times \{\mathsf{tt}, \mathsf{ff}\}, \langle q_{\text{init}}, t_{\text{init}}\rangle, F \times \{\mathsf{tt}, \mathsf{ff}\}, \Sigma, \mathcal{X}, \Delta')$ where

- $t_{\text{init}} = \mathsf{tt}$ if $0.0 \in J$ and $t_{\text{init}} = \mathsf{ff}$ if $0.0 \notin J$,
- $L(A) = L(B)$,
- The rules with a diagonal constraint in B is a proper subset of those in A: formally,

$$\left\{q_1 \xrightarrow[\alpha]{x-y\in_? I} q_2 \in \Delta'\right\} \subsetneq \left\{q_1 \xrightarrow[\alpha]{x-y\in_? I} q_2 \in \Delta\right\}.$$

We define a relation $\sim$ between configurations of A and B as follows

$$\langle p, \nu\rangle \sim \langle\langle q, t\rangle, \mu\rangle \overset{\text{def}}{\iff} p = q \quad \text{and} \quad \nu = \mu \quad \text{and} \quad \nu(\mathtt{C}_1) - \nu(\mathtt{C}_2) \in J \iff t = \mathsf{tt}.$$

Therefore, $\langle q, \nu\rangle \sim \langle\langle q, t\rangle, \nu\rangle$ holds if the tag t reflects whether $\nu(\mathtt{C}_1) - \nu(\mathtt{C}_2) \in J$ or not.

For timed transition, the following holds:

$$\langle q, \nu\rangle \sim \langle\langle q, t\rangle, \nu\rangle \implies \forall\delta \in \mathbb{R}_{\geq 0}.\langle q, \nu + \delta\rangle \sim \langle\langle q, t\rangle, \nu + \delta\rangle.$$

This states that the relation $\sim$ is compatible with timed transitions.

For discrete transitions, on the basis of the relation $\sim$, we define the set of transition rules Δ' of B as follows:

- If $q_1 \xrightarrow[\alpha]{x\in_? I} q_2 \in \Delta$, then $\langle q_1, t\rangle \xrightarrow[\alpha]{x\in_? I} \langle q_2, t\rangle \in \Delta'$.
- If $q_1 \xrightarrow[\alpha]{x-y\in_? I} q_2 \in \Delta$,

$$\frac{x - y \in_? I \not\equiv \mathtt{C}_1 - \mathtt{C}_2 \in_? J}{\langle q_1, t\rangle \xrightarrow[\alpha]{x-y\in_? I} \langle q_2, t\rangle \in \Delta',} \qquad \frac{x - y \in_? I \equiv \mathtt{C}_1 - \mathtt{C}_2 \in_? J}{\langle q_1, \mathsf{tt}\rangle \xrightarrow[\alpha]{\mathtt{C}_1\in_?[0:\omega)} \langle q_2, \mathsf{tt}\rangle \in \Delta'.}$$

  (Note that the checking $\mathtt{C}_1 \in_? [0 : \omega)$ always holds.)

- If $q_1 \xrightarrow[\alpha]{\mathsf{reset}(\mathtt{C}_1)} q_2 \in \Delta$, then

$$\frac{J = [0 : 0]}{\langle q_1, t\rangle \xrightarrow[\alpha]{\mathsf{reset}(\mathtt{C}_1)} \langle q_2, \mathsf{tt}\rangle \in \Delta',} \qquad \frac{J \neq [0 : 0]}{\langle q_1, t\rangle \xrightarrow[\alpha]{\mathsf{reset}(\mathtt{C}_1)} \langle q_2, \mathsf{ff}\rangle \in \Delta'.}$$

- If $q_1 \xrightarrow[\alpha]{\mathsf{reset}(\mathtt{C}_2)} q_2 \in \Delta$, then

$$\langle q_1, t\rangle \xrightarrow[\alpha]{\mathsf{reset}(\mathtt{C}_2)\,;\,\mathtt{C}_1 \notin_? J} \langle q_2, \mathsf{ff}\rangle, \quad \langle q_1, t\rangle \xrightarrow[\alpha]{\mathsf{reset}(\mathtt{C}_2)\,;\,\mathtt{C}_1 \in_? J} \langle q_2, \mathsf{tt}\rangle \in \Delta'.$$

- If $q_1 \xrightarrow[\alpha]{\mathsf{reset}(x)} q_2 \in \Delta$ and $x \notin \{\mathtt{C}_1, \mathtt{C}_2\}$, then $\langle q_1, t\rangle \xrightarrow[\alpha]{\mathsf{reset}(x)} \langle q_2, t\rangle \in \Delta'$.

On the definition, we have the following diagram:

$$(1)\quad \begin{tikzcd} \langle q,\nu\rangle \arrow[r, "\tau"', "\alpha"'] \arrow[d, phantom, "\wr"] & \langle q',\nu'\rangle \\ \langle\langle q,t\rangle,\nu\rangle & \end{tikzcd} \quad\Longrightarrow\quad \begin{tikzcd} \langle q,\nu\rangle \arrow[r, "\tau", "\alpha"'] \arrow[d, phantom, "\wr"] & \langle q',\nu'\rangle \arrow[d, phantom, "\wr"] \\ \langle\langle q,t\rangle,\nu\rangle \arrow[r, "\tau'", "\alpha"'] & \langle\langle q',t'\rangle,\nu'\rangle, \end{tikzcd}$$

$$(2)\quad \begin{tikzcd} \langle q,\nu\rangle \arrow[d, phantom, "\wr"] & \\ \langle\langle q,t\rangle,\nu\rangle \arrow[r, "\tau", "\alpha"'] & \langle\langle q',t'\rangle,\nu'\rangle \end{tikzcd} \quad\Longrightarrow\quad \begin{tikzcd} \langle q,\nu\rangle \arrow[r, "\tau'", "\alpha"'] \arrow[d, phantom, "\wr"] & \langle q',\nu'\rangle \arrow[d, phantom, "\wr"] \\ \langle\langle q,t\rangle,\nu\rangle \arrow[r, "\tau", "\alpha"'] & \langle\langle q',t'\rangle,\nu'\rangle. \end{tikzcd}$$

Therefore, we have $L(A) = L(B)$. By repeatedly applying this construction to the timed automaton A, we finally obtain a timed automaton C without diagonal constraints such that $L(A) = L(C)$. $\square$

2.5.2 Updatable Timed Automata

Updatable timed automata (UTA) are an extension of timed automata that allow to update clocks by using intervals [BDFP00a, BDFP00b, BDFP04].

An UTA A is a 6-tuple $A = (Q, q_{\text{init}}, F, \Sigma, \mathcal{X}, \Delta)$ where all component except Δ is the same as timed automata and $\Delta \subseteq Q \times \Sigma_\epsilon \times Act_{upd} \times Q$. Act_{upd} is the set of actions of updatable timed automata defined as the following grammar:

$$Act_{upd} ::= \mathsf{reset}(x) \mid x \in_? I \mid x \leftarrow I.$$

The operational semantics of UTA is defined in the same way as that of timed automata except the transition rule $p \xrightarrow[\alpha]{x\leftarrow I} q$. The meaning of the rule $p \xrightarrow[\alpha]{x\leftarrow I} q$ is defined as follows:

$$\frac{p \xrightarrow[\alpha]{x\leftarrow I} q \in \Delta \quad r \in I}{\langle p,\nu\rangle \xrightarrow[\alpha]{x\leftarrow I} \langle q, \nu[x := r]\rangle.}$$

It is the well-known result that adding update operations to timed automata does not enlarge the language class.

Theorem 2.5 ([BDFP00b, BDFP04])**.** Let A be an UTA. There is a timed automaton B without update operations that recognizes the same language as A: i.e., $L(A) = L(B)$.

Proof. Let $A = (Q, q_{\text{init}}, F, \Sigma, \mathcal{X}, \Delta)$ and we assume A is normalized by Proposition 2.1.

Step1. By the same argument of Proposition 2.1, we can assume the following on A:

- If $p \xrightarrow[\alpha]{x\leftarrow I} q \in \Delta$, we have one of $I = (k : k+1)$, $I = [k : k]$, or $I = (k : \omega)$.

Furthermore, we can remove transition rules of the form $p \xrightarrow[\alpha]{x\leftarrow (k:\omega)} q$ as follows:

- Let H be a natural number that is larger than any constant appearing in a constraint of A:

$$H > \left\{k : p \xrightarrow[\alpha]{x\in_?(k:k+1)} q \in \Delta\right\} \cup \left\{k : p \xrightarrow[\alpha]{x\in_?[k:k]} q \in \Delta\right\} \cup \left\{k : p \xrightarrow[\alpha]{x\in_?(k:\omega)} q \in \Delta\right\}.$$

- We replace any transition rule of the form $p \xrightarrow[\alpha]{x\leftarrow(k:\omega)} q$ by the following rules:

$$p \xrightarrow[\alpha]{x\leftarrow(k:k+1)} q, \quad p \xrightarrow[\alpha]{x\leftarrow[k+1:k+1]} q, \quad p \xrightarrow[\alpha]{x\leftarrow(k+1:k+2)} q, \quad \dots, \quad p \xrightarrow[\alpha]{x\leftarrow(H-1:H)} q, \quad p \xrightarrow[\alpha]{x\leftarrow[H:H]} q.$$

Clearly, this modification does not change the language of A.

Now we call $p \xrightarrow[\alpha]{x\leftarrow[k:k]} q$ a transition with *deterministic update* and $p \xrightarrow[\alpha]{x\leftarrow(k:k+1)} q$ a transition with *nondeterministic update.*

Step2. First, we remove all deterministic update from the UTA A and construct the following UTA:

$$B = (Q \times (\mathcal{X} \to \{0,1,\dots,K\}), (q_{\text{init}}, \mathbf{0}), F \times (\mathcal{X} \to \{0,1,\dots,K\}), \Sigma, \mathcal{X}, \Delta')$$

where $K = \max\left\{k : p \xrightarrow[\alpha]{x\leftarrow[k:k]} q \in \Delta\right\}$.

We define the following relation $\sim$ to relate configurations of A and B:

$$\langle q,\nu\rangle \sim \langle (q,f),\mu\rangle \overset{\text{def}}{\iff} \forall x \in \mathcal{X}.\ \nu(x) = \mu(x) + f(x).$$

On the basis of this relation, we define Δ' as follows:

- If $p \xrightarrow[\alpha]{x\leftarrow[k:k]} q \in \Delta$, then $(p,f) \xrightarrow[\alpha]{\mathsf{reset}(x)} (q, f[x := k]) \in \Delta'$.
- If $p \xrightarrow[\alpha]{x\leftarrow(k:k+1)} q \in \Delta$, then $(p,f) \xrightarrow[\alpha]{x\leftarrow(0:1)} (q, f[x := k]) \in \Delta'$.
- If $p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta$, then $(p,f) \xrightarrow[\alpha]{\mathsf{reset}(x)} (q, f[x := 0]) \in \Delta'$.
- If $p \xrightarrow[\alpha]{x\in_? I} q \in \Delta$, then $(p,f) \xrightarrow[\alpha]{x\in_? I\ominus f(x)} (q,f) \in \Delta'$ where

  $$(i:j)\ominus k \triangleq (i-k:j-k), \quad [i:j]\ominus k \triangleq [i-k:j-k], \quad (i:\omega)\ominus k \triangleq (i-k:\omega).$$

  It should be noted that $r+k \in I \iff r \in I\ominus k$ for an interval I, $r \in \mathbb{R}_{\geq 0}$, and $k \in \mathbb{N}$.

Clearly, the following diagram hold; therefore, $L(A) = L(B)$:

$$\begin{tikzcd}
\langle p,\nu\rangle \arrow[r, "\tau"', "\alpha"] \arrow[d, phantom, "\wr"] & \langle q,\nu'\rangle \arrow[d, phantom, "\wr"] & \langle p,\nu\rangle \arrow[r, "\tau"', "\alpha"] \arrow[d, phantom, "\wr"] & \exists\langle q,\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle (p,f),\mu\rangle \arrow[r, "\tau'"', "\alpha"] & \exists\langle (q,f'),\mu'\rangle, & \langle (p,f),\mu\rangle \arrow[r, "\tau'"', "\alpha"] & \langle (q,f'),\mu'\rangle,
\end{tikzcd}$$

$$\begin{tikzcd}
\langle p,\nu\rangle \arrow[r, rightsquigarrow, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,\nu'\rangle \arrow[d, phantom, "\wr"] & \langle p,\nu\rangle \arrow[r, rightsquigarrow, "\delta"] \arrow[d, phantom, "\wr"] & \exists\langle p,\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle (p,f),\mu\rangle \arrow[r, rightsquigarrow, "\delta"] & \exists\langle (p,f),\mu'\rangle, & \langle (p,f),\mu\rangle \arrow[r, rightsquigarrow, "\delta"] & \langle (p,f),\mu'\rangle.
\end{tikzcd}$$

Note that the above construction also removes nondeterministic updates except transitions of the form $p \xrightarrow[\alpha]{x\leftarrow(0:1)} q$.

Step3. Next, we remove a transition rule with nondeterministic update. We fix an update $\mathtt{C} \leftarrow (0:1)$ and construct the following UTA to remove transition rules with the update $\mathtt{C} \leftarrow (0:1)$:

$$B = (Q \cup \left\{q_{(0:1)}, \overline{q}_{(0:1)} : q \in Q\right\},\ q_{\mathrm{init}},\ F \cup \left\{\overline{q}_{(0:1)} : q \in F\right\},\ \Sigma,\ \mathcal{X}, \Delta').$$

We define the following relation $\sim$ to relate configurations of A and B:

$$\begin{aligned}
&\langle q, \nu\rangle \sim \langle q, \mu\rangle \iff \nu = \mu, \\
&\langle q, \nu\rangle \sim \langle q_{(0:1)}, \mu\rangle \iff \nu \restriction (\mathcal{X} \setminus \{\mathtt{C}\}) = \mu \restriction (\mathcal{X} \setminus \{\mathtt{C}\}) \ \wedge\ 0 \le \mu(\mathtt{C}) < \nu(\mathtt{C}) < 1, \\
&\langle q, \nu\rangle \sim \langle \overline{q}_{(0:1)}, \mu\rangle \iff \nu \restriction (\mathcal{X} \setminus \{\mathtt{C}\}) = \mu \restriction (\mathcal{X} \setminus \{\mathtt{C}\}) \ \wedge\ 0 \le \mu(\mathtt{C}) < \nu(\mathtt{C}) < 1.
\end{aligned}$$

For each location of the form $\overline{q}_{(0:1)}$, we have the following two transition rules:

$$\overline{q}_{(0:1)} \xrightarrow[\epsilon]{\mathtt{C} \in_? [0:1)} q_{(0:1)}, \quad \overline{q}_{(0:1)} \xrightarrow[\epsilon]{\mathtt{C} \in_? (0:1)\,\fatsemi\, \mathtt{C} \leftarrow [1:1]} q \in \Delta'.$$

A location $\overline{q}_{(0:1)}$ is used to nondeterministically update the clock $\mathtt{C}$ by $\mathtt{C} \leftarrow [1:1]$.

We define the set of transition rules Δ' as follows:

- If $p \xrightarrow[\alpha]{\mathtt{C} \leftarrow (0:1)} q \in \Delta$, then $p \xrightarrow[\alpha]{\mathsf{reset}(\mathtt{C})} \overline{q}_{(0:1)}$, $p_{(0:1)} \xrightarrow[\alpha]{\mathsf{reset}(\mathtt{C})} \overline{q}_{(0:1)} \in \Delta'$.
- If $p \xrightarrow[\alpha]{x \leftarrow (0:1)} q \in \Delta$ with $x \neq \mathtt{C}$, then $p \xrightarrow[\alpha]{x \leftarrow (0:1)} q$, $p_{(0:1)} \xrightarrow[\alpha]{\mathtt{C} \in_? [0:1)\,\fatsemi\, x \leftarrow (0:1)} \overline{q}_{(0:1)} \in \Delta'$.
- If $p \xrightarrow[\alpha]{\mathsf{reset}(\mathtt{C})} q \in \Delta$, then $p \xrightarrow[\alpha]{\mathsf{reset}(\mathtt{C})} q$, $p_{(0:1)} \xrightarrow[\alpha]{\mathsf{reset}(\mathtt{C})} q \in \Delta'$.
- If $p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta$ with $x \neq \mathtt{C}$, then

$$p \xrightarrow[\alpha]{\mathsf{reset}(x)} q, \quad p_{(0:1)} \xrightarrow[\alpha]{\mathtt{C} \in_? [0:1)\,\fatsemi\, \mathsf{reset}(x)} \overline{q}_{(0:1)} \in \Delta'.$$

- If $p \xrightarrow[\alpha]{\mathtt{C} \in_? I} q \in \Delta$, then

$$p \xrightarrow[\alpha]{\mathtt{C} \in_? I} q, \qquad I = (0:1) \implies p_{(0:1)} \xrightarrow[\alpha]{\mathtt{C} \in_? [0:1)} \overline{q}_{(0:1)} \in \Delta'.$$

- If $p \xrightarrow[\alpha]{x \in_? I} q \in \Delta$ with $x \neq \mathtt{C}$, then

$$p \xrightarrow[\alpha]{x \in_? I} q, \qquad p_{(0:1)} \xrightarrow[\alpha]{\mathtt{C} \in_? [0:1)\,\fatsemi\, x \in_? I} \overline{q}_{(0:1)} \in \Delta'.$$

We show $L(A) \subseteq L(B)$. Each discrete transition $\langle p, \nu\rangle \overset{\tau}{\underset{\alpha}{\to}} \langle p', \nu'\rangle$ of UTA A can be simulated by a discrete transition of the UTA B as follows:

$$\begin{tikzcd}
\langle p, \nu\rangle \arrow[r, phantom, "\overset{\tau}{\underset{\alpha}{\to}}"] \arrow[d, phantom, "\wr"] & \langle p', \nu'\rangle & & \langle p, \nu\rangle \arrow[r, phantom, "\overset{\tau}{\underset{\alpha}{\to}}"] \arrow[d, phantom, "\wr"] & \langle p', \nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle s, \mu\rangle & & \Longrightarrow & \langle s, \mu\rangle \arrow[r, phantom, "\overset{\tau'}{\underset{\alpha}{\to}}"] & \langle s', \mu'\rangle
\end{tikzcd}$$

where $s \in \{q, q_{(0:1)} : q \in Q\}$ and $s' \in \left\{q, \overline{q}_{(0:1)} : q \in Q\right\}$.

To state the similar property for timed transitions, we use the following notation:

$$
\begin{aligned}
& \langle \overline{p}_{(0:1)}, \nu \rangle \overset{\delta}{\mapsto} \langle p_{(0:1)}, \nu + \delta \rangle \\
\overset{\text{def}}{\iff} \;& \exists \delta_1, \delta_2.\ \delta = \delta_1 + \delta_2 \ \wedge\ \langle \overline{p}_{(0:1)}, \nu \rangle \overset{\delta_1}{\rightsquigarrow} \langle \overline{p}_{(0:1)}, \nu + \delta_1 \rangle \xrightarrow[\epsilon]{\mathtt{C}\in_?[0:1)} \langle p_{(0:1)}, \nu + \delta_1 \rangle \overset{\delta_2}{\rightsquigarrow} \langle p_{(0:1)}, \nu + \delta \rangle, \\
& \langle \overline{p}_{(0:1)}, \nu \rangle \overset{\delta}{\mapsto} \langle p, \mu \rangle \\
\overset{\text{def}}{\iff} \;& \exists \delta_1, \delta_2.\ \delta = \delta_1 + \delta_2 \ \wedge\ \langle \overline{p}_{(0:1)}, \nu \rangle \overset{\delta_1}{\rightsquigarrow} \langle \overline{p}_{(0:1)}, \nu + \delta_1 \rangle \xrightarrow[\epsilon]{\mathtt{C}\in_?(0:1) \fatsemi \mathtt{C}\leftarrow[1:1]} \langle p, \nu' \rangle \overset{\delta_2}{\rightsquigarrow} \langle p, \mu \rangle.
\end{aligned}
$$

Each timed transition of A can be simulated by a compound transition of B as follows:

$$
(1):\ \begin{tikzcd}
\langle p,\nu\rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,\nu'\rangle \\
\langle p,\mu\rangle &
\end{tikzcd}
\implies
\begin{tikzcd}
\langle p,\nu\rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle p,\mu\rangle \arrow[r, squiggly, "\delta"] & \langle p,\mu'\rangle.
\end{tikzcd}
$$

$$
(2):\ \nu(\mathtt{C}) + \delta < 1 \ \wedge\
\begin{tikzcd}
\langle p,\nu\rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,\nu'\rangle \\
\langle \overline{p}_{(0:1)},\mu\rangle &
\end{tikzcd}
\implies
\begin{tikzcd}
\langle p,\nu\rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle \overline{p}_{(0:1)},\mu\rangle \arrow[r, mapsto, "\delta"] & \langle p_{(0:1)},\mu'\rangle.
\end{tikzcd}
$$

$$
(3):\ \nu(\mathtt{C}) + \delta \geq 1 \ \wedge\
\begin{tikzcd}
\langle p,\nu\rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,\nu'\rangle \\
\langle \overline{p}_{(0:1)},\mu\rangle &
\end{tikzcd}
\implies
\begin{tikzcd}
\langle p,\nu\rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle \overline{p}_{(0:1)},\mu\rangle \arrow[r, mapsto, "\delta"] & \langle p,\mu'\rangle.
\end{tikzcd}
$$

Therefore, $L(A) \subseteq L(B)$.

Conversely, we show $L(B) \subseteq L(A)$. It should be noted that for a timed word $w^t \in L(B)$ there is an acceptable run π of B of the following form:

$$
\pi = \langle q_{\text{init}}, \mathbf{0} \rangle \overset{\delta_1}{\Mapsto} \langle s_1, \mu_1 \rangle \xrightarrow[\alpha_1]{\tau_1} \langle s_1', \mu_1' \rangle \overset{\delta_2}{\Mapsto} \langle s_2, \mu_2 \rangle \xrightarrow[\alpha_2]{\tau_2} \langle s_2', \mu_2' \rangle \overset{\delta_3}{\Mapsto} \cdots \overset{\delta_n}{\Mapsto} \langle s_n, \nu_n \rangle \xrightarrow[\alpha_n]{\tau_n} \langle s_n', \nu_n' \rangle
$$

where $\mathsf{tw}(\pi) = w^t$, $\overset{\delta_i}{\Mapsto} = \overset{\delta_i}{\rightsquigarrow}$ or $\overset{\delta_i}{\Mapsto} = \overset{\delta_i}{\mapsto}$, $s_i \in \{q, q_{(0:1)} : q \in Q\}$, and $s_i' \in \left\{q, \overline{q}_{(0:1)} : q \in Q\right\}$.

The UTA B can simulate UTA A by the following diagrams:

$$
\begin{tikzcd}
& \langle p',\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle s,\mu\rangle \arrow[r, "\tau"' , "\alpha"'] & \langle s',\mu'\rangle
\end{tikzcd}
\implies
\begin{tikzcd}
\langle p,\nu\rangle \arrow[r, "\tau'", "\alpha"'] \arrow[d, phantom, "\wr"] & \langle p',\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle s,\mu\rangle \arrow[r, "\tau", "\alpha"'] & \langle s',\mu'\rangle,
\end{tikzcd}
$$

$$
\begin{tikzcd}
& \langle p,\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle p,\mu\rangle \arrow[r, squiggly, "\delta"] & \langle p,\mu'\rangle
\end{tikzcd}
\implies
\begin{tikzcd}
\langle p,\nu\rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle p,\mu\rangle \arrow[r, squiggly, "\delta"] & \langle p,\mu'\rangle,
\end{tikzcd}
$$

$$
\begin{tikzcd}
& \langle p,\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle \overline{p}_{(0:1)},\mu\rangle \arrow[r, mapsto, "\delta"] & \langle p_{(0:1)},\mu'\rangle
\end{tikzcd}
\implies
\begin{tikzcd}
\langle p,\nu\rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle \overline{p}_{(0:1)},\mu\rangle \arrow[r, mapsto, "\delta"] & \langle p_{(0:1)},\mu'\rangle,
\end{tikzcd}
$$

$$
\begin{tikzcd}
& \langle p,\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle \overline{p}_{(0:1)},\mu\rangle \arrow[r, mapsto, "\delta"] & \langle p,\mu'\rangle.
\end{tikzcd}
\implies
\begin{tikzcd}
\langle p,\nu\rangle \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\wr"] & \langle p,\nu'\rangle \arrow[d, phantom, "\wr"] \\
\langle \overline{p}_{(0:1)},\mu\rangle \arrow[r, mapsto, "\delta"] & \langle p,\mu'\rangle.
\end{tikzcd}
$$

Combining these diagrams, we have $L(B) \subseteq L(A)$; thus $L(A) = L(B)$.

Step 4. After Step 3, we obtain an updatable timed automaton where if it has a transition $p \xrightarrow[\alpha]{x \leftarrow I} q$, then $I = [1 : 1]$. By taking Step 2 again, we can remove such transition without introducing transitions of the form $p' \xrightarrow[\alpha]{x \leftarrow (0:1)} q'$. Therefore, we finally obtain a timed automaton without any updatable operations that is equivalent to the original updatable timed automaton A. □

Undecidability of Updatable Timed Automata with Diagonal Constraints.

As we have seen, adding diagonal constraints or update operations does not enlarge the language class of timed automata. We consider updatable timed automata with diagonal constraints (UTA with diagonal constraints).

An UTA with diagonal constraints A is a 6-tuple $A = (Q, q_{\mathrm{init}}, F, \Sigma, \mathcal{X}, \Delta)$ where $\Delta \subseteq Q \times \Sigma_\epsilon \times Act_{\mathit{upd\&diag}} \times Q$. $Act_{\mathit{upd\&diag}}$ is the set of actions of updatable timed automata defined as the following grammar:

$$Act_{\mathit{upd\&diag}} ::= \mathsf{reset}(x) \mid x \in_? I \mid x \leftarrow I \mid x - y \in_? I.$$

The operational semantics of UTA with diagonal constraints is defined in the same way as updatable timed automata and timed automata with diagonal constraints. Unlike updatable timed automata and timed automata with diagonal constraints, the class is more expressive than timed automata. It is formalized by the following result.

Theorem 2.6 ([BDFP00a, BDFP04])**.** The location (and configuration) reachability problem of updatable timed automata with diagonal constraints is undecidable.

Proof. This theorem was shown by encoding Minsky's two counter machines into the class of timed automata. Since two counter machines are Turing-complete [Min61, Min67, HU79], we can reduce the undecidable decision problem, the halting problem of Turing machines, into the location reachability problem. □

Chapter 3

Timed Pushdown Automata

In this chapter, as a preliminary to Chapter 4 and 5, we review the models of computation, pushdown automata, pushdown timed automata (PTA) of Bouajjani et al. [BER94], dense-timed pushdown automata (DTPDA) of Abdulla et al [AAS12a], and timed pushdown automata (TPDA) of Clemente and Lasota [CL15a]. Although this chapter will review existing results of classes of timed pushdown automata, we show a new result that the reachability problem of PTA is already EXPTIME-hard. This EXPTIME-hardness refines the known result of Abdulla et al. that the reachability problem of DTPDA is EXPTIME-complete since DTPDA can easily simulate PTA but not vice versa.

The contents of this section (the relationship of three classes of timed pushdown automata, PTA, DTPDA, and TPDA) is summarized as follows:

Structure: Roughly speaking, we can see the three classes as follows:

$$\begin{aligned} \text{PTA} &= \text{timed automata} + \text{stack}, \\ \text{DTPDA} &= \text{timed automata} + \text{timed stack}, \\ \text{TPDA} &= \text{DTPDA} + \text{diagonal constraints for global and local clocks}. \end{aligned}$$

Expressiveness: The three classes are equally expressive: **PTA = DTPDA = TPDA**.

Decidability: The reachability problem of the three classes are EXPTIME-complete.

3.1 Pushdown Automata

Pushdown automata are an extension of finite automata with a single stack. Formally, a pushdown automaton A is a 7-tuple $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, Z, \Delta)$ where

- Q is a finite set of control locations, q_{init} is the initial location, and $F \subseteq Q$ is a set of accepting locations;
- Σ is a finite input alphabet;
- Γ is a finite stack alphabet and $Z \in \Gamma$ is the initial stack symbol;
- $\Delta \subseteq Q \times \Sigma_\epsilon \times \Gamma \times \Gamma^* \times Q$ is a finite set of transition rules.
  - To denote a transition rule $\langle p, \alpha, \gamma, w, q \rangle \in \Delta$, we write $p \xhookrightarrow[\alpha]{\gamma/w} q$.

The operational semantics of the pushdown automaton A is defined as an infinite labeled transition system $T_A = (Q \times \Gamma^*, \Rightarrow)$ where the set of transitions $\Rightarrow$ is defined as follows:

$$\frac{p \xhookrightarrow[\alpha]{\gamma/w} q \in \Delta \quad w' \in \Gamma^*}{\langle p, w'\gamma \rangle \overset{\alpha}{\Rightarrow} \langle q, w'w \rangle.}$$

A state of the infinite transition system T_A (or a configuration of the pushdown automaton A) is a pair $\langle q, w\rangle$ of a location $q \in Q$ and stack $w \in \Gamma^*$. A transition $\langle p, w'\gamma\rangle \Rightarrow \langle q, w'w\rangle$ means that we pop the stack top symbol γ and then push a sequence of symbols w.

We define the language $L(A)$ of the pushdown automaton A as follows:

$$L(A) \triangleq \left\{\alpha_1\alpha_2\cdots\alpha_n : \langle q_{\text{init}}, Z\rangle \overset{\alpha_1}{\Longrightarrow} \boldsymbol{c}_1 \overset{\alpha_2}{\Longrightarrow} \cdots \overset{\alpha_n}{\Longrightarrow} \langle q_F, w\rangle,\ q_F \in F\right\}.$$

We consider the following three decision problems on pushdown automata:

Emptiness Problem. The emptiness problem of pushdown automata decides whether or not $L(A) = \emptyset$.

Location Reachability Problem. For a given location q, the location reachability problem of pushdown automata $\langle q_{\text{init}}, Z\rangle \Rightarrow^*_? \langle q, \exists w\rangle$ decides whether or not we can reach a configuration $\langle q, w\rangle$ with some stack w from the initial configuration $\langle q_{\text{init}}, Z\rangle$.

Configuration Reachability Problem. For a given configuration $\langle q, w\rangle$, the configuration reachability problem of pushdown automata $\langle q_{\text{init}}, Z\rangle \Rightarrow^*_? \langle q, w\rangle$ decides whether or not we can reach the configuration $\langle q, w\rangle$ from the initial configuration $\langle q_{\text{init}}, Z\rangle$.

It is a well-known result that these three decision problems can be solved in polynomial time for the size of a given input pushdown automaton (and a given configuration).

Theorem 3.1 ([BEM97, FWW97])**.** The emptiness and location reachability problems can be solved in polynomial time with respect to the size of an input pushdown automaton.

Theorem 3.2 ([BEM97, FWW97])**.** The configuration reachability problem $\langle q_{\text{init}}, Z\rangle \Rightarrow^*_? \langle q, w\rangle$ can be solved in polynomial time with respect to the size of an input pushdown automaton and the length of w.

3.2 Nonstandard Formulation of Pushdown Automata

We give a nonstandard formulation of pushdown automata. On this formulation, a pushdown automaton B is a 6-tuple $B = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \Delta)$ without the initial stack symbol where each component is the same as that of the standard formulation except Δ:

- $\Delta \subseteq Q \times \Sigma_\epsilon \times Act_{\text{PDA}} \times Q$ is a finite set of transition rules. Act_{PDA} is the set of actions for pushdown automata that is defined by the following grammar:

$$Act_{\text{PDA}} ::= \mathsf{push}(\gamma) \mid \mathsf{pop}(\gamma) \mid \mathsf{nop}$$

where $\gamma \in \Gamma$. To denote a transition rule $\langle p, \alpha, \tau, q\rangle \in \Delta$, we write $p \xrightarrow[\alpha]{\tau} q$.

We call pushdown automata of the nonstandard formulation nonstandard pushdown automata. The operational semantics of a nonstandard pushdown automaton is defined as an infinite labeled transition system $T_B = (Q \times \Gamma^*, \to)$. The set of labeled transitions $\to$ is defined as follows:

$$\frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta}{\langle p, w\rangle \xrightarrow[\alpha]{\mathsf{push}(\gamma)} \langle q, w\gamma\rangle}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta}{\langle p, w\gamma\rangle \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} \langle q, w\rangle}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta}{\langle p, w\rangle \xrightarrow[\alpha]{\mathsf{nop}} \langle q, w\rangle}.$$

We also define the language $L(B)$ of a nonstandard pushdown automaton:

$$L(B) \triangleq \left\{\alpha_1\alpha_2\cdots\alpha_n : \langle q_{\text{init}}, \epsilon\rangle \xrightarrow[\alpha_1]{\tau_1} \boldsymbol{c}_1 \xrightarrow[\alpha_2]{\tau_2} \cdots \xrightarrow[\alpha_n]{\tau_n} \langle q_F, w\rangle,\ q_F \in F\right\}.$$

The initial configuration is the pair of the initial location q_{init} and the empty stack.

The standard and nonstandard pushdown automata are equally expressive.

Lemma 3.1. Let A be a pushdown automaton. There is a nonstandard pushdown automaton B such that $L(A) = L(B)$.

Conversely, for a nonstandard pushdown automaton B', there is a pushdown automaton A' such that $L(A') = L(B')$.

Proof. We extend nonstandard pushdown automata by the following transition rule:

$$p \xrightarrow[\alpha]{\tau_1 \fatsemi \tau_2} q.$$

This rule is intended to perform two actions at a single transition; therefore, the semantics of this rule is defined as follows:

$$\frac{\langle p, w\rangle \xrightarrow[\alpha]{\tau_1} \langle p', w''\rangle \quad \langle p', w''\rangle \xrightarrow[\epsilon]{\tau_2} \langle q, w'\rangle \quad (p' \text{ is a fresh location})}{\langle p, w\rangle \xrightarrow[\alpha]{\tau_1 \fatsemi \tau_2} \langle q, w'\rangle}$$

We also use more general forms like $p \xrightarrow[\alpha]{\tau_1 \fatsemi \tau_2 \fatsemi \cdots \fatsemi \tau_n} q$. It is clear that adding this transition rules does not enlarge the language class of nonstandard pushdown automata.

Let $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, Z, \Delta)$. From the pushdown automaton A, we construct the following nonstandard pushdown automaton B:

$$B = (Q \cup \{q'_{\text{init}}\}, q'_{\text{init}}, F, \Sigma, \Gamma, \Delta')$$

where Δ' is defined as follows:

$$\frac{}{q'_{\text{init}} \xrightarrow[\epsilon]{\mathsf{push}(Z)} q_{\text{init}} \in \Delta'}, \qquad \frac{p \xhookrightarrow[\alpha]{\gamma/\epsilon} q \in \Delta}{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta'}, \qquad \frac{p \xhookrightarrow[\alpha]{\gamma/\gamma_1\gamma_2\dots\gamma_n} q \in \Delta \quad n \geq 1 \quad \gamma_i \in \Gamma}{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma) \fatsemi \mathsf{push}(\gamma_1) \fatsemi \mathsf{push}(\gamma_2) \fatsemi \cdots \fatsemi \mathsf{push}(\gamma_n)} q \in \Delta'},$$

It is easily verified that $L(A) = L(B)$ from this construction.

From nonstandard formulation to standard formulation Let $B' = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \Delta)$. We construct the following pushdown automaton A':

$$A' = (Q, q_{\text{init}}, F, \Sigma, \Gamma \cup \{Z\}, Z, \Delta')$$

where Δ' is defined as follows:

$$\frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta \quad \gamma' \in \Gamma \cup \{Z\}}{p \xhookrightarrow[\alpha]{\gamma'/\gamma'\gamma} q \in \Delta'}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta}{p \xhookrightarrow[\alpha]{\gamma/\epsilon} q \in \Delta'}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta \quad \gamma \in \Gamma}{p \xhookrightarrow[\alpha]{\gamma/\gamma} q \in \Delta'},$$

To show $L(A') = L(B')$, we define a relation $\sim$ between configurations of A' and B':

$$\langle q, Zw\rangle \sim \langle q, w\rangle.$$

It can be easily verified that this relation forms a bisimulation between A' and B'; therefore, $L(A') = L(B')$.

□

3.3 Pushdown Timed Automata

Bouajjani, Echahed, and Robbana introduced *pushdown timed automata (PTA)* by extending timed automata with a single stack [BER94]. Formally, a pushdown timed automaton A is a 7-tuple $A = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ where

- Q is a finite set of control locations, $q_{\mathrm{init}} \in Q$ is the initial location, $F \subseteq Q$ is a set of accepting locations;
- Σ is a finite input alphabet; Γ is a finite stack alphabet;
- $\mathcal{X}$ is a finite set of clocks;
- $\Delta \subseteq Q \times \Sigma_\epsilon \times \mathit{Act}_{\mathrm{PTA}} \times Q$ is a finite set of transition rules.
  - To denote a transition rule $\langle p, \alpha, \tau, q\rangle \in \Delta$, we write $p \xrightarrow[\alpha]{\tau} q$.

$\mathit{Act}_{\mathrm{PTA}}$ is the set of actions of pushdown timed automata defined by the following grammar:

$$\mathit{Act}_{\mathrm{PTA}} ::= x \in_? I \mid \mathsf{reset}(x) \mid \mathsf{push}(\gamma) \mid \mathsf{pop}(\gamma) \mid \mathsf{nop}.$$

The transition rules $x \in_? I$ and $\mathsf{reset}(x)$ come from timed automata and $\mathsf{push}(\gamma)$, $\mathsf{pop}(\gamma)$, and nop come from pushdown automata (of the nonstandard formulation).

The operational semantics of the PTA A is defined as an infinite labeled transition system $T_A = (Q \times (\mathcal{X} \to \mathbb{R}_{\geq 0}) \times \Gamma^*, \to, \rightsquigarrow)$ where the set of discrete transitions $\to$ is defined as follows:

$$\frac{p \xrightarrow[\alpha]{x \in_? I} q \in \Delta \quad \nu \models x \in I}{\langle p, \nu, w\rangle \xrightarrow[\alpha]{x \in_? I} \langle q, \nu, w\rangle,} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta}{\langle p, \nu, w\rangle \xrightarrow[\alpha]{\mathsf{reset}(x)} \langle q, \nu[x := 0], w\rangle,}$$

$$\frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta}{\langle p, \nu, w\rangle \xrightarrow[\alpha]{\mathsf{push}(\gamma)} \langle q, \nu, w\gamma\rangle,} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta}{\langle p, \nu, w\gamma\rangle \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} \langle q, \nu, w\rangle,} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta}{\langle p, \nu, w\rangle \xrightarrow[\alpha]{\mathsf{nop}} \langle q, \nu, w\rangle,}$$

and the set of timed transitions $\rightsquigarrow$ is defined as follows:

$$\frac{\delta \in \mathbb{R}_{\geq 0}}{\langle p, \nu, w\rangle \overset{\delta}{\rightsquigarrow} \langle p, \nu + \delta, w\rangle.}$$

We define the language of the PTA A in the same way as timed automata:

$$L(A) \triangleq \left\{ \mathtt{tw}(\pi) : \pi = \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \overset{\delta_1}{\rightsquigarrow} \xrightarrow[\alpha_1]{\tau_1} \cdots \overset{\delta_n}{\rightsquigarrow} \xrightarrow[\alpha_n]{\tau_n} \langle q_F, \nu, w\rangle,\ q_F \in F \right\}.$$

Location Reachability Problem of PTA and Pushdown Digital Automata. We consider the location reachability problem of PTA. For a given location q, the location reachability problem $\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^*_? \langle q, {}^\exists\nu, {}^\exists w\rangle$ decides whether or not we can reach a configuration $\langle q, \nu, w\rangle$ with some valuation ν and stack w from the initial configuration $\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle$.

As we showed the decidability of the location reachability problem of timed automata, we can also show the decidability of the location reachability problem of PTA by considering *pushdown digital automata*.

Let $A = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ be a pushdown timed automaton. The pushdown digital automaton of A is the following pushdown automaton $\mathcal{D}(A)$:

$$\mathcal{D}(A) = (Q \times \mathbb{D}, \langle q_{\mathrm{init}}, \mathcal{D}(\mathbf{0})\rangle, F \times \mathbb{D}, \Sigma, \Gamma, \Delta')$$

where the set of transition rules Δ' is defined as follows: first, we define transition rules that correspond to discrete transitions of A

$$\frac{p \xrightarrow[\alpha]{x \in ?I} q \in \Delta \quad \boldsymbol{d} \models x \in I}{\langle p, \boldsymbol{d} \rangle \xrightarrow[\alpha]{x \in ?I} \langle q, \boldsymbol{d} \rangle \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta}{\langle p, \boldsymbol{d} \rangle \xrightarrow[\alpha]{\mathsf{reset}(x)} \langle q, \boldsymbol{d}[x := 0] \rangle \in \Delta',}$$

$$\frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta}{\langle p, \boldsymbol{d} \rangle \xrightarrow[\alpha]{\mathsf{push}(\gamma)} \langle q, \boldsymbol{d} \rangle \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta}{\langle p, \boldsymbol{d} \rangle \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} \langle q, \boldsymbol{d} \rangle \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta}{\langle p, \boldsymbol{d} \rangle \xrightarrow[\alpha]{\mathsf{nop}} \langle q, \boldsymbol{d} \rangle \in \Delta',}$$

next, we define transition rules that correspond to timed transitions of A

$$\frac{\boldsymbol{d} \vdash^* \boldsymbol{d'}}{\langle p, \boldsymbol{d} \rangle \xrightarrow[\epsilon]{\mathtt{nop}} \langle p, \boldsymbol{d'} \rangle \in \Delta'.}$$

(We also write $\langle p, \boldsymbol{d} \rangle \xrightarrow[\epsilon]{\mathtt{time}} \langle p, \boldsymbol{d'} \rangle$ if $\boldsymbol{d} \vdash^* \boldsymbol{d'}$ instead of $\langle p, \boldsymbol{d} \rangle \xrightarrow[\epsilon]{\mathtt{nop}} \langle p, \boldsymbol{d'} \rangle$.)

To relate the pushdown timed automaton A and the pushdown digital automaton $\mathcal{D}(A)$, we define the following relation on their configurations:

$$\langle q, \nu, w \rangle \sim \langle (q, \boldsymbol{d}), w \rangle \overset{\text{def}}{\iff} \nu \models \boldsymbol{d}.$$

As we have shown the similar relation between a timed automaton and the digital automaton forms a bisimulation, we can show the above relation $\sim$ forms a bisimulation between A and $\mathcal{D}(A)$ and the following holds.

Lemma 3.2. The pushdown digital automaton $\mathcal{D}(A)$ can the pushdown timed automaton A based on the relation $\sim$.

$$\begin{array}{ccc} \langle p, \nu, w \rangle \xrightarrow[\alpha]{\tau} \langle q, \nu', w' \rangle & & \langle p, \nu, w \rangle \xrightarrow[\alpha]{\tau} \langle q, \nu', w' \rangle \\ \wr & \Longrightarrow & \wr \qquad\qquad\quad \wr \\ \langle (p, \boldsymbol{d}), w \rangle & & \langle (p, \boldsymbol{d}), w \rangle \xrightarrow[\alpha]{\tau} \exists \langle (q, \boldsymbol{d'}), w' \rangle. \end{array}$$

$$\begin{array}{ccc} \langle p, \nu, w \rangle \overset{\delta}{\rightsquigarrow} \langle p, \nu', w \rangle & & \langle p, \nu, w \rangle \overset{\delta}{\rightsquigarrow} \langle p, \nu', w \rangle \\ \wr & \Longrightarrow & \wr \qquad\qquad\quad \wr \\ \langle (p, \boldsymbol{d}), w \rangle & & \langle (p, \boldsymbol{d}), w \rangle \xrightarrow[\epsilon]{\mathtt{time}} \exists \langle (p, \boldsymbol{d'}), w \rangle. \end{array}$$

Lemma 3.3. The pushdown timed automaton A can forwardly simulate the pushdown digital automaton $\mathcal{D}(A)$ based on the relation $\sim$.

$$\begin{array}{ccc} \langle p, \nu, w \rangle & & \langle p, \nu, w \rangle \xrightarrow[\alpha]{\tau} \exists \langle q, \nu', w' \rangle \\ \wr & \Longrightarrow & \wr \qquad\qquad\quad \wr \\ \langle (p, \boldsymbol{d}), w \rangle \xrightarrow[\alpha]{\tau} \langle (q, \boldsymbol{d'}), w' \rangle & & \langle (p, \boldsymbol{d}), w \rangle \xrightarrow[\alpha]{\tau} \langle (q, \boldsymbol{d'}), w' \rangle \end{array}$$

$$\begin{array}{ccc} \langle p, \nu, w \rangle & & \langle p, \nu, w \rangle \overset{\delta}{\rightsquigarrow} \exists \langle p, \nu', w \rangle \\ \wr & \Longrightarrow & \wr \qquad\qquad\quad \wr \\ \langle (p, \boldsymbol{d}), w \rangle \xrightarrow[\alpha]{\mathtt{time}} \langle (p, \boldsymbol{d'}), w \rangle & & \langle (p, \boldsymbol{d}), w \rangle \xrightarrow[\alpha]{\mathtt{time}} \langle (p, \boldsymbol{d'}), w \rangle \end{array}$$

These two lemmas lead to the following theorem.

Theorem 3.3. Let A be a pushdown timed automaton and $\mathcal{D}(A)$ be the corresponding pushdown digital automaton. The following location reachability of the timed automaton A and the digital automaton D_A are equivalent:

$$\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \langle q, {}^{\exists}\nu, {}^{\exists}w\rangle \iff \langle q_{\text{init}}, \mathcal{D}(\mathbf{0}), \epsilon\rangle \to^* \langle (q, {}^{\exists}\boldsymbol{d}), {}^{\exists}w\rangle.$$

This theorem and the decidability of the location reachability problem of pushdown automata imply the decidability of the location reachability problem of timed pushdown automata.

Corollary 3.1. The location reachability problem $\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \langle q, {}^{\exists}\nu, {}^{\exists}w\rangle$ of pushdown timed automata is EXPTime-complete.

Proof. First, we show the problem is in EXPTime. For a given pushdown timed automaton A, the size of the corresponding pushdown digital automaton is exponential in the size of A. Since we can solve the location reachability problem of pushdown automata in polynomial time, the corresponding location reachability problem of the pushdown digital automaton can be solved in exponential time with respect to the size of A.

Next, to show the problem is EXPTime-hard, we use the following EXPTime-complete problem [HLMS12]:

> Let P be a pushdown automaton and $M_1, M_2, \dots, M_k$ be k finite automata. The language emptiness problem $L(P) \cap \bigcap_{i=1}^{k} L(M_i) =_? \emptyset$ is EXPTime-complete for the size of P and M_i.

We also use the following PSPACE-complete problem [Koz77]:

> Let $M_1, M_2, \dots, M_k$ be k finite automata. The language emptiness problem of $\bigcap_{i=1}^{k} L(M_i) =_? \emptyset$ is PSPACE-complete for the size of M_i.

As we have seen in the proof of Theorem 2.1 of Chapter 1, we can construct a timed automaton B in polynomial time for the size of the automata M_i such that $\bigcap_{i=1}^{k} L(M_i) = \emptyset \iff L(B) = \emptyset$.

Here we assume the alphabet of B is Σ and $k = 3$ to explain our construction. The following holds from the construction of Theorem 2.1:

$$\begin{array}{ll} & \sigma_1\ s_1 \sharp\ t_1\ \sharp'\ u_1 \quad \sigma_2\ s_2 \sharp\ t_2\ \sharp'\ u_2 \quad \cdots \quad \sigma_n\ s_n \sharp\ t_n\ \sharp'\ u_n \in \mathit{Untime}(L(B)) \\ \iff & \sigma_1\sigma_2 \dots \sigma_n \in \bigcap_{i=1}^{k} L(M_i) \end{array}$$

We can construct a pushdown timed automaton C in polynomial time with respect to the size of P such that

- If $\sigma_1\sigma_2 \dots \sigma_n \in L(P)$, then there is a timed word $\tau \in L(C)$ such that

$$\mathit{Untime}(\tau) = \sigma_1\ s_1 \sharp\ t_1\ \sharp'\ u_1 \quad \sigma_2\ s_2 \sharp\ t_2\ \sharp'\ u_2 \quad \cdots \quad \sigma_n\ s_n \sharp\ t_n\ \sharp'\ u_n.$$

- $L(C) = \mathit{Untime}^{-1}(\mathit{Untime}(L(C))) \cap \mathbf{TW}(\Sigma)$.

From the construction, $L(P) \cap \bigcap_{i=1}^{k} L(M_i) = \emptyset \iff L(C) \cap L(B) = \emptyset$ holds.

Finally, we can construct a pushdown timed automaton D by product construction for B and C in polynomial time for the size of B and C where $L(D) = L(B) \cap L(C)$. $\square$

We also have the EXPTIME-completeness of the emptiness problem of pushdown timed automata.

Corollary 3.2. The emptiness problem of PTA, which decides whether or not $L(A) = \emptyset$ for a given PTA, is EXPTIME-complete.

Proof. First, we show that the emptiness problem of PTA is in EXPTIME. Let $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ be a PTA. We can construct the following PTA B in linear time with respect to the size of A:

$$B = (Q \cup \{q_{\text{goal}}\}, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta'),$$
$$\Delta' = \Delta \cup \left\{ q_F \xrightarrow[\epsilon]{\mathsf{nop}} q_{\text{goal}} : q_F \in F \right\}.$$

It is clear that $L(A) \neq \emptyset$ holds for A iff $\langle q_{\text{init}}, \mathbf{0}\rangle \Rightarrow^* \langle q_{\text{goal}}, {}^{\exists}\nu\rangle$ holds on B. Since the location reachability problem of PTA is in EXPTIME, the emptiness problem of PTA is also in EXPTIME.

Next, we show that the emptiness problem of PTA is EXPTIME-hard. For a query $\langle q_{\text{init}}, \mathbf{0}\rangle \Rightarrow^* \langle q_{\text{goal}}, {}^{\exists}\nu\rangle$ of the location reachability problem, we can construct the following PTA B in linear time with respect to the size of A:

$$B = (Q, q_{\text{init}}, \{q_{\text{goal}}\}, \Sigma, \Gamma, \mathcal{X}, \Delta).$$

It is clear that $\langle q_{\text{init}}, \mathbf{0}\rangle \Rightarrow^* \langle q_{\text{goal}}, {}^{\exists}\nu\rangle$ holds iff $L(B) \neq \emptyset$. Since the location reachability problem of PTA is EXPTIME-hard, the emptiness problem of PTA is EXPTIME-hard. □

3.4 Hierarchy Theorem of Pushdown Timed Automata

In the present section, we show the intuitive property for pushdown timed automata that pushdown timed automata with $(n+1)$-clocks are more expressive than pushdown timed automata with n-clocks.

To state formally this, we write $\mathbf{PTA}_n$ for the language class of pushdown timed automata with n-clocks. Furthermore, we use the following timed language L_n:

$$L_n \triangleq \left\{ (a_n, t_n)(a_{n-1}, t_{n-1}) \dots (a_1, t_1)(b_1, t'_1) \dots (b_{n-1}, t'_{n-1})(b_n, t'_n) : t'_i - t_i = i \right\}.$$

On the basis of these notions, our result is the following (Theorem 3.4):

$$\forall n \geq 1,\ L_{n+1} \in \mathbf{PTA}_{n+1} \ \wedge\ L_{n+1} \notin \mathbf{PTA}_n.$$

Since $\mathbf{PTA}_n \subseteq \mathbf{PTA}_{n+1}$ is clear, this result shows that the strict expressiveness hierarchy $\mathbf{PTA}_0 \subsetneq \mathbf{PTA}_1 \subsetneq \mathbf{PTA}_2 \subsetneq \cdots \subsetneq \mathbf{PTA}_i \subsetneq \cdots$.

3.4.1 Preliminaries for Theorem 3.4

We need some notions and propositions to show this theorem.

Only in the present section, we use the following notation temporarily

- $\mathbb{P} \triangleq \mathbb{N} \setminus \{0\}$ denotes the set of positive natural numbers.
- $\boldsymbol{c} \Rightarrow \boldsymbol{c'}$ denotes either a discrete or timed transition.
- $\boldsymbol{c} \underset{r}{\Rightarrow^*} \boldsymbol{c'}$ denotes a sequence of transitions $\Rightarrow$ where $r \in \mathbb{R}_{\geq 0}$ is the elapsed time between $\boldsymbol{c}$ and $\boldsymbol{c'}$.
- For a transition $\xrightarrow[\alpha]{\tau}$, if the action label τ is not important, then we omit it and write simply $\xrightarrow[\alpha]{}$.

Definition 3.1 (Indistinguishable configurations)**.** Let A be a PTA and $\langle p_1, \nu_1, w_1\rangle$ and $\langle p_2, \nu_2, w_2\rangle$ be configurations of A. These two configurations cannot be distinguished if $\langle p_1, \nu_1, w_1\rangle \cong \langle p_2, \nu_2, w_2\rangle$ where $\cong$ is defined as follows:

$$p_1 = p_2, \qquad \mathcal{D}(\nu_1) = \mathcal{D}(\nu_2), \qquad w_1 = w_2.$$

■

Proposition 3.1. Let $\langle q, \nu, w\rangle$ be a configuration where there are no clocks x such that $\mathit{frac}(\nu(x)) = 0.0$. There exists a real number $0 < \delta < 1$ such that $\langle q, \nu, w\rangle \cong \langle q, \nu + \delta, w\rangle$.

Proof. Recall the definition of $<_2$ in the chapter of timed automata:

$$\nu <_2 \nu' \overset{\text{def}}{\iff} \begin{cases} \text{there is no clock } x \text{ such that } \mathit{frac}(\nu(x)) = 0.0 \text{ and} \\ \nu' < \nu + (1.0 - \mathit{max\_fract}(\nu)). \end{cases}$$

where

$$\mathit{max\_fract}(\nu) \triangleq \max\{\mathit{frac}(\nu(x)) : x \in \mathcal{X}\}.$$

From the assumption, there exists δ such that $\nu <_2 \nu + \delta$. Since there exists a digital valuation $\boldsymbol{d}$ such that $\nu \models \boldsymbol{d}$ and $\nu + \delta \models \boldsymbol{d}$, we have $\langle q, \nu, w\rangle \cong \langle q, \nu + \delta, w\rangle$. □

Proposition 3.2. Let A be a PTA, and $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are indistinguishable configurations.

$$\begin{tikzcd} \boldsymbol{c}_1 \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\cong"] & \boldsymbol{c}_1' & & \boldsymbol{c}_1 \arrow[r, squiggly, "\delta"] \arrow[d, phantom, "\cong"] & \boldsymbol{c}_1' \arrow[d, phantom, "\cong"] \\ \boldsymbol{c}_2 & & \Longrightarrow & \boldsymbol{c}_2 \arrow[r, squiggly, "\exists\delta'"] & \exists\boldsymbol{c}_2'. \end{tikzcd}$$

Proof. Let $\boldsymbol{c}_1 = \langle p_1, \nu_1, w_1\rangle$ and $\boldsymbol{c}_2 = \langle p_2, \nu_2, w_2\rangle$. We have the following diagram for some digital valuations $\boldsymbol{d}$ and $\boldsymbol{d'}$:

$$\begin{array}{ccc} \nu_1 & \leq & \nu_1 + \delta \\ \rotatebox{90}{$\models$} & & \rotatebox{90}{$\models$} \\ \boldsymbol{d} & \vdash^* & \boldsymbol{d'}. \end{array}$$

Since $\nu_1 \models \boldsymbol{d}$ and $\nu_2 \models \boldsymbol{d'}$, Proposition 2.10 of Chapter 2 implies there is $\delta' \in \mathbb{R}_{\geq 0}$ that satisfies the following and it concludes the proof:

$$\begin{array}{ccc} \nu_2 & \leq & \nu_2 + \delta' \\ \rotatebox{90}{$\models$} & & \rotatebox{90}{$\models$} \\ \boldsymbol{d} & \vdash^* & \boldsymbol{d'}. \end{array}$$

□

The same property also holds for discrete transitions.

Proposition 3.3. Let A be a PTA and $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are indistinguishable configurations.

$$\begin{tikzcd} \boldsymbol{c}_1 \arrow[r, "\tau", "\alpha"'] \arrow[d, phantom, "\cong"] & \boldsymbol{c}_1' & & \boldsymbol{c}_1 \arrow[r, "\tau", "\alpha"'] \arrow[d, phantom, "\cong"] & \boldsymbol{c}_1' \arrow[d, phantom, "\cong"] \\ \boldsymbol{c}_2 & & \Longrightarrow & \boldsymbol{c}_2 \arrow[r, "\tau", "\alpha"'] & \exists\boldsymbol{c}_2'. \end{tikzcd}$$

These two propositions lead to the following lemma.

Lemma 3.4. Let A be a PTA of which the set of accepting locations is F, and $\boldsymbol{c}_1, \boldsymbol{c}_2$ be indistinguishable configurations of A.

If $\boldsymbol{c}_1 \Rightarrow^* \langle q_F, \_, \_\rangle$ where $q_F \in F$, then also $\boldsymbol{c}_2 \Rightarrow^* \langle q_F, \_, \_\rangle$.

Remark: hereafter if a component is irrelevant to a discussion, we omit the component and simply write as $\langle q_F, \_, \_\rangle$.

We need the following technical proposition.

Proposition 3.4. We assume the following part of a computation of a PTA:

$$\langle q_1, \nu_1, \_\rangle \overset{\delta_1}{\rightsquigarrow} \xrightarrow[\alpha_1]{} \overset{\delta_2}{\rightsquigarrow} \xrightarrow[\alpha_2]{} \overset{\delta_3}{\rightsquigarrow} \cdots \overset{\delta_n}{\rightsquigarrow} \langle q_2, \nu_2, \_\rangle. \quad (n \geq 1)$$

If $\sum_{i=1}^{n} \delta_i < \nu_2(x)$, then $\nu_1(x) = \nu_2(x) - \sum_{i=1}^{n} \delta_i$.

Proof. We proceed by induction on n. First, as the base case of the induction, we consider the following one:

$$\langle q_1, \nu_1, \_\rangle \overset{\delta_1}{\rightsquigarrow} \langle q_2, \nu_2, \_\rangle.$$

For this case, $\nu_1(x) = \nu_2(x) - \delta_1$ holds trivially.

Next, as the induction step, we consider the following one:

$$\langle q_1, \nu_1, \_\rangle \overset{\delta_1}{\rightsquigarrow}\xrightarrow[\alpha_1]{}\overset{\delta_2}{\rightsquigarrow} \cdots \overset{\delta_{n-1}}{\rightsquigarrow} \langle p, \nu, \_\rangle \xrightarrow[\alpha_n]{} \langle p', \nu', \_\rangle \overset{\delta_n}{\rightsquigarrow} \langle q_2, \nu_2, \_\rangle.$$

Since we cannot reset x on the transition $\xrightarrow[\alpha_n]{}$, $\sum_{i=1}^{n-1} \delta_i < \nu(x)$. Indeed, if we reset, then $\nu_2(x) = \delta_n$ and it contradicts to $\sum_{i=1}^{n} \delta_i < \nu_2(x)$. Therefore, by the induction hypothesis, we have $\nu_1(x) = \nu(x) - \sum\limits_{i=1}^{n-1} \delta_i$. It suffices to show $\nu(x) = \nu_2(x) - \delta_n$ and it is verified by the following:

1. $\nu'(x) = \nu_2(x) - \delta_n > 0$ since $\nu_2(x) > \sum\limits_{i=1}^{n} \delta_i \geq \delta_n$.
2. Among $\langle p, \nu, \_\rangle \xrightarrow[\alpha_n]{} \langle p', \nu', \_\rangle$, we cannot reset x and thus $\nu(x) = \nu'(x)$. Indeed, if $\nu'(x) = 0.0$, it contradicts to $\nu'(x) > 0$.

□

We will use the following special case of Proposition 3.4.

Proposition 3.5. We assume the following part of a computation of a PTA:

$$\langle q_1, \nu_1, \_\rangle \overset{\delta_1}{\rightsquigarrow}\xrightarrow[\alpha_1]{}\overset{\delta_2}{\rightsquigarrow}\xrightarrow[\alpha_2]{}\overset{\delta_3}{\rightsquigarrow} \cdots \overset{\delta_n}{\rightsquigarrow} \langle q_2, \nu_2, \_\rangle.$$

If $0 < \sum\limits_{i=1}^{n} \delta_i < 1$ and $\nu_2(x) \in \mathbb{P}$, then $\mathit{frac}(\nu_1(x)) = 1.0 - \sum\limits_{i=1}^{n} \delta_i$.

3.4.2 1-clock language

It is clear that the following 1-clock language can be recognized by a 1-clock pushdown timed automaton:

$$L_1 = \left\{(a_1, t_1)(b_1, t_1') : t_1' - t_1 = 1\right\}.$$

We prove that the language L_1 cannot be recognized by any 0-clock pushdown timed automaton. To this end, we assume a 0-clock PTA A_0 where $L(A_0) = L_1$ and a computation π of A_0 such that

$$\pi = \langle q_{\text{init}}, \nu, \epsilon\rangle \Rightarrow^*\xrightarrow[a_1]{} \langle q_{a_1}, \nu_{a_1}, \_\rangle \Rightarrow^*\xrightarrow[b_1]{} \langle q_{b_1}, \nu_{b_1}, \_\rangle \Rightarrow^* \langle q_F, \nu_F, \_\rangle$$

and $\mathtt{tw}(\pi) = (a_1, t_1)(b_1, t_1') \in L_1$.

Claim: When A_0 reads b_1, there is a clock whose value is in $\mathbb{P}$.
We decompose π as follows:

$$\pi = \langle q_{\text{init}}, \nu, \epsilon\rangle \Rightarrow^*\overset{\delta}{\rightsquigarrow} \langle q''_{b_1}, \nu''_{b_1}, \_\rangle \xrightarrow[\epsilon]{}\overset{0.0}{\rightsquigarrow}\xrightarrow[\epsilon]{}\overset{0.0}{\rightsquigarrow}\xrightarrow[\epsilon]{} \cdots \overset{0.0}{\rightsquigarrow} \langle q'_{b_1}, \nu'_{b_1}, \_\rangle \xrightarrow[b_1]{} \langle q_{b_1}, \nu_{b_1}, \_\rangle \Rightarrow^* \langle q_F, \nu_F, \_\rangle$$

where $\delta > 0.0$. Our claim is rewritten as there is a clock x such that $\nu''_{b_1}(x) \in \mathbb{P}$. We show it by contradiction; assume there are no clocks x such that $\nu''_{b_1}(x) \in \mathbb{P}$.

- Since $\delta > 0.0$, we have $\nu''_{b_1}(x) \neq 0$. This and the assumption imply that there are no clocks x such that $\nu''_{b_1}(x) \in \mathbb{N}$. Thus, Proposition 3.1 implies that there is a positive real $0 < \delta' < 1$ such that

$$\langle q_{\mathrm{init}}, \nu, \epsilon \rangle \Rightarrow^{*} \overset{\delta+\delta'}{\rightsquigarrow} \langle q''_{b_1}, \nu''_{b_1} + \delta', \_ \rangle$$

  where $\langle q''_{b_1}, \nu''_{b_1}, \_ \rangle \cong \langle q''_{b_1}, \nu''_{b_1} + \delta', \_ \rangle$.

- Since $\langle q''_{b_1}, \nu''_{b_1}, \_ \rangle \cong \langle q''_{b_1}, \nu''_{b_1} + \delta', \_ \rangle$, there is a computation

$$\pi' = \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow^{*} \langle q''_{b_1}, \nu''_{b_1} + \delta', \_ \rangle \xrightarrow[\epsilon]{} \overset{0.0}{\rightsquigarrow} \xrightarrow[\epsilon]{} \overset{0.0}{\rightsquigarrow} \xrightarrow[\epsilon]{} \cdots \overset{0.0}{\rightsquigarrow} \xrightarrow[b_1]{} \boldsymbol{c} \Rightarrow^{*} \langle q_F, \nu'_F, \_ \rangle.$$

  Therefore, $\mathsf{tw}(\pi') = (a_1, t_1)(b_1, t'_1 + \delta') \in L(A_1)$.

- However, since $t'_1 - t_1 = 1$ and $0 < \delta' < 1$, $\mathsf{tw}(\pi') \notin L_1$.

The above argument states that, to recognize the language L_1, we need clocks at least one. Since there are no clocks on A_0, $L(A_0) \neq L_1$ and thus $L_1 \notin \mathbf{PTA}_0$. More formally, we can show the following based on the above argument:

> If $(a_1, t_1)(b_1, t'_1) \in L(A_0)$ and $t'_1 - t_1 = 1$,
> then $(a_1, u_1)(b_1, u'_1) \in L(A_0)$ with $1 < u'_1 - u_1 < 2$.

3.4.3 2-clock language

On the basis of the previous argument, we show that $L_2 \in \mathbf{PTA}_2$ but $L_2 \notin \mathbf{PTA}_1$ where L_2 is defined as follows

$$L_2 = \left\{ (a_2, t_2)(a_1, t_1)(b_1, t'_1)(b_2, t'_2) : t'_1 - t_1 = 1,\ t'_2 - t_2 = 2 \right\}.$$

It is clear that a 2-clock PTA can recognizes the timed language L_2.

To prove $L_2 \notin \mathbf{PTA}_1$, we consider the following proposition.

Proposition 3.6. If a 1-clock PTA A_1 accepts a timed word of L_2

$$(a_2, t_2)(a_1, t_1)(b_1, t'_1)(b_2, t'_2) \in L_2$$

that satisfies $0 < t'_2 - t'_1 < 1$, then there is another timed word

$$(a_2, u_2)(a_1, u_1)(b_1, u'_1)(b_2, u'_2) \in L(A_1)$$

with $1 < u'_1 - u_1 < 2$.

This immediately leads to $L_2 \notin \mathbf{PTA}_1$ because there is a timed word that satisfies the assumption of this proposition; for example $(a_2, 0.0)(a_1, 0.4)(b_1, 1.4)(b_2, 2.0)$.

Proof. We assume the following computation:

$$\pi = \begin{array}{l} \langle q_{\mathrm{init}}, \nu, \epsilon \rangle \Rightarrow^{*} \xrightarrow[a_2]{} \langle q_{a_2}, \nu_{a_2}, \_ \rangle \Rightarrow^{*} \xrightarrow[a_1]{} \\ \langle q_{a_1}, \nu_{a_1}, \_ \rangle \Rightarrow^{*} \xrightarrow[b_1]{} \langle q_{b_1}, \nu_{b_1}, \_ \rangle \Rightarrow^{*} \xrightarrow[b_2]{} \langle q_{b_2}, \nu_{b_2}, \_ \rangle \Rightarrow^{*} \langle q_F, \nu_F, \_ \rangle \end{array}$$

where q_F is an accepting location of A_1 and $\mathsf{tw}(\pi) = (a_2, t_2)(a_1, t_1)(b_1, t'_1)(b_2, t'_2) \in L_2$ with $0 < t'_2 - t'_1 < 1$.

Claim: When A_1 reads b_1, there is a clock whose fractional part is $t'_2 - t'_1$.
We decompose π as follows:

$$\langle q_{\mathrm{init}}, \nu, \epsilon\rangle \Rightarrow^* \langle q'_{b_1}, \nu'_{b_1}, \_\rangle \xrightarrow[b_1]{} \langle q_{b_1}, \nu_{b_1}, \_\rangle \underset{\theta}{\Rightarrow^*} \langle q'_{b_2}, \nu'_{b_2}, \_\rangle \xrightarrow[b_2]{} \langle q_{b_2}, \nu_{b_2}, \_\rangle \Rightarrow^* \langle q_F, \nu_F, \_\rangle$$

where $\theta = t'_2 - t'_1$.

We show $\mathit{frac}(\nu'_{b_1}(y)) = 1.0 - \theta$ for some clock y. On the basis of the argument for the 1-clock language and **Claim 1**, there is a clock y such that $\nu'_{b_2}(y) \in \mathbb{P}$. We have $0 < \theta < 1$ from the assumption for π. Proposition 3.5 implies $\mathit{frac}(\nu_{b_1}(y)) = 1 - \theta$ and $1 - \theta > 0$ implies $\mathit{frac}(\nu_{b'_1}(y)) = \mathit{frac}(\nu_{b_1}(y))$.

Now we decompose π as follows:

$$\pi = \langle q_{\mathrm{init}}, \nu, \epsilon\rangle \Rightarrow^* \overset{\delta}{\rightsquigarrow} \langle q'_{b_1}, \nu'_{b_1}, \_\rangle \xrightarrow[b_1]{} \langle q_{b_1}, \nu_{b_1}, \_\rangle \Rightarrow^* \langle q_F, \nu_F, \_\rangle.$$

There are no clocks y such that $\nu'_{b_1}(y) \in \mathbb{P}$ because the 1-clock pushdown automaton has only a single clock x and $\nu'_{b_1}(x) \notin \mathbb{P}$ on the basis of the above argument. Therefore, by the same argument for the 1-clock language L_1, there is δ' and a computation π' such that $0 < \delta' < 1$ and

$$\pi' = \langle q_{\mathrm{init}}, \nu, \epsilon\rangle \Rightarrow^* \overset{\delta+\delta'}{\rightsquigarrow} \langle q'_{b_1}, \nu'_{b_1} + \delta', \_\rangle \Rightarrow^* \langle q_F, \_, \_\rangle.$$

Since $\mathtt{tw}(\pi') = (a_2, t_2)(a_1, t_1)(b_1, t'_1 + \delta')(b_2, u_2)$ and $0 < \delta' < 1$, it finishes the proof. □

3.4.4 n-clock language

We can easily generalize the above argument for any n-clock language with $n > 2$.

We use the following timed language:

$$L_n = \left\{(a_n, t_n)(a_{n-1}, t_{n-1}) \dots (a_1, t_1)(b_1, t'_1) \dots (b_{n-1}, t'_{n-1})(b_n, t'_n) : t'_i - t_i = i\right\}.$$

It is clear that an n-clock PTA can recognizes the timed language L_n. On the basis of the above argument, we show the following.

Proposition 3.7. Let m be a natural number with $m < n$, and A_m be an m-clock pushdown timed automaton. If A_m accepts a timed word of L_n

$$(a_n, t_n)(a_{n-1}, t_{n-1}) \dots (a_1, t_1)(b_1, t'_1) \dots (b_{n-1}, t'_{n-1})(b_n, t'_n) \in L_n$$

that satisfies the following

- $\forall 1 \le i < n.\ 0 < t'_i - t'_n < 1$ and
- $\forall 1 \le i, j \le n.\ i \ne j \implies t'_i - t'_n \ne t'_j - t'_n$,

then there is another timed word

$$(a_n, u_n)(a_{n-1}, u_{n-1}) \dots (a_1, u_1)(b_1, u'_1) \dots (b_{n-1}, u'_{n-1})(b_n, u'_n) \in L(A_m)$$

such that $1 < u'_1 - u_1 < 2$.

Proof. We assume an accepting computation π that satisfies the assumption:

$$\mathtt{tw}(\pi) = (a_n, t_n)(a_{n-1}, t_{n-1}) \dots (a_1, t_1)(b_1, t'_1) \dots (b_{n-1}, t'_{n-1})(b_n, t'_n) \in L_n$$

When A_m reads b_i $(1 < i \le n)$, there is a clock whose fractional part is $t'_i - t'_1$. We decompose π as follows:

$$\langle q_{\text{init}}, \nu, \epsilon\rangle \Rightarrow^* \langle q'_{b_1}, \nu'_{b_1}, \_\rangle \xrightarrow[b_1]{} \langle q_{b_1}, \nu_{b_1}, \_\rangle \xRightarrow[t'_i - t'_1]{}{}^* \langle q'_{b_i}, \nu'_{b_i}, \_\rangle \xrightarrow[b_i]{} \langle q_{b_i}, \nu_{b_i}, \_\rangle \Rightarrow^* \langle q_F, \nu_F, \_\rangle.$$

On the basis of the argument for the language L_2, we can show that there is a clock x_i such that $\nu'_{b_i}(x_i) \in \mathbb{P}$. By Proposition 3.5, $\mathit{frac}(\nu_{b_n}(x_i)) = 1.0 - (t'_i - t'_1) > 0$ (here we use the assumption $0 < t'_i - t'_1 < 1$) and therefore $\mathit{frac}(\nu'_{b_n}(x_i)) = 1.0 - (t'_i - t'_1)$.

Let $\{y_1, y_2, \ldots, y_m\}$ be clocks of the m-clocks pushdown timed automaton A_m. We decompose π as follows:

$$\langle q_{\text{init}}, \nu, \epsilon\rangle \Rightarrow^* \overset{\delta}{\rightsquigarrow} \langle q'_{b_1}, \nu'_{b_1}, \_\rangle \xrightarrow[b_1]{} \langle q_{b_1}, \nu_{b_1}, \_\rangle \Rightarrow^* \langle q_F, \nu_F, \_\rangle.$$

The above argument requires $0 < \nu'_{b_1}(y_i) < 1$ (thus $\nu'_{b_1}(y_i) \notin \mathbb{P}$) for $1 \le i \le m$. Therefore, by the same argument for the languages L_1 and L_2, there is δ' and a computation π' such that $0 < \delta' < 1$ and

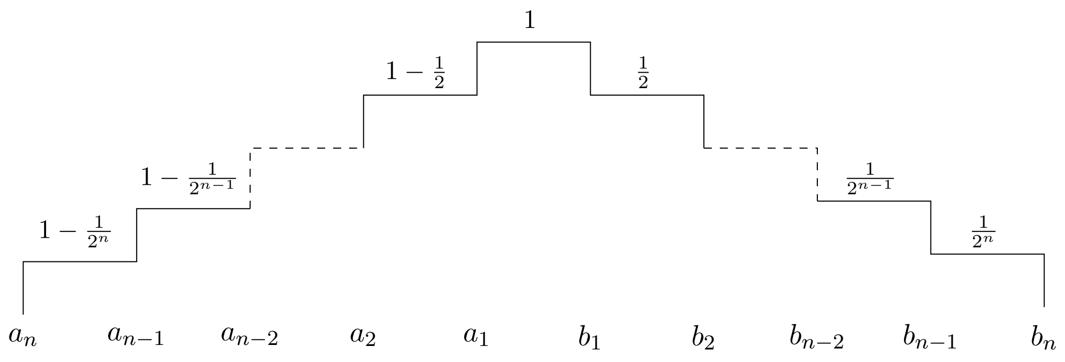
$$\pi' = \langle q_{\text{init}}, \nu, \epsilon\rangle \Rightarrow^* \overset{\delta+\delta'}{\rightsquigarrow} \langle q'_{b_1}, \nu'_{b_1} + \delta', \_\rangle \Rightarrow^* \langle q_F, \_, \_\rangle.$$

Since $\mathtt{tw}(\pi') = (a_n, t_n)(a_{n-1}, t_{n-1}) \ldots (a_1, t_1)(b_1, t'_1 + \delta') \ldots (b_{n-1}, u_{n-1})(b_n, u_n)$ and $0 < \delta' < 1$, it finishes the proof. □

This proposition immediately implies that L_n cannot be recognized by any m-clock PTA with $m < n$.

Theorem 3.4. For any positive natural number n, $L_n \in \mathbf{PTA}_n$ but $L_n \notin \mathbf{PTA}_{n-1}$.

Proof. We show that there is a timed word that satisfies the assumption of Proposition 3.7. It suffices to take the following timed word:



By Proposition 3.7, A_m accepts a timed word $(a_n, \_)(a_{n-1}, \_) \ldots (a_1, u)(b_1, u') \ldots (b_{n-1}, \_)(b_n, \_)$ where $1 < u' - u < 2$. Since this word does not belong to L_n, $L(A_m) \neq L_n$ holds. □

This theorem and the trivial property $\mathbf{PTA}_n \subseteq \mathbf{PTA}_{n+1}$ lead to the expressiveness hierarchy of pushdown timed automata.

Corollary 3.3.

$$\mathbf{PTA}_0 \subsetneq \mathbf{PTA}_1 \subsetneq \mathbf{PTA}_2 \subsetneq \cdots \subsetneq \mathbf{PTA}_n \subsetneq \cdots$$

Our proof does not depend on the existence of the stack, we can also show the same result for timed automata.

Corollary 3.4 ([PS12, HKW95])**.** Let $\mathbf{TA}_i$ be the language class of timed automata with i-clocks.

$$\mathbf{TA}_0 \subsetneq \mathbf{TA}_1 \subsetneq \mathbf{TA}_2 \subsetneq \cdots \subsetneq \mathbf{TA}_n \subsetneq \cdots$$

The same result was already known in the theory of timed automata [PS12, HKW95]. However, there is a little difference between their result and ours because they do not allow ϵ-transitions in their formulation of timed automata. Timed automata with ϵ-transitions are more expressive than timed automata without ϵ-transitions [BPDG98, BHR09].

3.5 Dense-Timed Pushdown Automata

We review the model of computation called *dense-timed pushdown automata* [AAS12a, CL15a]. Abdulla, Atig, and Stenman introduced dense-timed pushdown automata (DT-PDA) by extending (updatable) timed automata with a timed stack. Formally, a dense-timed pushdown automaton A is a 7-tuple $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ where

- Q is a finite set of control locations, $q_{\text{init}} \in Q$ is the initial location, $F \subseteq Q$ is a set of accepting locations;
- Σ is a finite input alphabet; Γ is a finite stack alphabet;
- $\mathcal{X}$ is a finite set of clocks;
- $\Delta \subseteq Q \times \Sigma_\epsilon \times Act_{\text{DTPDA}} \times Q$ is a finite set of transition rules.
  - To denote a transition rule $\langle p, \alpha, \tau, q\rangle \in \Delta$, we write $p \xrightarrow[\alpha]{\tau} q$.

Act_{DTPDA} is the set of actions of dense-timed pushdown automata defined by the following grammar:

$$Act_{\text{DTPDA}} ::= x \in_? I \mid x \leftarrow I \mid \mathsf{push}(\gamma, I) \mid \mathsf{pop}(\gamma, I) \mid \mathsf{nop}.$$

A configuration of DTPDA $\langle q, \nu, \xi\rangle$ is a triple of a location q, valuation $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$, and timed stack $\xi \in (\Gamma \times \mathbb{R}_{\geq 0})^*$. Since a configuration of PTA is a triple of a location q, valuation ν, and (untimed) stack $w \in \Gamma^*$, the timed stack is peculiar to DTPDA. The operational semantics of the DTPDA A is defined as an infinite labeled transition system $T_A = (Q \times (\mathcal{X} \to \mathbb{R}_{\geq 0}) \times (\Gamma \times \mathbb{R}_{\geq 0})^*, \to, \rightsquigarrow)$ where the set of discrete transitions $\to$ is defined as follows:

$$\frac{p \xrightarrow[\alpha]{x \in_? I} q \in \Delta \quad \nu \models x \in I}{\langle p, \nu, \xi\rangle \xrightarrow[\alpha]{x \in_? I} \langle q, \nu, \xi\rangle}, \qquad \frac{p \xrightarrow[\alpha]{x \leftarrow I} q \in \Delta \quad r \in I}{\langle p, \nu, \xi\rangle \xrightarrow[\alpha]{x \leftarrow I} \langle q, \nu[x := r], \xi\rangle},$$

$$\frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma, I)} q \in \Delta \quad r \in I}{\langle p, \nu, \xi\rangle \xrightarrow[\alpha]{\mathsf{push}(\gamma, I)} \langle q, \nu, \xi\langle\gamma, r\rangle\rangle}, \quad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma, I)} q \in \Delta \quad r \in I}{\langle p, \nu, \xi\langle\gamma, r\rangle\rangle \xrightarrow[\alpha]{\mathsf{pop}(\gamma, I)} \langle q, \nu, \xi\rangle}, \quad \frac{p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta}{\langle p, \nu, \xi\rangle \xrightarrow[\alpha]{\mathsf{nop}} \langle q, \nu, \xi\rangle},$$

and the set of timed transitions $\rightsquigarrow$ is defined as follows:

$$\frac{\delta \in \mathbb{R}_{\geq 0}}{\langle p, \nu, \xi\rangle \overset{\delta}{\rightsquigarrow} \langle p, \nu + \delta, \xi + \delta\rangle}$$

where $\xi + \delta$ is defined as follows:

$$\langle\gamma_1, r_1\rangle\langle\gamma_2, r_2\rangle \dots \langle\gamma_n, r_n\rangle + \delta \triangleq \langle\gamma_1, r_1 + \delta\rangle\langle\gamma_2, r_2 + \delta\rangle \dots \langle\gamma_n, r_n + \delta\rangle.$$

It should be noted that timed transitions modify all the values of a stack.

We define the language of the DTPDA A in the same way as PTA:

$$L(A) \triangleq \left\{ \mathtt{tw}(\pi) : \pi = \langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \overset{\delta_1}{\rightsquigarrow} \xrightarrow[\alpha_1]{\tau_1} \cdots \overset{\delta_n}{\rightsquigarrow} \xrightarrow[\alpha_n]{\tau_n} \langle q_F, \nu, \xi\rangle,\ q_F \in F \right\}.$$

Normalizing DTPDA. As we translated an updatable timed automaton to a timed automaton while preserving its language, we can also translate a DTPDA to a DTPDA without update operations while preserving its language.

Lemma 3.5. Let A be a DTPDA. There is a DTPDA B such that $L(A) = L(B)$ and $I = [0 : 0]$ if $p \xrightarrow[\alpha]{x \leftarrow I} q \in \Delta_B$ where Δ_B is the finite set of transition rules of B.

Proof. This proposition is shown in the same way as Theorem 2.5 of Chapter 1 that removes updatable operations from updatable timed automata. $\square$

Location Reachability Problem of DTPDA. We consider the location reachability problem of DTPDA. For a given location q, the location reachability problem $\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^*_? \langle q, {}^\exists\nu, {}^\exists\xi\rangle$ decides whether or not we can reach a configuration $\langle q, \nu, w\rangle$ with some valuation ν and timed stack ξ from the initial configuration $\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle$. Abdulla et al. showed that the location reachability problem of DTPDA is EXPTIME-complete [AAS12a].

In comparison with pushdown timed automata, the decidability proof of the location reachability problem of DTPDA is difficult due to the presence of timed stack. For example, let us suppose to apply the region abstraction technique to DTPDA and obtained the corresponding pushdown automata. We cannot naively reflect timed transitions in the obtained pushdown automata because we only modify the stack top element on ordinary pushdown automata. Since there are many other reasons why the classical region abstraction technique does not work well in DTPDA, in the next section, we will see another decidability proof of the location reachability problem of DTPDA based on the untiming theorem shown by Clemente and Lasota [CL15a]. In Chapter 5, we will show the decidability of the location reachability problem of DTPDA through the region abstraction technique.

3.6 Timed Pushdown Automata and Untiming Theorem

We define a model of computation called *timed pushdown automata* to accurately state the result of Clemente and Lasota in their paper [CL15a]. A timed pushdown automaton (TPDA) A is a 7-tuple $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ where the finite set of transition rules Δ only differs from that of DTPDA:

- $\Delta \subseteq Q \times \Sigma_\epsilon \times \mathit{Act}_{\text{TPDA}} \times Q$ is a finite set of transition rules.

To define the set of action of TPDA $\mathit{Act}_{\text{TPDA}}$, we introduce some notation. For a clock set X, we define a constraint formula φ generated from atomic propositions $x \in_? I$ and $x - y \in_? I$:

$$\varphi ::= x \in_? I \mid x - y \in_? I \mid \varphi \wedge \varphi$$

where $x, y \in X$. We write $\Phi(X)$ to denote the set of constraint formulae on the clock set X. For a valuation $\nu : X \to \mathbb{R}_{\geq 0}$ and constraint formula $\varphi \in \Phi(X)$, we define $\nu \models \varphi$ inductively: $\nu \models \varphi_1 \wedge \varphi_2$ if $\nu \models \varphi_1$ and $\nu \models \varphi_2$.

$\mathit{Act}_{\text{TPDA}}$ is the set of actions of pushdown timed automata defined by the following grammar:

$$\mathit{Act}_{\text{TPDA}} ::= \mathsf{check}(\varphi) \mid \mathsf{reset}(x) \mid \mathsf{push}(\gamma, I) \mid \mathsf{pop}(\gamma, \psi) \mid \mathsf{nop}$$

where $\varphi \in \Phi(\mathcal{X})$ and $\psi \in \Phi(\mathcal{X} \cup \{z\})$.

A configuration of TPDA $\langle q, \nu, \xi\rangle$ is a triple of a location q, valuation $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$, and timed stack $\xi \in (\Gamma \times (\{z\} \to \mathbb{R}_{\geq 0}))^*$. The set of configurations $Q \times (\mathcal{X} \to \mathbb{R}_{\geq 0}) \times (\Gamma \times (\{z\} \to \mathbb{R}_{\geq 0}))^*$ is isomorphic to that of DTPDA; therefore, to denote a configuration $\langle q, \nu, \langle\gamma_1, \{z \mapsto r_1\}\rangle \dots \langle\gamma_n, \{z \mapsto r_n\}\rangle\rangle$, we simply write $\langle q, \nu, \langle\gamma_1, r_1\rangle \dots \langle\gamma_n, r_n\rangle\rangle$.

Clemente and Lasota considered that a timed stack $\langle \gamma_1, \{z \mapsto r_1\} \rangle \dots \langle \gamma_n, \{z \mapsto r_n\} \rangle$ is a sequence of pairs of stack symbol γ_i and *local clock* called z. The formulation of TPDA does not allow update operations but allows diagonal constraints. The operational semantics of the TPDA A is defined as an infinite labeled transition system $T_A = (Q \times (\mathcal{X} \to \mathbb{R}_{\geq 0}) \times (\Gamma \times \mathbb{R}_{\geq 0})^*, \to, \rightsquigarrow)$ where the set of discrete transitions $\to$ is defined as follows:

$$\frac{p \xrightarrow[\alpha]{\mathsf{check}(\varphi)} q \in \Delta \quad \nu \models \varphi}{\langle p, \nu, \xi \rangle \xrightarrow[\alpha]{\mathsf{check}(\varphi)} \langle q, \nu, \xi \rangle}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta}{\langle p, \nu, \xi \rangle \xrightarrow[\alpha]{\mathsf{reset}(x)} \langle q, \nu[x := 0], \xi \rangle}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta}{\langle p, \nu, \xi \rangle \xrightarrow[\alpha]{\mathsf{nop}} \langle q, \nu, \xi \rangle},$$

$$\frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma, I)} q \in \Delta \quad r \in I}{\langle p, \nu, \xi \rangle \xrightarrow[\alpha]{\mathsf{push}(\gamma, I)} \langle q, \nu, \xi \langle \gamma, r \rangle \rangle}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma, \psi)} q \in \Delta \quad \nu \cup \{z \mapsto r\} \models \psi}{\langle p, \nu, \xi \langle \gamma, r \rangle \rangle \xrightarrow[\alpha]{\mathsf{pop}(\gamma, \psi)} \langle q, \nu, \xi \rangle},$$

and the set of timed transitions $\rightsquigarrow$ is defined as the same as DTPDA. We can also define the language $L(A)$ of a TPDA A as the same as DTPDA.

Clemente and Lasota showed the following normalization lemma [CL15a, CL15b].

Lemma 3.6. Let A be a TPDA. There is a TPDA B such that $L(A) = L(B)$ and $I = [0 : 0]$ for $p \xrightarrow[\alpha]{\mathsf{push}(\gamma, I)} q \in \Delta_B$ where Δ_B is the set of transition rules of the TPDA B.

This lemma enables the following important properties of normalized TPDA.

Proposition 3.8. Let $\langle q, \nu, \langle \gamma_1, r_1 \rangle \langle \gamma_2, r_2 \rangle \dots \langle \gamma_n, r_n \rangle \rangle$ be a reachable configuration on a normalized TPDA:

$$\langle q_{\text{init}}, \nu, \epsilon \rangle \Rightarrow^* \langle q, \nu, \langle \gamma_1, r_1 \rangle \langle \gamma_2, r_2 \rangle \dots \langle \gamma_n, r_n \rangle \rangle.$$

For each $i \in \{1, 2, \dots, n-1\}$, $r_{i+1} \geq r_i$.

On the basis of this proposition, Clemente and Lasota showed the untiming theorem of TPDA. It means we can remove all the local clocks in the timed stack of a TPDA while preserving its language.

Theorem 3.5 ([CL15a])**.** Let A be a TPDA. There is a PTA B such that

- $L(A) = L(B)$.
- The number of states and stack symbols of B is exponential in the size of A.
- The number of clocks of B is linear in the size of A.

Since the emptiness problem and location reachability problem of PTA is decidable, this theorem leads to the decidability of the emptiness problem and location reachability problem of TPDA.

Corollary 3.5. The emptiness problem and the location reachability problem of TPDA are EXPTIME-complete.

Proof. As with Corollary 3.2, since we can reduce the emptiness problem to location reachability problem in linear time and vice versa, it suffices to show the EXPTIME-completeness of the emptiness problem.

The EXPTIME-hardness of the emptiness problem of TPDA is immediately shown by Corollary 3.2 because any PTA is also TPDA.

We show the emptiness problem of TPDA is in EXPTIME. Let A be a TPDA and B be the corresponding PTA obtained by applying the above untiming theorem. On PTA, by the proof of Corollary 3.1, the emptiness problem can be solved in time linear in the number of states and stack symbols and exponential in the number of clocks. Therefore, the emptiness problem of B is solved in exponential time with respect to the size of A. $\square$

We also have the following result.

Corollary 3.6. PTA and DTPDA are equally expressive.

Proof. It is clear that the language class of DTPDA subsumes that of PTA: $\mathbf{PTA} \subseteq \mathbf{DTPDA}$. Now we assume a DTPDA A and construct a PTA such that $L(A) = L(B)$. By Lemma 3.5, we can normalize A into a DTPDA A' and then, A' can be seen as a TPDA. Therefore, there is a PTA corresponding to A' and we have $L(A) = L(B)$. This means that $\mathbf{PTA} \supseteq \mathbf{DTPDA}$. $\square$

Chapter 4

Timed Pushdown Automata with Multiple Local Clocks

We extend TPDA by the following features:

- Multiple local clocks; therefore, a configuration of MTPDA is of the form $\langle q, \nu, \langle \gamma_1, \mu_1 \rangle \dots \langle \gamma_n, \mu_n \rangle \rangle$ where $\mu_i : \mathscr{Z} \to \mathbb{R}_{\geq 0}$ is a clock valuation on the local clocks $\mathscr{Z}$ ($\mathscr{Z}$ is a finite set of local clocks). On the other hand, a configuration of TPDA is of the form $\langle q, \nu, \langle \gamma_1, r_1 \rangle \dots \langle \gamma_n, r_n \rangle \rangle$ where $r_i \in \mathbb{R}_{\geq 0}$.
- Resetting and checking local clocks by actions $\mathsf{reset}(z)$, $z \in_? I$ and $z - x \in_? I$ where z is a local clock. On DTPDA of Abdulla et al. and TPDA of Clemente and Lasota, such actions are not allowed.

We show the untiming theorem of MTPDA that removes local clocks from a given MTPDA. Alternatively, for a given MTPDA A, we can construct a PTA B such that $L(A) = L(B)$. These results will be shown as Theorem 4.3 and Corollary 4.1.

This chapter is based on the paper [Uez18].

Some Notation on Intervals In the present chapter, we consider intervals generated by integers as follows:

$$\begin{array}{ll}
(a : b) \triangleq \{r \in \mathbb{R} : a < r < b\}, & (a : b] \triangleq \{r \in \mathbb{R} : a < r \leq b\}, \\
[a : b) \triangleq \{r \in \mathbb{R} : a \leq r < b\}, & [a : b] \triangleq \{r \in \mathbb{R} : a \leq r \leq b\}, \\
(a : \omega) \triangleq \{r \in \mathbb{R} : a < r\}, & [a : \omega) \triangleq \{r \in \mathbb{R} : a \leq r\}, \\
(-\omega : a) \triangleq \{r \in \mathbb{R} : r < a\}, & (-\omega : a] \triangleq \{r \in \mathbb{R} : r \leq a\}
\end{array}$$

where $a, b \in \mathbb{Z}$.

It is clear that, for any interval I, there exists the unique interval J such that $a \in I \iff -a \in J$. For an interval I, we write $-I$ to denote the unique interval that satisfies $a \in I \iff -a \in -I$.

Let I and J be intervals. We write $I \sqsubset J$ if $a < b$ for any $a \in I$ and $b \in J$. For example, $(-\omega : 2) \sqsubset [2 : 3]$ but $(-\omega : 2] \not\sqsubset [2 : 3]$. We write $I \sqsubseteq J$ to denote $I = J$ or $I \sqsubset J$.

Let I be a non-empty interval ($I \neq \emptyset$). We define $I_\downarrow$ and $I_\uparrow$ as follows:

$$I_\downarrow \triangleq \bigcup_{J \sqsubset I} J, \quad I_\uparrow \triangleq \bigcup_{I \sqsubset J} J.$$

For any non-empty interval I, $I_{\downarrow}$ and $I_{\uparrow}$ are intervals since the following holds for these operators:

$$\begin{array}{ll} (-\omega : \_)_{\downarrow} = (-\omega : \_]_{\downarrow} = \emptyset, & (\_ : \omega)_{\uparrow} = [\_ : \omega)_{\uparrow} = \emptyset, \\ [a : \_]_{\downarrow} = [a : \_)_{\downarrow} = (-\omega : a), & [\_ : a]_{\uparrow} = (\_ : a]_{\uparrow} = (a : \omega), \\ (a : \_]_{\downarrow} = (a : \_)_{\downarrow} = (-\omega : a], & [\_ : a)_{\uparrow} = (\_ : a)_{\uparrow} = [a : \omega). \end{array}$$

We also use the following simple properties on intervals.

Proposition 4.1.

- Let I be an interval and $r_1 \leq r_2 \leq r_3$ be real numbers. If $r_1 \in I$ and $r_3 \in I$, then $r_2 \in I$.
- Let I be a non-empty interval. For any real number $r \in \mathbb{R}$, one of the following holds:
$$r \in I_{\downarrow} \qquad \text{or} \qquad r \in I \qquad \text{or} \qquad r \in I_{\uparrow}.$$
- For a non-empty interval I, we have $I_{\downarrow} \sqsubset I \sqsubset I_{\uparrow}$.

4.1 Formalization of Timed Pushdown Automata with Multiple Local Clocks

We define timed pushdown automata with multiple local clocks (MTPDA).

An MTPDA A is a 8-tuple $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta)$ where

- Q is a finite set of locations, $q_{\text{init}} \in Q$ is the initial location, $F \subseteq Q$ is a set of accepting locations;
- Σ is a finite input alphabet, Γ is a finite stack alphabet;
- $\mathcal{X}$ is a finite set of global clocks, $\mathcal{Z}$ is a finite set of local clocks;
  - $\mathcal{X}$ and $\mathcal{Z}$ are disjoint, i.e., $\mathcal{X} \cap \mathcal{Z} = \emptyset$.
- $\Delta \subseteq Q \times \Sigma_{\epsilon} \times \mathit{Act}_{\text{MTPDA}} \times Q$ is a finite set of transition rules. To denote a transition rule $\langle p, \alpha, \tau, q \rangle \in \Delta$, we write $p \xrightarrow[\alpha]{\tau} q$.
  - $\mathit{Act}_{\text{MTPDA}}$ is the set of actions of MTPDA and it is defined by the following grammar:
$$\begin{array}{rcl} \tau \in \mathit{Act}_{\text{MTPDA}} & ::= & \mathsf{push}(\gamma) \mid \mathsf{pop}(\gamma) \mid \mathsf{nop} \mid \mathsf{reset}(c) \\ & \mid & c \in_{?} I \mid c - c' \in_{?} I \end{array}$$
    where $\gamma \in \Gamma$, $c, c' \in \mathcal{X} \cup \mathcal{Z}$, and I is an interval.

We call an MTPDA A *K-TPDA* if A has K-local clocks $\{z_1, z_2, \ldots, z_K\}$. Let $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta)$ be a K-TPDA. A configuration of A, $\langle q, \nu, w \rangle$, is a triple of a location $q \in Q$, valuation $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$, and timed stack $w \in (\Gamma \times (\mathcal{Z} \to \mathbb{R}_{\geq 0}))^*$. We call a stack element $\langle \gamma, \mu \rangle$ *stack frame.* The initial configuration of A, $\langle q_{\text{init}}, \mathbf{0}, \epsilon \rangle$, is the triple of the initial location, 0-valued valuation, and empty stack.

We define the operational semantics of the K-TPDA A as an infinite labeled transition system $T_A = (Q \times (\mathcal{X} \to \mathbb{R}_{\geq 0}) \times (\Gamma \times (\mathcal{Z} \to \mathbb{R}_{\geq 0}))^*, \to, \rightsquigarrow)$. First, we define the set of (discrete) transitions $\to$ as follows:

$$\frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta}{\langle p, \nu, w \rangle \xrightarrow[\alpha]{\mathsf{push}(\gamma)} \langle q, \nu, w \, \langle \gamma, \mathbf{0} \rangle \rangle}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta}{\langle p, \nu, w \, \langle \gamma, \mu \rangle \rangle \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} \langle q, \nu, w \rangle}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta}{\langle p, \nu, w \rangle \xrightarrow[\alpha]{\mathsf{nop}} \langle q, \nu, w \rangle},$$

$$\frac{p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta \quad x \in \mathcal{X}}{\langle p, \nu, w\rangle \xrightarrow[\alpha]{\mathsf{reset}(x)} \langle q, \nu[x := 0], w\rangle,} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{reset}(z)} q \in \Delta \quad z \in \mathcal{Z}}{\langle p, \nu, w\langle \gamma, \mu\rangle\rangle \xrightarrow[\alpha]{\mathsf{reset}(z)} \langle q, \nu, w\langle \gamma, \mu[z := 0]\rangle\rangle,}$$

$$\frac{p \xrightarrow[\alpha]{x \in_? I} q \in \Delta \quad x \in \mathcal{X} \quad \nu \models x \in I}{\langle p, \nu, w\rangle \xrightarrow[\alpha]{x \in_? I} \langle q, \nu, w\rangle,} \qquad \frac{p \xrightarrow[\alpha]{z \in_? I} q \in \Delta \quad z \in \mathcal{Z} \quad \mu \models z \in I}{\langle p, \nu, w\langle \gamma, \mu\rangle\rangle \xrightarrow[\alpha]{x \in_? I} \langle q, \nu, w\langle \gamma, \mu\rangle\rangle,}$$

$$\frac{p \xrightarrow[\alpha]{c_1 - c_2 \in_? I} q \in \Delta \quad \{c_1, c_2\} \subseteq \mathcal{X} \quad \nu \models c_1 - c_2 \in I}{\langle p, \nu, w\rangle \xrightarrow[\alpha]{c_1 - c_2 \in_? I} \langle q, \nu, w\rangle,}$$

$$\frac{p \xrightarrow[\alpha]{c_1 - c_2 \in_? I} q \in \Delta \quad \{c_1, c_2\} \not\subseteq \mathcal{X} \quad \nu \cup \mu \models c_1 - c_2 \in I}{\langle p, \nu, w\,\langle \gamma, \mu\rangle\rangle \xrightarrow[\alpha]{c_1 - c_2 \in_? I} \langle q, \nu, w\,\langle \gamma, \mu\rangle\rangle.}$$

Next, we define the set of (timed) transitions $\rightsquigarrow$ as follows:

$$\frac{\delta \in \mathbb{R}_{\geq 0}}{\langle p, \nu, w\rangle \overset{\delta}{\rightsquigarrow} \langle p, \nu + \delta, w + \delta\rangle}$$

where $(\langle \gamma_1, \mu_1\rangle\langle \gamma_2, \mu_2\rangle \dots \langle \gamma_n, \mu_n\rangle) + \delta$ is defined as follows:

$$(\langle \gamma_1, \mu_1\rangle\langle \gamma_2, \mu_2\rangle \dots \langle \gamma_n, \mu_n\rangle) + \delta \triangleq \langle \gamma_1, \mu_1 + \delta\rangle\langle \gamma_2, \mu_2 + \delta\rangle \dots \langle \gamma_n, \mu_n + \delta\rangle.$$

As with timed automata and timed pushdown automata, we define the language of the K-TPDA A as follows:

$$L(A) \triangleq \left\{ \mathtt{tw}(\pi) : \pi = \langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \overset{\delta_1}{\rightsquigarrow} \xrightarrow[\alpha_1]{\tau_1} \cdots \overset{\delta_n}{\rightsquigarrow} \xrightarrow[\alpha_n]{\tau_n} \langle q_F, \nu, w\rangle,\ q_F \in F \right\}.$$

We also define the set of timed words, $L_\epsilon(A)$, that is accepted by a configuration whose location is an accepting location and stack is empty as follows:

$$L_\epsilon(A) \triangleq \left\{ \mathtt{tw}(\pi) : \pi = \langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \overset{\delta_1}{\rightsquigarrow} \xrightarrow[\alpha_1]{\tau_1} \cdots \overset{\delta_n}{\rightsquigarrow} \xrightarrow[\alpha_n]{\tau_n} \langle q_F, \nu, \epsilon\rangle,\ q_F \in F \right\}.$$

Proposition 4.2. Let A be an MTPDA. There is an MTPDA B such that $L(A) = L_\epsilon(B)$.

Proof. Let $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta)$. We construct the following MTPDA B:

$$B = (Q \cup \{q_B\}, q_{\text{init}}, \{q_B\}, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta')$$

where q_B is a fresh location and Δ' is defined as follows:

$$\Delta' = \Delta \cup \left\{ q_F \xrightarrow[\epsilon]{\mathsf{nop}} q_B : q_F \in F \right\} \cup \left\{ q_B \xrightarrow[\epsilon]{\mathsf{pop}(\gamma)} q_B : \gamma \in \Gamma \right\}.$$

By this construction, if there is a computation of A $\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \overset{\delta_1}{\rightsquigarrow} \xrightarrow[\alpha_1]{\tau_1} \cdots \overset{\delta_n}{\rightsquigarrow} \xrightarrow[\alpha_n]{\tau_n} \langle q_F, \nu, w\rangle$ where $q_F \in F$, then we also have the following computation of B:

$$\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \overset{\delta_1}{\rightsquigarrow} \xrightarrow[\alpha_1]{\tau_1} \cdots \overset{\delta_n}{\rightsquigarrow} \xrightarrow[\alpha_n]{\tau_n} \langle q_F, \nu, w\rangle \overset{\delta'_0}{\rightsquigarrow} \xrightarrow[\epsilon]{\mathsf{nop}} \langle q_B, \nu', w\rangle \overset{\delta'_1}{\rightsquigarrow} \xrightarrow[\epsilon]{\mathsf{pop}(\gamma_1)} \cdots \overset{\delta'_m}{\rightsquigarrow} \xrightarrow[\epsilon]{\mathsf{pop}(\gamma_m)} \langle q_B, \nu'', \epsilon\rangle.$$

This implies $L(A) \subseteq L_\epsilon(B)$. A similar property also implies $L_\epsilon(B) \subseteq L(A)$. □

On the basis of this proposition, hereafter we mainly consider the timed language accepted by accepting locations and the empty stack.

4.1.1 Compared to Timed Pushdown Automata

We recall the set of actions Act_{PTA} allowed on pushdown timed automata of Bouajjani et al.:

$$Act_{\mathrm{PTA}} ::= x \in_? I \mid \mathsf{reset}(x) \mid \mathsf{push}(\gamma) \mid \mathsf{pop}(\gamma) \mid \mathsf{nop}.$$

Since all the kinds of transition rules on PTA are allowed on MTPDA, the language class of MTPDA is greater than or equal to that of PTA. Furthermore, since the language classes of dense-timed pushdown automata **DTPDA** and **TPDA** equal to the language class of PTA **PTA**, we have the following.

Proposition 4.3.

$$\mathbf{PTA} = \mathbf{DTPDA} = \mathbf{TPDA} \subseteq \mathbf{MTPDA}.$$

We have a question whether or not **MTPDA** is strictly larger than **PTA**. Our main result of MTPDA, the untiming theorem of MTPDA (Theorem 4.3 and Corollary 4.1 of Section 4.7), states **PTA** = **MTPDA**.

4.1.2 Extending MTPDA by Useful Transition Rules

We consider useful extra actions on MTPDA and show that adding these rules to MTPDA does not enlarge the expressiveness of MTPDA.

$c \notin_? I$: A new action $c \notin_? I$ that checks whether or not the clock c does not belong to the interval I. The semantics of a transition rule $p \xrightarrow[\alpha]{c \notin_? I} q$ is defined as follows:

$$\frac{c \in \mathcal{X} \quad \nu \not\models c \in I}{\langle p, \nu, w\rangle \xrightarrow[\alpha]{c \notin_? I} \langle q, \nu, w\rangle}, \qquad \frac{c \in \mathcal{Z} \quad \mu \not\models c \in I}{\langle p, \nu, w\langle\gamma,\mu\rangle\rangle \xrightarrow[\alpha]{c \notin_? I} \langle q, \nu, w\langle\gamma,\mu\rangle\rangle}.$$

$\tau_1 \fatsemi \tau_2$: A new action $\tau_1 \fatsemi \tau_2$ that performs two actions τ_1 and τ_2 sequentially at a single transition without time-elapsings. The semantics of a transition rule $p \xrightarrow[\alpha]{\tau_1 \fatsemi \tau_2} q$ is defined as follows:

$$\frac{\langle p, \nu, w\rangle \xrightarrow[\alpha]{\tau_1} \langle p', \nu', w'\rangle \quad \langle p', \nu', w'\rangle \xrightarrow[\epsilon]{\tau_2} \langle p'', \nu'', w''\rangle}{\langle p, \nu, w\rangle \xrightarrow[\alpha]{\tau_1 \fatsemi \tau_2} \langle p'', \nu'', w''\rangle}$$

We also use the more general forms such as $p \xrightarrow[\alpha]{\tau_1 \fatsemi \tau_2 \fatsemi \cdots \fatsemi \tau_n} q$.

$\mathsf{check}(\gamma)$: A new action $\mathsf{check}(\gamma)$ that checks whether or not a stack is not empty and its top symbol is γ. The semantics of a transition rule $p \xrightarrow[\alpha]{\mathsf{check}(\gamma)} q$ is defined as follows:

$$\frac{}{\langle p, \nu, w\langle\gamma,\mu\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma)} \langle q, \nu, w\langle\gamma,\mu\rangle\rangle}$$

$\mathsf{check}(\epsilon)$: A new action $\mathsf{check}(\gamma)$ that checks whether or not a stack is empty. The semantics of a transition rule $p \xrightarrow[\alpha]{\mathsf{check}(\epsilon)} q$ is defined as follows:

$$\frac{}{\langle p, \nu, \epsilon\rangle \xrightarrow[\alpha]{\mathsf{check}(\epsilon)} \langle q, \nu, \epsilon\rangle}$$

rew(γ): A new action rew(γ) that rewrites the stack top symbol to γ. The semantics of a transition rule $p \xrightarrow[\alpha]{\mathsf{rew}(\gamma)} q$ is defined as follows:

$$\frac{}{\langle p, \nu, w\langle \gamma', \mu\rangle\rangle \xrightarrow[\alpha]{\mathsf{rew}(\gamma)} \langle q, \nu, w\langle \gamma, \mu\rangle\rangle}$$

Lemma 4.1. The above five types of transition rules $c \notin_? I$, $\tau_1 \fatsemi \tau_2$, $\mathsf{check}(\gamma)$, $\mathsf{check}(\epsilon)$, and $\mathsf{rew}(\gamma)$ do not enlarge the expressiveness of MTPDA.

Proof. First, we remove transition rules of the form $p \xrightarrow[\alpha]{c\notin_? I} q$. It is easily verified that, for any interval I, there are intervals $I_1, I_2, \ldots, I_n$ such that $\mathbb{R}_{\geq 0} \setminus I = \bigcup_{i=1}^{n} I_i$; for example, $\mathbb{R}_{\geq 0} \setminus (3:5] = [0:3] \cup (5:\omega)$. Following this property, we can replace a transition $p \xrightarrow[\alpha]{c\notin_? I} q$ by transitions $p \xrightarrow[\alpha]{c\in_? I_i} q$ for each $i = 1, \ldots, n$ such that $\mathbb{R}_{\geq 0} \setminus I = \bigcup_{i=1}^{n} I_i$ while preserving its language.

Next, we remove transition rules of the form $p \xrightarrow[\alpha]{\tau_1 \fatsemi \tau_2} q$. This can be removed in the same way as Proposition 2.2.

Finally, we remove $\mathsf{check}(\gamma)$, $\mathsf{check}(\epsilon)$, and $\mathsf{rew}(\gamma)$ simultaneously. Let $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta)$ be an MTPDA where Δ may contains the above three types of actions. We construct the following MTPDA B:

$$B = (Q^{\Gamma_\perp}, q_{\text{init}}^{\perp}, F^{\Gamma_\perp}, \Sigma, \Gamma_\perp, \mathcal{X}, \mathcal{Z}, \Delta')$$

where $\Gamma_\perp = \{\perp\} \cup \Gamma$, $Q^{\Gamma_\perp} = \{q^\chi : q \in Q, \chi \in \Gamma_\perp\}$, and $F^{\Gamma_\perp} = \left\{q_F^\chi : q_F \in F, \chi \in \Gamma_\perp\right\}$. We use a marked location $q^\perp$ to denote that the stack of a configuration is empty and q^γ to denote the stack top symbol is γ. Formally, we define a correspondence relation $\sim$ between configurations of A and B as follows:

$\langle q, \nu, \epsilon\rangle \sim \langle q^\perp, \nu, \epsilon\rangle$,
$\langle q, \nu, \langle \gamma, \mu\rangle\rangle \sim \langle q^\gamma, \nu, \langle \perp, \mu\rangle\rangle$,
$\langle q, \nu, \langle \gamma_1, \mu_1\rangle\langle \gamma_2, \mu_2\rangle \ldots \langle \gamma_n, \mu_n\rangle\langle \gamma_{n+1}, \mu_{n+1}\rangle\rangle \sim \langle q^{\gamma_{n+1}}, \nu, \langle \perp, \mu_1\rangle\langle \gamma_1, \mu_2\rangle \ldots \langle \gamma_{n-1}, \mu_n\rangle\langle \gamma_n, \mu_{n+1}\rangle\rangle$.

We define Δ' as follows to have the relation $\sim$ form a bisimulation between A and B:

If $p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta$, then we add $p^\chi \xrightarrow[\alpha]{\mathsf{push}(\chi)} q^\gamma$ to Δ' for each $\chi \in \Gamma_\perp$.

If $p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta$, then we add $p^\gamma \xrightarrow[\alpha]{\mathsf{pop}(\chi)} q^\chi$ to Δ' for each $\chi \in \Gamma_\perp$.

If $p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta$, then we add $p^\chi \xrightarrow[\alpha]{\mathsf{nop}} q^\chi$ to Δ' for each $\chi \in \Gamma_\perp$.

If $p \xrightarrow[\alpha]{\mathsf{reset}(c)} q \in \Delta$, then we add $p^\chi \xrightarrow[\alpha]{\mathsf{reset}(c)} q^\chi$ to Δ' for each $\chi \in \Gamma_\perp$.

If $p \xrightarrow[\alpha]{c\in_? I} q \in \Delta$, then we add $p^\chi \xrightarrow[\alpha]{c\in_? I} q^\chi$ to Δ' for each $\chi \in \Gamma_\perp$.

If $p \xrightarrow[\alpha]{c_1 - c_2 \in_? I} q \in \Delta$, then we add $p^\chi \xrightarrow[\alpha]{c_1 - c_2 \in_? I} q^\chi$ to Δ' for each $\chi \in \Gamma_\perp$.

If $p \xrightarrow[\alpha]{\mathsf{check}(\gamma)} q \in \Delta$, then we add $p^\gamma \xrightarrow[\alpha]{\mathsf{nop}} q^\gamma$ to Δ'.

If $p \xrightarrow[\alpha]{\mathsf{check}(\epsilon)} q \in \Delta$, then we add $p^\perp \xrightarrow[\alpha]{\mathsf{nop}} q^\perp$ to Δ'.

If $p \xrightarrow[\alpha]{\mathsf{rew}(\gamma)} q \in \Delta$, then we add $p^{\gamma'} \xrightarrow[\alpha]{\mathsf{nop}} q^\gamma$ to Δ' for each $\gamma' \in \Gamma$.

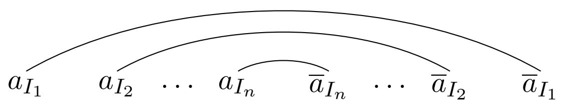
Since we can easily check the relation forms a bisimulation between A and B, we have $L(A) = L(B)$. □

4.1.3 Example of 1-TPDA

Let $\mathcal{I}$ be a finite set of intervals. Let us consider the following context-free language where each letter is indexed by an interval of $\mathcal{I}$:

$$L = \{a_{I_1}a_{I_2}\dots a_{I_n}\overline{a}_{I_n}\dots\overline{a}_{I_2}\overline{a}_{I_1} : I_i \in \mathcal{I}\}\,.$$

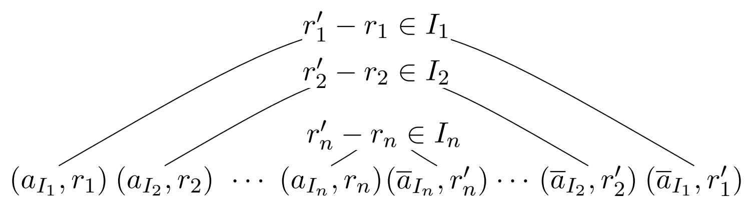
We can pair a_{I_i} and $\overline{a}_{I_i}$ for each i as follows:



By imposing a timing condition for each pair of a_{I_i} and $\overline{a}_{I_i}$, we consider the following timed language:

$$L_{\mathrm{ex}_1} \triangleq \left\{(a_{I_1}, r_1)\dots(a_{I_n}, r_n)(\overline{a}_{I_n}, r'_n)\dots(\overline{a}_{I_1}, r'_1) \ : \ I_i \in \mathcal{I},\ r'_i - r_i \in I_i\right\}.$$

This timed language requires that the elapsed timed between the corresponding symbols a_{I_i} and $\overline{a}_{I_i}$ belongs to I_i. It is depicted as follows:


$r'_1 - r_1 \in I_1$
$r'_2 - r_2 \in I_2$
$r'_n - r_n \in I_n$
(a_{I_1}, r_1) (a_{I_2}, r_2) $\cdots$ $(a_{I_n}, r_n)(\overline{a}_{I_n}, r'_n)$ $\cdots$ $(\overline{a}_{I_2}, r'_2)$ $(\overline{a}_{I_1}, r'_1)$

For example, the following timed word belongs to L_{ex_1}:

$$(a_{[2:3]}, 0.3)(a_{(0:1)}, 0.8)(\overline{a}_{(0:1)}, 1.7)(\overline{a}_{[2:3]}, 2.4) \in L_{\mathrm{ex}_1}$$

because $2.4 - 0.3 \in [2:3]$ and $1.7 - 0.8 \in (0:1)$.

The timed language L_{ex_1} is recognized by the following 1-TPDA where its stack alphabet is $\{Z\} \cup \mathcal{I}$ and a single local clock is z:

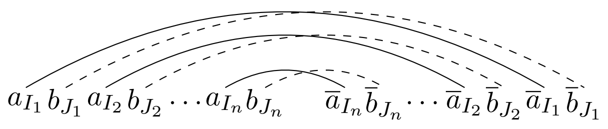
$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[r, "\mathsf{push}(Z)", "\epsilon"'] & q_1 \arrow[loop above, "a_I", "\mathsf{push}(I)"'] \arrow[r, "\mathsf{nop}", "\epsilon"'] & q_2 \arrow[loop above, "\overline{a}_I", "z \in_? I \fatsemi \mathsf{pop}(I)"'] \arrow[r, "\mathsf{pop}(Z)", "\epsilon"'] & \boxed{q_3}
\end{tikzcd}$$

(On this diagram, for the sake of simplicity, we write transition rules $q_1 \xrightarrow[a_I]{\mathsf{push}(I)} q_1$ and $q_2 \xrightarrow[\overline{a}_I]{z\in_? I \,\fatsemi\, \mathsf{pop}(a_I)} q_2$ parameterized by an interval I. Precisely speaking, we should write $q_1 \xrightarrow[a_{I_1}]{\mathsf{push}(I_1)} q_1$, $q_1 \xrightarrow[a_{I_2}]{\mathsf{push}(I_2)} q_1$ and so on for the finite set of intervals $\mathcal{I} = \{I_1, I_2, \dots\}$.)

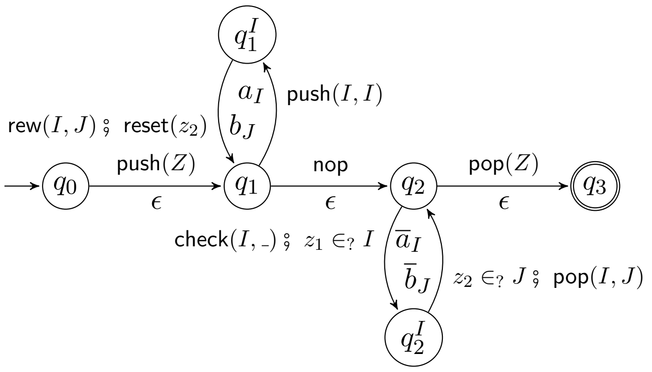
4.1.4 Example of 2-TPDA

We extend the above 1-TPDA language L_{ex_1} as follows:

$$L_{\mathrm{ex}_2} \triangleq \left\{ \begin{array}{l} (a_{I_1}, r_1)(b_{J_1}, r'_1)\dots(a_{I_n}, r_n)(b_{J_n}, r'_n) \\ (\overline{a}_{I_n}, \overline{r}_n)(\overline{b}_{J_n}, \overline{r'}_n)\dots(\overline{a}_{I_1}, \overline{r}_1)(\overline{b}_{J_1}, \overline{r'}_1) \\ : I_i \in \mathcal{I}, J_i \in \mathcal{I}, \overline{r}_i - r_i \in I_i, \overline{r'}_i - r'_i \in J_i \end{array} \right\}.$$


$a_{I_1}\, b_{J_1}\, a_{I_2}\, b_{J_2} \cdots a_{I_n} b_{J_n} \quad \overline{a}_{I_n} \overline{b}_{J_n} \cdots \overline{a}_{I_2} \overline{b}_{J_2} \overline{a}_{I_1} \overline{b}_{J_1}$

To accept this language, we consider 2-TPDA that includes two local clocks z_1 and z_2. We use z_1 and z_2 to represent the elapsed times after symbol a_{I_i} and b_{J_i} appear, respectively. When we read a_{I_i}, we push a new stack frame. After that, when we read b_{J_i}, we reset the clock z_2. Following this idea, we define 2-TPDA whose stack alphabet is $\{Z\} \cup (\mathcal{I} \times \mathcal{I})$ as follows:



A stack symbol (I, J) is pushed on to the stack after we read a_I and b_J. To achieve this, we take the following steps: (i) when we read a_I, we push (I, I) and then (ii) when we read b_J, we rewrite the second component by J. On this diagram, for the sake of simplicity, we write parameterized transition rules such as $\mathsf{push}(I, I)$; therefore, in practice, we instantiate these transition rules by concrete intervals.

4.1.5 Important Property of MTPDA: Monotonicity in Stack

We define an important notion and property of MTPDA.

Definition 4.1 (Monotonically Decreasing Stack)**.** Let $\langle q, \nu, w \rangle$ be a configuration of an MTPDA.

The stack w is a *monotonically decreasing stack* if one of the following holds:

- w is a non-empty stack, $w = \langle \gamma_1, \mu_1 \rangle \dots \langle \gamma_n, \mu_n \rangle$, and $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_n$.
- w is the empty stack.

■

Discrete and timed transitions preserve the monotonicity of monotonically decreasing stacks.

Proposition 4.4. Let $\langle q, \nu, w \rangle$ be a configuration where w is monotonically decreasing stack.

- If $\langle q, \nu, w \rangle \xrightarrow[\alpha]{\tau} \langle q', \nu', w' \rangle$, then w' is also a monotonically decreasing stack.
- If $\langle q, \nu, w \rangle \overset{\delta}{\rightsquigarrow} \langle q, \nu', w' \rangle$, then w' is also a monotonically decreasing stack.

Proof. First, we consider timed transitions: $\langle q, \nu, \langle \gamma_1, \mu_1 \rangle \dots \langle \gamma_n, \mu_n \rangle \rangle \overset{\delta}{\rightsquigarrow} \langle q, \nu + \delta, \langle \gamma_1, \mu_1 + \delta \rangle \dots \langle \gamma_n, \mu_n + \delta \rangle \rangle$. We need to show $\mu_i + \delta \geq \mu_{i+1} + \delta$. It is clear from $\mu_i \geq \mu_{i+1}$.

Next, we consider the following two types of discrete transitions because our claim is trivial for the other types of discrete transitions:

$$(1) : \langle q, \nu, \langle \gamma_1, \mu_1 \rangle \dots \langle \gamma_n, \mu_n \rangle \rangle \xrightarrow[\alpha]{\mathsf{push}(\gamma)} \langle q, \nu, \langle \gamma_1, \mu_1 \rangle \dots \langle \gamma_n, \mu_n \rangle \langle \gamma, \mathbf{0} \rangle \rangle,$$

$$(2) : \langle q, \nu, \langle \gamma_1, \mu_1 \rangle \dots \langle \gamma_{n-1}, \mu_{n-1} \rangle \langle \gamma_n, \mu_n \rangle \rangle \xrightarrow[\alpha]{\mathsf{reset}(z)} \langle q, \nu, \langle \gamma_1, \mu_1 \rangle \dots \langle \gamma_{n-1}, \mu_{n-1} \rangle \langle \gamma_n, \mu_n[z \coloneqq 0] \rangle \rangle.$$

For the case (1), it suffices to show $\mu_n \geq \mathbf{0}$ and it is clear.
For the case (2), it suffices to show $\mu_{n-1}(z) \geq \mu_n[z \coloneqq 0](z)$ and it is clear. □

By this proposition, a stack of a configuration that is reachable from the initial configuration is monotonically decreasing.

Proposition 4.5. Let $\langle q, \nu, w\rangle$ be a configuration.
If $\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \langle q, \nu, w\rangle$, then the stack w is a monotonically decreasing stack.

We will use this proposition to show an important lemmas, Lemma 4.8 and 4.10, in Section 4.4.2.

4.2 Proof Outline of Untiming Theorem of MTPDA

The main result of this chapter is the following (Corollary 4.1 in Section 4.7):

Untiming Theorem of MTPDA.
Let A be an MTPDA. There is a PTA B such that $L(A) = L(B)$.

We prove this by the following step:

Section 4.3 For a given MTPDA, while preserving its language, we remove transition rules of the form $p \xrightarrow[\alpha]{x-x'\in_? I} q$, $p \xrightarrow[\alpha]{z-z'\in_? I} q$, $p \xrightarrow[\alpha]{z\in_? I} q$, and $p \xrightarrow[\alpha]{x-z\in_? I}$ where $z, z' \in \mathcal{Z}$ are local clocks and $x, x' \in \mathcal{X}$ are global clocks.

Section 4.4–4.6 For a given MTPDA, while preserving its language, we remove transition rules of the form $p \xrightarrow[\alpha]{z-x\in_? I} q$ where $z \in \mathcal{Z}$ is a local clock and $x \in \mathcal{X}$ is a global clock.

Section 4.7 Through Section 4.4–4.6, we can remove transition rules that inspect clocks except rules of the form $p \xrightarrow[\alpha]{x\in_? I} q$ while preserving the language of a given MTPDA. Alternatively, we can only inspect global clocks by constraints of the form $x \in_? I$; therefore, the presence of local clocks does not affect computations of the given MTPDA. Following this argument, we prove our main result, Corollary 4.1.

4.3 Removing Transition Rules with Actions $x - x' \in_? I$, $z - z' \in_? I$, $z \in_? I$, or $x - z \in_? I$

Lemma 4.2. Let A be an MTPDA. There is an MTPDA B that satisfies the following:

- $L(A) = L(B)$.
- There are no transition rules of the form $p \xrightarrow[\alpha]{z\in_? I} q$ where z is a local clock.

Proof. We assume $A = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta)$ and construct the following MTPDA B:

$$B = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma, \mathcal{X} \cup \{y\}, \mathcal{Z}, \Delta')$$

where y is a fresh global clock and Δ' is defined as follows:

$$\frac{p \xrightarrow[\alpha]{z\in_? I} q \in \Delta}{p \xrightarrow[\alpha]{\mathsf{reset}(y) ⨾\ z-y\in_? I} q \in \Delta'}, \qquad \frac{p \xrightarrow[\alpha]{\tau} q \in \Delta \quad \tau \not\equiv z \in_? I}{p \xrightarrow[\alpha]{\tau} q \in \Delta'}.$$

It is clear that $L(A) = L(B)$. □

It should be noted that the proof of this lemma inserts some transition rules using $z - y \in_? I$ where z is a local clock and y is a global clock.

Lemma 4.3. Let A be an MTPDA. There is an MTPDA B that satisfies the following:

- $L(A) = L(B)$.
- There are no transition rules of the form $p \xrightarrow[\alpha]{x-z\in_? I} q$ where x and z are global and local clocks, respectively.

Proof. We assume $A = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta)$ and construct the following MTPDA B:

$$B = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta')$$

where y is a fresh global clock and Δ' is defined as follows:

$$\frac{p \xrightarrow[\alpha]{x-z\in_? I} q \in \Delta}{p \xrightarrow[\alpha]{z-x\in_? -I} q \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{\tau} q \in \Delta \quad \tau \not\equiv x - z \in_? I}{p \xrightarrow[\alpha]{\tau} q \in \Delta'.}$$

It is clear that $L(A) = L(B)$. ☐

Again, it should be noted that the proof of this lemma inserts transition rules using $z - x \in_? I$ where z is a local clock and x is a global clock.

Lemma 4.4. Let A be an MTPDA. There is an MTPDA B that satisfies the following:

- $L(A) = L(B)$.
- There are no transition rules of the form $p \xrightarrow[\alpha]{x_1-x_2\in_? I} q$.

Proof. We can construct an MTPDA B that satisfies the condition in the same construction as Theorem 2.4 of Chapter 2. Recall that Theorem 2.4 removes diagonal constraints from a timed automaton while preserving its language. ☐

Lemma 4.5. Let A be an MTPDA. There is an MTPDA B that satisfies the following:

- $L(A) = L(B)$.
- There are no transition rules of the form $p \xrightarrow[\alpha]{z_1-z_2\in_? I} q$ where z_1 and z_2 are local clocks.

Proof. Although we can show this in the same way as Theorem 2.4 of Chapter 2, we construct an MTPDA B that satisfies the condition just for the sake of completeness.

We assume $A = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta)$ and fix an action $z_1 - z_2 \in I$. We remove transition rules with the action $z_1 - z_2 \in I$ by constructing the following MTPDA B:

$$B = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma \times \{\mathsf{tt}, \mathsf{ff}\}, \mathcal{X}, \mathcal{Z}, \Delta').$$

Before defining Δ', we define a relation between configurations of A and B:

$$\begin{aligned} & \langle q, \nu, \langle \gamma_1, \mu_1 \rangle \dots \langle \gamma_n, \mu_n \rangle \rangle \sim \langle q, \nu, \langle (\gamma_1, b_1), \mu_1 \rangle \dots \langle (\gamma_n, b_n), \mu_n \rangle \rangle \\ \overset{\mathrm{def}}{\iff} \quad & b_i = \mathsf{tt} \iff \mu_i \models z_1 - z_2 \in I. \end{aligned}$$

We define Δ' to have the above relation form a bisimulation between A and B:

$$\frac{p \xrightarrow[\alpha]{z_1-z_2\in_? I} q \in \Delta}{p \xrightarrow[\alpha]{\mathsf{check}(\langle \gamma, \mathsf{tt} \rangle)} q \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{c_1-c_2\in_? J} q \in \Delta \quad c_1 - c_2 \in_? J \not\equiv z_1 - z_2 \in_? I}{p \xrightarrow[\alpha]{c_1-c_2\in_? J} q \in \Delta',}$$

$$\frac{p \xrightarrow[\alpha]{c \in_? I} q \in \Delta}{p \xrightarrow[\alpha]{c \in_? I} q \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta \quad b \in \{\mathsf{tt}, \mathsf{ff}\}}{p \xrightarrow[\alpha]{\mathsf{pop}(\langle \gamma, b \rangle)} q \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta}{p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta',}$$

$$\frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta \quad 0.0 \in I}{p \xrightarrow[\alpha]{\mathsf{push}(\langle \gamma, \mathsf{tt} \rangle)} q \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta \quad 0.0 \notin I}{p \xrightarrow[\alpha]{\mathsf{push}(\langle \gamma, \mathsf{ff} \rangle)} q \in \Delta',}$$

$$\frac{p \xrightarrow[\alpha]{\mathsf{reset}(c)} q \in \Delta \quad c \neq z_1 \quad c \neq z_2}{p \xrightarrow[\alpha]{\mathsf{reset}(c)} q \in \Delta',}$$

$$\frac{p \xrightarrow[\alpha]{\mathsf{reset}(z_1)} q \in \Delta \quad b \in \{\mathsf{tt}, \mathsf{ff}\}}{p \xrightarrow[\alpha]{\mathsf{reset}(z_1) \fatsemi\, z_2 \in_? -I \fatsemi\, \mathsf{check}(\langle \gamma, b \rangle) \fatsemi\, \mathsf{rew}(\langle \gamma, \mathsf{tt} \rangle)} q \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{reset}(z_1)} q \in \Delta \quad b \in \{\mathsf{tt}, \mathsf{ff}\}}{p \xrightarrow[\alpha]{\mathsf{reset}(z_1) \fatsemi\, z_2 \notin_? -I \fatsemi\, \mathsf{check}(\langle \gamma, b \rangle) \fatsemi\, \mathsf{rew}(\langle \gamma, \mathsf{ff} \rangle)} q \in \Delta',}$$

$$\frac{p \xrightarrow[\alpha]{\mathsf{reset}(z_2)} q \in \Delta \quad b \in \{\mathsf{tt}, \mathsf{ff}\}}{p \xrightarrow[\alpha]{\mathsf{reset}(z_2) \fatsemi\, z_1 \in_? I \fatsemi\, \mathsf{check}(\langle \gamma, b \rangle) \fatsemi\, \mathsf{rew}(\langle \gamma, \mathsf{tt} \rangle)} q \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{reset}(z_2)} q \in \Delta \quad b \in \{\mathsf{tt}, \mathsf{ff}\}}{p \xrightarrow[\alpha]{\mathsf{reset}(z_2) \fatsemi\, z_1 \notin_? I \fatsemi\, \mathsf{check}(\langle \gamma, b \rangle) \fatsemi\, \mathsf{rew}(\langle \gamma, \mathsf{ff} \rangle)} q \in \Delta'.}$$

It can be easily verified that the relation $\sim$ forms a bisimulation between A and B in the same argument as the proof of Theorem 2.4. □

We summarize the above lemmas as the following lemma.

Lemma 4.6. Let A be an MTPDA. There is an MTPDA B that satisfies the following:

- $L(A) = L(B)$.
- There are no transition rules with actions of the form $z \in_? I$, $x - z \in_? I$, $x_1 - x_2 \in_? I$, and $z_1 - z_2 \in_? I$.

4.4 Predicting MTPDA: Preliminary to Remove $z - x \in_? I$

We construct a type of MTPDA called *predicting* MTPDA from an MTPDA.

On predicting MTPDA, instead of pushing a new stack frame onto a stack by $\mathsf{push}(\gamma)$, we push a frame with a predicting interval J that is either one of $I_\downarrow$, I, or $I_\uparrow$ by $\mathsf{push}((\gamma, J))$. Furthermore, instead of popping a stack frame together with a predicting interval J, we pop a frame by an action $x - z \in_? J \fatsemi \mathsf{pop}((\gamma, J))$ while checking whether or not our prediction J is correct. This is depicted as follows:

$$\langle p_1, \nu, \begin{array}{|c|}\hline w \\ \hline\end{array} \rangle \xrightarrow[\alpha]{\mathsf{push}(\gamma)} \langle p_2, \nu, \begin{array}{|c|}\langle (\gamma, J), \mathbf{0} \rangle \\ \hline w \\ \hline\end{array} \rangle \to^* \langle p_3, \nu', \begin{array}{|c|}\langle (\gamma, J), \mu \rangle \\ \hline w' \\ \hline\end{array} \rangle \xrightarrow{z - x \in_? J \fatsemi\, \mathsf{pop}((\gamma, J))} \langle p_4, \nu', \begin{array}{|c|}\hline w' \\ \hline\end{array} \rangle.$$

We nondeterministically push $\mathsf{push}((\gamma, I_\downarrow))$, $\mathsf{push}((\gamma, I))$, and $\mathsf{push}((\gamma, I_\uparrow))$; therefore, the constructed predicting MTPDA accepts the language of the original MTPDA (Lemma 4.7).

Predicting MTPDA is useful to analyze the structure of stack and we obtain an important property Lemma 4.8. On the basis of the construction of predicting MTPDA and this lemma, in the following two sections, we will remove transitions with $z - x \in_? I$.

4.4.1 Predicting MTPDA

Let $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta)$ be an MTPDA. From the MTPDA, we construct the following predicting MTPDA B:

$$B = (Q, q_{\text{init}}, F, \Sigma, \Gamma \times \{I_\downarrow, I, I_\uparrow\}, \mathcal{X}, \mathcal{Z}, \Delta').$$

The set of transition rules Δ' is defined as follows:

$$\frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta \quad J \in \{I_\downarrow, I, I_\uparrow\}}{p \xrightarrow[\alpha]{\mathsf{push}(\gamma,J)} q \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta}{p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J) ⨾\ z-x\in_? J ⨾\ \mathsf{pop}(\gamma,J)} q \in \Delta',}$$

$$\frac{p \xrightarrow[\alpha]{\tau} q \in \Delta \quad \tau \neq \mathsf{push}(\gamma) \quad \tau \neq \mathsf{pop}(\gamma)}{p \xrightarrow[\alpha]{\tau} q \in \Delta'.}$$

Remark: For the sake of simplicity, we simply write $\mathsf{push}(\gamma, J)$, $\mathsf{pop}(\gamma, J)$, $\mathsf{check}(\gamma, J)$, and $\mathsf{rew}(\gamma, J)$ instead of $\mathsf{push}((\gamma, J))$, $\mathsf{pop}((\gamma, J))$, $\mathsf{check}((\gamma, J))$, and $\mathsf{rew}((\gamma, J))$, respectively.

We can show the obtained MTPDA B is equivalent to the original MTPDA A.

Lemma 4.7. $L(A) = L(B)$ and $L_\epsilon(A) = L_\epsilon(B)$.

Proof. We show the following relation $\sim$ on the configurations of A and B forms a bisimulation between A and B:

$$\langle q, \nu, \langle \gamma_1, \mu_1\rangle\langle \gamma_2, \mu_2\rangle \ldots \langle \gamma_n, \mu_n\rangle\rangle \ \sim\ \langle q, \nu, \langle (\gamma_1, J_1), \mu_1\rangle\langle (\gamma_2, J_2), \mu_2\rangle \ldots \langle (\gamma_n, J_n), \mu_n\rangle\rangle.$$

First, to show $L(A) \subseteq L(B)$ and $L_\epsilon(A) \subseteq L_\epsilon(B)$, we consider the following diagram:

$$\begin{tikzcd}
\langle p,\nu,w_A\rangle \arrow[r, "\tau", "\alpha"'] & \langle q,\nu',w'_A\rangle & & \langle p,\nu,w_A\rangle \arrow[r, "\tau", "\alpha"'] & \langle q,\nu',w'_A\rangle \\
& \wr & \Longrightarrow & \wr & \wr \\
& \langle q,\nu',w'_B\rangle & & \langle p,\nu,w_B\rangle \arrow[r, "\tau'", "\alpha"'] & \langle q,\nu',w'_B\rangle.
\end{tikzcd}$$

Although we also need the same diagram for timed transitions, we omit it because the diagram immediately holds.

We proceed by case analysis on τ. Here we only consider the case $\tau = \mathsf{pop}(\gamma)$. The other cases are trivial from the definition. Let us consider the following diagram:

$$\begin{tikzcd}
\langle p,\nu,w\langle\gamma,\mu\rangle\rangle \arrow[r, "\mathsf{pop}(\gamma)", "\alpha"'] & \langle q,\nu,w\rangle \\
& \wr \\
& \langle q,\nu,w_B\rangle.
\end{tikzcd}$$

By Proposition 4.1, there is an interval $J \in \{I_\downarrow, I, I_\uparrow\}$ such that $\mu(z) - \nu(x) \in J$; therefore, we have the following and we can show $L(A) \subseteq L(B)$ and $L_\epsilon(A) \subseteq L_\epsilon(B)$.

$$\begin{tikzcd}
\langle p,\nu,w\langle\gamma,\mu\rangle\rangle \arrow[r, "\mathsf{pop}(\gamma)", "\alpha"'] & \langle q,\nu,w\rangle \\
\wr & \wr \\
\langle p,\nu,w\langle(\gamma,J),\mu\rangle\rangle \arrow[r, "\mathsf{check}(\gamma{,}J) ⨾\ z-x\in_? J ⨾\ \mathsf{pop}(\gamma{,}J)", "\alpha"'] & \langle q,\nu,w_B\rangle.
\end{tikzcd}$$

Next, to show $L(B) \subseteq L(A)$ and $L_\epsilon(B) \subseteq L_\epsilon(A)$, we consider the following diagram:

$$\begin{tikzcd}
\langle p,\nu,w_A\rangle & & & \langle p,\nu,w_A\rangle \arrow[r, "\tau'", "\alpha"'] & \langle q,\nu',w'_A\rangle \\
\wr & & \Longrightarrow & \wr & \wr \\
\langle p,\nu,w_B\rangle \arrow[r, "\tau", "\alpha"'] & \langle q,\nu',w'_B\rangle & & \langle p,\nu,w_B\rangle \arrow[r, "\tau", "\alpha"'] & \langle q,\nu',w'_B\rangle.
\end{tikzcd}$$

We also need the same diagram for timed transitions $\rightsquigarrow$. Since we can easily show these diagrams easily by case analysis on τ, we omit the proof. $\square$

4.4.2 Stack Structure of Predicting MTPDA without $\mathsf{reset}(z)$

Let A be an MTPDA such that it does not have a transition rule including an action $\mathsf{reset}(z)$. Here we show the following important properties of the predicting MTPDA obtained from A.

Lemma 4.8. Let B be the predicting MTPDA obtained from A and π be an accepting computation $\pi = \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \langle q_F, \nu', \epsilon\rangle$.

For the computation π, we consider a predicting stack w that appears in π, i.e.,

$$\pi = \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \langle q, \nu, w\rangle \Rightarrow^* \langle q_F, \nu', \epsilon\rangle.$$

We have $w \in \Upsilon(I_{\uparrow})^* \cdot \Upsilon(I)^* \cdot \Upsilon(I_{\downarrow})^*$ where Υ is defined as follows:

$$\Upsilon(J) \triangleq (\Gamma \times \{J\}) \times (\mathcal{Z} \to \mathbb{R}_{\geq 0}).$$

To show this lemma, first we show the following.

Lemma 4.9. Let C be an MTPDA. We consider a computation from the initial configuration

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \langle p, \nu, \ldots \langle \gamma_m, \mu_m\rangle \ldots \langle \gamma_n, \mu_n\rangle\rangle \underset{\star}{\Rightarrow}^* \langle p', \nu', \ldots \langle \gamma_m, \mu'_m\rangle\rangle$$

that satisfies the following conditions

- among $\underset{\star}{\Rightarrow}^*$, there are no configurations whose stack height is less than m; (alternatively, among $\underset{\star}{\Rightarrow}^*$, the frame including γ_m have not been popped.)
- Furthermore, there are no transitions that include $\mathsf{reset}(z)$ in $\underset{\star}{\Rightarrow}^*$.

For such a computation, we have $\mu_n(z) - \nu(x) \leq \mu'_m(z) - \nu'(x)$.

Proof. By Proposition 4.5, we have $\mu_n \leq \mu_m$. Let δ be the elapsed time among $\underset{\star}{\Rightarrow}^*$. Since our assumption that there are no transition rules with $\mathsf{reset}(z)$, $\mu'_m(z) = \mu_m(z) + \delta$ holds. For the global clock x, we have $\nu'(x) \leq \nu(x) + \delta$. Combining them, we have the following:

$$\begin{array}{rcl} \mu_n(z) - \nu(z) & \leq & \mu_m(z) - \nu(z) \\ & = & (\mu_m(z) + \delta) - (\nu(x) + \delta) \\ & \leq & \mu'_m(z) - \nu'(x). \end{array}$$

$\square$

Now we prove Lemma 4.8 based on the above lemma.

Proof of Lemma 4.8. We decompose π as follows:

$$\begin{array}{l} \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \\ \langle q, \nu, w = \langle(\gamma_1, J_1), \mu_1\rangle \ldots \langle(\gamma_i, J_i), \mu_i\rangle\langle(\gamma_j, J_j), \mu_j\rangle \ldots \langle(\gamma_k, J_k), \mu_k\rangle\rangle \Rightarrow^* \\ \langle q_1, \nu_1, \langle(\gamma_1, J_1), \mu'_1\rangle \ldots \langle(\gamma_i, J_i), \mu'_i\rangle\langle(\gamma_j, J_j), \mu'_j\rangle\rangle \xrightarrow[\alpha]{\mathsf{pop}(\gamma_j)} \\ \langle q_2, \nu_2, \langle(\gamma_1, J_1), \mu'_1\rangle \ldots \langle(\gamma_i, J_i), \mu'_i\rangle\rangle \underset{\star}{\Rightarrow}^* \\ \langle q_3, \nu_3, \langle(\gamma_1, J_1), \mu''_1\rangle \ldots \langle(\gamma_i, J_i), \mu''_i\rangle\rangle \xrightarrow[\alpha']{\mathsf{pop}(\gamma_i)} \langle q_4, \nu_4, \langle(\gamma_1, J_1), \mu''_1\rangle \ldots\rangle \Rightarrow^* \langle q_F, \nu', \epsilon\rangle. \end{array}$$

Since there are no transition rules with $\mathsf{reset}(z)$ in the MTPDA A, we can apply Lemma 4.9 and have $\mu'_j(z) - \nu_1(x) \leq \mu''_i(z) - \nu_3(x)$. Furthermore, the presence of the transitions $\xrightarrow[\alpha]{\mathsf{pop}(\gamma_j)}$

and $\xrightarrow[\alpha]{\mathsf{pop}(\gamma_i)}$ implies $\mu'_j(z) - \nu_1(x) \in J_j$ and $\mu''_i(z) - \nu_3(x) \in J_i$, respectively. Therefore, we have $J_j \sqsubseteq J_i$.

This argument states $J_1 \sqsupseteq J_2 \sqsupseteq \cdots \sqsupseteq J_i \sqsupseteq J_j \sqsupseteq \cdots \sqsupseteq J_k$ and therefore $w \in \Upsilon(I_\uparrow)^* \cdot \Upsilon(I)^* \cdot \Upsilon(I_\downarrow)^*$. □

Lemma 4.9 also leads to the following useful property.

Lemma 4.10. Let us consider the following computation of the MTPDA A:

$$\begin{aligned}
&\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \\
&\langle q_1, \nu, \dots \langle \gamma_h, \mu_h\rangle \dots \langle \gamma_i, \mu_i\rangle \dots \langle \gamma_j, \mu_j\rangle\rangle \xrightarrow[\alpha]{\mathsf{pop}(\gamma_j)} \\
&\langle q_2, \nu, \dots \langle \gamma_h, \mu_h\rangle \dots \langle \gamma_i, \mu_i\rangle \dots\rangle \underset{\star}{\Rightarrow}^* \\
&\langle q_3, \nu', \dots, \langle \gamma_h, \mu'_h\rangle \dots \langle \gamma_i, \mu'_i\rangle\rangle \underset{\star}{\Rightarrow}^* \langle q_4, \nu'', \dots \langle \gamma_h, \mu''_h\rangle\rangle \xrightarrow[\alpha']{\mathsf{pop}(\gamma_h)} \langle q_5, \nu'', \dots\rangle.
\end{aligned}$$

If $\mu_j(z) - \nu(x) \in J$ and $\mu''_h(z) - \nu''(x) \in J$, then $\mu'_i(z) - \nu'(x) \in J$.

Proof. By Lemma 4.9, we have $\mu_j(z) - \nu(x) \le \mu'_i(z) - \nu'(x) \le \mu''_h(z) - \nu''(x)$. The presence of $\xrightarrow[\alpha]{\mathsf{pop}(\gamma_j)}$ and $\xrightarrow[\alpha']{\mathsf{pop}(\gamma_h)}$ implies $\mu_j(z) - \nu(x) \in J$ and $\mu''_h(z) - \nu''(x) \in J$, respectively. Now, Proposition 4.1 implies $\mu'_i(z) - \nu'(x) \in J$. □

4.4.3 Example: Removing $z - x \in_? I$ based on Predicting MTPDA

Lemma 4.8 and 4.10 imply an important idea to remove $z - x \in_? I$. By Lemma 4.8, any stack w appearing in a computation of the MTPDA A, which does not contain $\mathsf{reset}(z)$, takes the following form:

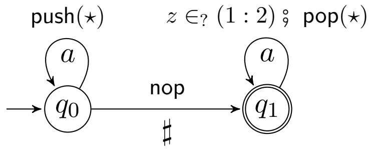
$$\begin{aligned}
w = {} & \underline{\langle(\gamma_1, I_\uparrow), \mu_1\rangle} \dots \langle(\gamma, I_\uparrow), \mu\rangle \dots \underline{\langle(\gamma_\ell, I_\uparrow), \mu_\ell\rangle} \\
& \underline{\langle(\gamma_{\ell+1}, I), \mu_{\ell+1}\rangle} \dots \underline{\langle(\gamma_n, I), \mu_n\rangle} \\
& \underline{\langle(\gamma_{n+1}, I_\downarrow), \mu_{n+1}\rangle} \dots \underline{\langle(\gamma_m, I_\downarrow), \mu_m\rangle}.
\end{aligned}$$

By Lemma 4.10, we only need the exact values of the local clock z of the above underlined six frames. Indeed, for example, we do not need the exact value of z in the frame $\langle(\gamma, I_\uparrow), \mu\rangle$ because the frame automatically satisfies $z - x \in_? I$ if we can safely pop the frames $\langle(\gamma_1, I_\uparrow), \mu_1\rangle$ and $\langle(\gamma_\ell, I_\uparrow), \mu_\ell\rangle$ by Lemma 4.10. On the basis of this intuition, we prepare extra six global clocks that reflect the values of the above six underlined frames and replace $z - x \in_? I$ with a diagonal constraint between global clocks.

Let us consider the following timed language:

$$L = \left\{ \langle a, t_1\rangle \dots \langle a, t_n\rangle \langle \sharp, t\rangle \langle \overline{a}, t'_n\rangle \dots \langle \overline{a}, t'_1\rangle : t'_i - t_i \in (1:2) \right\}.$$

The following 1-TPDA A recognizes this language by the empty stack ($L_\epsilon(A) = L$):



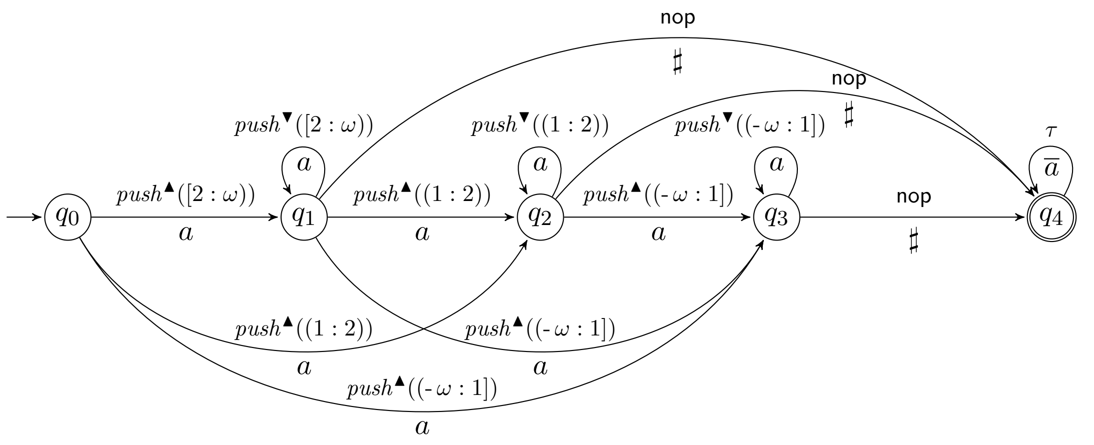
For the sake of simplicity, we do not remove the action $z \in_? (1:2)$ by Lemma 4.2 and directly treat it. By Lemma 4.8, a stack of the predicting MTPDA obtained from the 1-TPDA forms the following:

$$\begin{aligned}
&\big\langle(\gamma_1, [2:\omega)), \mu_1\big\rangle \big\langle(\gamma_2, [2:\omega)), \mu_2\big\rangle \dots \big\langle(\gamma_{\ell-1}, [2:\omega)), \mu_{\ell-1}\big\rangle \big\langle(\gamma_\ell, [2:\omega)), \mu_\ell\big\rangle \\
&\big\langle(\gamma_{\ell+1}, (1:2)), \mu_{\ell+1}\big\rangle \big\langle(\gamma_{\ell+2}, (1:2)), \mu_{\ell+2}\big\rangle \dots \big\langle(\gamma_{m-1}, (1:2)), \mu_{m-1}\big\rangle \big\langle(\gamma_m, (1:2)), \mu_m\big\rangle \\
&\big\langle(\gamma_{m+1}, (\text{-}\,\omega:1]), \mu_{m+1}\big\rangle \big\langle(\gamma_{m+2}, (\text{-}\,\omega:1]), \mu_{m+2}\big\rangle \dots \big\langle(\gamma_{n-1}, (\text{-}\,\omega:1]), \mu_{n-1}\big\rangle \big\langle(\gamma_n, (\text{-}\,\omega:1]), \mu_n\big\rangle.
\end{aligned}$$

Furthermore, by Lemma 4.10, we need six clocks $x^\blacktriangle_{[2:\omega)}$, $x^\blacktriangledown_{[2:\omega)}$, $x^\blacktriangle_{(1:2)}$, $x^\blacktriangledown_{(1:2)}$, $x^\blacktriangle_{(\text{-}\,\omega:1]}$, and $x^\blacktriangledown_{(\text{-}\,\omega:1]}$ where $x^\blacktriangle_{[2:\omega)}$ is the clock for the frame $\big\langle\big(\gamma_1,[2:\omega)\big),\mu_1\big\rangle$ and $x^\blacktriangledown_{[2:\omega)}$ is the clock for the frame $\big\langle\big(\gamma_\ell,[2:\omega)\big),\mu_\ell\big\rangle$ and so on. To distinguish the frame $\langle(\gamma_1,[2:\omega)),\mu_1\rangle$ associated with the clock $x^\blacktriangle_{[2:\omega)}$ from the other frames, we add a mark $\cdot^\blacktriangle$ as $\langle(\gamma_1,[2:\omega)^\blacktriangle),\mu_1\rangle$. Following this idea, the above stack is modified as follows:

$$\big\langle\big(\gamma_1,[2:\omega)^\blacktriangle\big),\mu_1\big\rangle\big\langle\big(\gamma_2,[2:\omega)\big),\mu_2\big\rangle\dots\big\langle\big(\gamma_{\ell-1},[2:\omega)\big),\mu_{\ell-1}\big\rangle\big\langle\big(\gamma_\ell,[2:\omega)^\blacktriangledown\big),\mu_\ell\big\rangle$$
$$\big\langle\big(\gamma_{\ell+1},(1:2)^\blacktriangle\big),\mu_{\ell+1}\big\rangle\big\langle\big(\gamma_{\ell+2},(1:2)\big),\mu_{\ell+2}\big\rangle\dots\big\langle\big(\gamma_{m-1},(1:2)\big),\mu_{m-1}\big\rangle\big\langle\big(\gamma_m,(1:2)^\blacktriangledown\big),\mu_m\big\rangle$$
$$\big\langle\big(\gamma_{m+1},(\text{-}\,\omega:1]^\blacktriangle\big),\mu_{m+1}\big\rangle\big\langle\big(\gamma_{m+2},(\text{-}\,\omega:1]\big),\mu_{m+2}\big\rangle\dots\big\langle\big(\gamma_{n-1},(\text{-}\,\omega:1]\big),\mu_{n-1}\big\rangle\big\langle\big(\gamma_n,(\text{-}\,\omega:1]^\blacktriangledown\big),\mu_n\big\rangle\,.$$

Now we define a predicting MTPDA that recognizes L by the empty stack:



q_0
q_1
q_2
q_3
q_4
$push^\blacktriangle([2:\omega))$
a
$push^\blacktriangle((1:2))$
a
$push^\blacktriangle((\text{-}\,\omega:1])$
a
nop
$\sharp$
$push^\blacktriangledown([2:\omega))$
a
$push^\blacktriangledown((1:2))$
a
$push^\blacktriangledown((\text{-}\,\omega:1])$
a
τ
$\overline{a}$
nop
$\sharp$
nop
$\sharp$
$push^\blacktriangle((1:2))$
a
$push^\blacktriangle((\text{-}\,\omega:1])$
a
$push^\blacktriangle((\text{-}\,\omega:1])$
a

where

$$\begin{array}{rcl}
push^\blacktriangle(J) & \triangleq & \mathsf{push}(J)\,;\,\mathsf{reset}(x^\blacktriangle_J),\\
push^\blacktriangledown(J) & \triangleq & \mathsf{check}(J^\blacktriangle)\,;\,\mathsf{push}(J^\blacktriangledown)\,;\,\mathsf{reset}(x^\blacktriangledown_J)\ +\ \mathsf{check}(J^\blacktriangledown)\,;\,\mathsf{rew}(J)\,;\,\mathsf{push}(J^\blacktriangledown)\,;\,\mathsf{reset}(x^\blacktriangledown_J),\\
pop^\blacktriangle(J) & \triangleq & \mathsf{check}(J^\blacktriangle)\,;\,x^\blacktriangle_J \in_? J\,;\,\mathsf{pop}(J^\blacktriangle),\\
pop^\blacktriangledown(J) & \triangleq & \mathsf{check}(J^\blacktriangledown)\,;\,x^\blacktriangledown_J \in_? J\,;\,\mathsf{pop}(J^\blacktriangledown),\\
\tau & = & pop^\blacktriangle([2:\omega)) + pop^\blacktriangle((1:2)) + pop^\blacktriangle((\text{-}\,\omega:1])\\
 & + & pop^\blacktriangledown([2:\omega)) + pop^\blacktriangledown((1:2)) + pop^\blacktriangledown((\text{-}\,\omega:1])\\
 & + & \mathsf{pop}([2:\omega)) + \mathsf{pop}((1:2)) + \mathsf{pop}((\text{-}\,\omega:1]).
\end{array}$$

It should be noted that there are no transition rules with $z \in_? (1:2)$; therefore, we do not inspect any local clocks.

Let us consider an acceptable computation of the predicting MTPDA (in the following, we only write clocks that are assigned to some frame):

$$\langle q_0,\emptyset,\epsilon\rangle \overset{0.0}{\rightsquigarrow}\xrightarrow[a]{push^\blacktriangle([2:\omega))}$$
$$\langle q_1,\Big\{x^\blacktriangle_{[2:\omega)}\mapsto 0.0\Big\},\langle[2:\omega)^\blacktriangle,0.0\rangle\rangle \overset{0.3}{\rightsquigarrow}\xrightarrow[a]{push^\blacktriangle((1:2))}$$
$$\langle q_2,\Big\{x^\blacktriangle_{[2:\omega)}\mapsto 0.3, x^\blacktriangle_{(1:2)}\mapsto 0.0\Big\},\langle[2:\omega)^\blacktriangle,0.3\rangle\langle(1:2)^\blacktriangle,0.0\rangle\rangle \overset{0.1}{\rightsquigarrow}\xrightarrow[a]{push^\blacktriangledown((1:2))}$$
$$\langle q_2,\Big\{x^\blacktriangle_{[2:\omega)}\mapsto 0.4, x^\blacktriangle_{(1:2)}\mapsto 0.1, x^\blacktriangledown_{(1:2)}\mapsto 0.0\Big\},\langle[2:\omega)^\blacktriangle,0.4\rangle\langle(1:2)^\blacktriangle,0.1\rangle\langle(1:2)^\blacktriangledown,0.0\rangle\rangle \overset{0.2}{\rightsquigarrow}\xrightarrow[a]{push^\blacktriangledown((1:2))}$$
$$\langle q_2,\Big\{x^\blacktriangle_{[2:\omega)}\mapsto 0.6, x^\blacktriangle_{(1:2)}\mapsto 0.3, x^\blacktriangledown_{(1:2)}\mapsto 0.0\Big\},\langle[2:\omega)^\blacktriangle,0.6\rangle\langle(1:2)^\blacktriangle,0.3\rangle\langle(1:2),0.2\rangle\langle(1:2)^\blacktriangledown,0.0\rangle\rangle \overset{0.0}{\rightsquigarrow}\xrightarrow[\#]{\mathsf{nop}}$$
$$\langle q_4,\Big\{x^\blacktriangle_{[2:\omega)}\mapsto 0.6, x^\blacktriangle_{(1:2)}\mapsto 0.3, x^\blacktriangledown_{(1:2)}\mapsto 0.0\Big\},\langle[2:\omega)^\blacktriangle,0.6\rangle\langle(1:2)^\blacktriangle,0.3\rangle\langle(1:2),0.2\rangle\langle(1:2)^\blacktriangledown,0.0\rangle\rangle \overset{1.1}{\rightsquigarrow}\xrightarrow[\overline{a}]{\tau}$$
$$\langle q_4,\Big\{x^\blacktriangle_{[2:\omega)}\mapsto 1.7, x^\blacktriangle_{(1:2)}\mapsto 1.4\Big\},\langle[2:\omega)^\blacktriangle,1.7\rangle\langle(1:2)^\blacktriangle,1.4\rangle\langle(1:2),1.3\rangle\rangle \overset{0.1}{\rightsquigarrow}\xrightarrow[\overline{a}]{\tau}$$
$$\langle q_4,\Big\{x^\blacktriangle_{[2:\omega)}\mapsto 1.8, x^\blacktriangle_{(1:2)}\mapsto 1.5\Big\},\langle[2:\omega)^\blacktriangle,1.8\rangle\langle(1:2)^\blacktriangle,1.5\rangle\rangle \overset{0.1}{\rightsquigarrow}\xrightarrow[\overline{a}]{\tau}$$
$$\langle q_4,\Big\{x^\blacktriangle_{[2:\omega)}\mapsto 1.9\Big\},\langle[2:\omega)^\blacktriangle,1.9\rangle\rangle \overset{0.05}{\rightsquigarrow}\xrightarrow[\overline{a}]{\tau}\langle q_4,\emptyset,\epsilon\rangle.$$

It is important that we only require three global clocks $x^{\blacktriangle}_{[2:\omega)}$, $x^{\blacktriangle}_{(1:2)}$, and $x^{\blacktriangledown}_{(1:2)}$ and do not need the value of the local clock in the frame $\langle (1:2), \_ \rangle$ when we pop the frame.

Next, let us consider an unacceptable computation:

$$\begin{aligned}
&\langle q_0, \emptyset, \epsilon \rangle \overset{0.0}{\rightsquigarrow} \xrightarrow[a]{push^{\blacktriangle}((1:2))} \\
&\langle q_2, \left\{ x^{\blacktriangle}_{(1:2)} \mapsto 0.0 \right\}, \langle (1:2)^{\blacktriangle}, 0.0 \rangle \rangle \overset{1.0}{\rightsquigarrow} \xrightarrow[a]{push^{\blacktriangle}((1:2))} \\
&\langle q_2, \left\{ x^{\blacktriangle}_{(1:2)} \mapsto 1.0, x^{\blacktriangledown}_{(1:2)} \mapsto 0.0 \right\}, \langle (1:2)^{\blacktriangle}, 1.0 \rangle \langle (1:2)^{\blacktriangledown}, 0.0 \rangle \rangle \overset{0.1}{\rightsquigarrow} \xrightarrow[a]{push^{\blacktriangle}((1:2))} \\
&\langle q_2, \left\{ x^{\blacktriangle}_{(1:2)} \mapsto 1.1, x^{\blacktriangledown}_{(1:2)} \mapsto 0.0 \right\}, \langle (1:2)^{\blacktriangle}, 1.1 \rangle \langle (1:2), 0.1 \rangle \langle (1:2)^{\blacktriangledown}, 0.0 \rangle \rangle \overset{1.6}{\rightsquigarrow} \xrightarrow[\sharp]{\mathsf{nop}} \\
&\langle q_4, \left\{ x^{\blacktriangle}_{(1:2)} \mapsto 2.7, x^{\blacktriangledown}_{(1:2)} \mapsto 1.6 \right\}, \langle (1:2)^{\blacktriangle}, 2.7 \rangle \langle (1:2), 1.7 \rangle \langle (1:2)^{\blacktriangledown}, 1.6 \rangle \rangle \overset{0.1}{\rightsquigarrow} \xrightarrow[\overline{a}]{\tau} \\
&\langle q_4, \left\{ x^{\blacktriangle}_{(1:2)} \mapsto 2.8 \right\}, \langle (1:2)^{\blacktriangle}, 2.8 \rangle \langle (1:2), 1.8 \rangle \rangle \overset{1.1}{\rightsquigarrow} \\
&\langle q_4, \left\{ x^{\blacktriangle}_{(1:2)} \mapsto 3.9 \right\}, \langle (1:2)^{\blacktriangle}, 3.9 \rangle \langle (1:2), 2.9 \rangle \rangle \xrightarrow[\overline{a}]{\tau} \\
&\langle q_4, \left\{ x^{\blacktriangle}_{(1:2)} \mapsto 3.9 \right\}, \langle (1:2)^{\blacktriangle}, 3.9 \rangle \rangle \overset{1.5}{\rightsquigarrow} \xrightarrow[\overline{a}]{\tau} \textbf{STUCK}.
\end{aligned}$$

The following last part is important:

$$\begin{aligned}
&\langle q_2, \left\{ x^{\blacktriangle}_{(1:2)} \mapsto 3.9 \right\}, \langle (1:2)^{\blacktriangle}, 3.9 \rangle \langle (1:2), 2.9 \rangle \rangle \xrightarrow[\overline{a}]{\tau} \\
&\langle q_2, \left\{ x^{\blacktriangle}_{(1:2)} \mapsto 3.9 \right\}, \langle (1:2)^{\blacktriangle}, 3.9 \rangle \rangle \overset{1.5}{\rightsquigarrow} \xrightarrow[\overline{a}]{\tau} \textbf{STUCK}.
\end{aligned}$$

Since we do not record the value of the local clock of the frame $\langle (1:2), \_ \rangle$, we cannot check $z \in_? (1:2)$ and pop the frame wrongly. However, it is not critical; because we can safely reject this computation when we try to pop the frame $\langle (1:2)^{\blacktriangle}, \_ \rangle$.

On the basis of this construction, in the next section, we will remove a transition rule with $z - x \in I$ from an MTPDA without $\mathsf{reset}(z)$. Furthermore, we generalize the construction for MTPDA that allows $\mathsf{reset}(z)$.

4.5 Removing Transition Rules with $z - x \in_? I$ from MTPDA ***without*** $\mathsf{reset}(z)$

Let A be an MTPDA and $z - x \in_? I$ be an action of A. Furthermore, we assume that A does not have a transition rule with $\mathsf{reset}(z)$. In the present section, by the technique of predicting MTPDA, we will construct an MTPDA C such that $L(A) = L(C)$ and C does not have transition rules with $z - x \in_? I$.

We denote the MTPDA A as $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta)$ and construct the following predicting MTPDA B obtained from the MTPDA A and interval I:

$$B = (Q, q_{\text{init}}, F, \Sigma, \Gamma \times \{I_{\downarrow}, I, I_{\uparrow}\}, \mathcal{X}, \mathcal{Z}, \Delta_B).$$

4.5.1 Predicting MTPDA C not having $z - x \in_? I$

We construct the following MTPDA C that has no transitions with $z - x \in I$:

$$C = (Q, q_{\text{init}}, F, \Sigma, \Gamma_C, \mathcal{X}_C, \mathcal{Z}, \Delta_C)$$

where

- $\Gamma_C \triangleq \Gamma \times \{J^{\blacktriangle}, J, J^{\blacktriangledown} : J \in \{I_{\downarrow}, I, I_{\uparrow}\}\}$,
- $\mathcal{X}_C \triangleq \mathcal{X} \cup \{x^{\blacktriangle}_J, x^{\blacktriangledown}_J : J \in \{I_{\downarrow}, I, I_{\uparrow}\}\}$.

On the basis of Lemma 4.8 and the explanation of Section 4.4.3, we consider stacks of the following form:

$$\begin{array}{l}\langle(\gamma_1, I_\uparrow^\blacktriangle), \mu_1\rangle\, \langle(\gamma_2, I_\uparrow), \mu_2\rangle \,\dots\, \langle(\gamma_{\ell-1}, I_\uparrow), \mu_{\ell-1}\rangle\, \langle(\gamma_\ell, I_\uparrow^\blacktriangledown), \mu_\ell\rangle \\ \langle(\gamma_{\ell+1}, I^\blacktriangle), \mu_{\ell+1}\rangle\, \langle(\gamma_{\ell+2}, I), \mu_{\ell+2}\rangle \,\dots\, \langle(\gamma_{n-1}, I), \mu_{n-1}\rangle\, \langle(\gamma_n, I^\blacktriangledown), \mu_n\rangle \\ \langle(\gamma_{n+1}, I_\downarrow^\blacktriangle), \mu_{n+1}\rangle\, \langle(\gamma_{n+2}, I_\downarrow), \mu_{n+2}\rangle \,\dots\, \langle(\gamma_{m-1}, I_\downarrow), \mu_{m-1}\rangle\, \langle(\gamma_m, I_\downarrow^\blacktriangledown), \mu_m\rangle\end{array}$$

We prepare six clocks $x_J^\blacktriangle, x_J^\blacktriangledown$ for $J \in \{I_\downarrow, I, I_\uparrow\}$ to remember the value of the local clock z in the frame with $J^\blacktriangle$ or $J^\blacktriangledown$, respectively. Following these ideas, we now define the set of transition rules Δ_C as follows.

Case $p \xrightarrow[\alpha]{\mathsf{push}(\gamma,J)} q \in \Delta_B$**:** We add transition rules to Δ_C as follows:

- The following rule corresponds to pushing a frame onto the empty stack:

$$p \xrightarrow[\alpha]{\mathsf{check}(\epsilon) ⨟ \mathsf{push}(\gamma, J^\blacktriangle) ⨟ \mathsf{reset}(x_J^\blacktriangle)} q.$$

- The following rules correspond to pushing a frame onto the stack whose top symbol has J:

$$p \xrightarrow[\alpha]{\mathsf{check}(\gamma', J^\blacktriangle) ⨟ \tau} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma', J) ⨟ \tau} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma', J^\blacktriangledown) ⨟ \mathsf{rew}(J) ⨟ \tau} q$$

  where $\tau = \mathsf{push}(\gamma, J^\blacktriangledown) ⨟ \mathsf{reset}(x_J^\blacktriangledown)$. The first rule induces a transition of the form $wJ^\blacktriangle \to wJ^\blacktriangle J^\blacktriangledown$, the second rule induces a transition of the form $wJ^\blacktriangle J \to wJ^\blacktriangle JJ^\blacktriangledown$, and the third rule induces a transition of the form $wJ^\blacktriangle JJ^\blacktriangledown \to wJ^\blacktriangle JJJ^\blacktriangledown$.

- The following rules correspond to pushing a frame onto the stack whose top symbol has K and $J \sqsubset K$:

$$p \xrightarrow[\alpha]{\mathsf{check}(\gamma', K^\blacktriangle) ⨟ \tau} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma', K) ⨟ \tau} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma', K^\blacktriangledown) ⨟ \tau} q$$

  where $\tau = \mathsf{push}(\gamma, J^\blacktriangle) ⨟ \mathsf{reset}(x_J^\blacktriangle)$.

Case $p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J) ⨟ z - x \in_? J ⨟ \mathsf{pop}(\gamma,J)} q \in \Delta_B$**:** We add transition rules to Δ_C as follows:

$$p \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^\blacktriangle) ⨟ x_J^\blacktriangle - x \in_? J ⨟ \mathsf{pop}(\gamma, J^\blacktriangle)} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^\blacktriangledown) ⨟ x_J^\blacktriangledown - x \in_? J ⨟ \mathsf{pop}(\gamma, J^\blacktriangledown)} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma, J) ⨟ \mathsf{pop}(\gamma, J)} q.$$

Compared to the original rule, the last rule does not check the value of the local clock z. This, however, does not cause any trouble by Lemma 4.10.

Case $p \xrightarrow[\alpha]{z - x \in_? I} q \in \Delta_B$**:** For the action to be removed, if some global clock is assigned to the stack top frame, then we use it to check $z - x \in_? I$. Otherwise, if the stack top frame has the prediction I, then we permit a transition. Therefore, we add the following transition rules:

$$p \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^\blacktriangle) ⨟ x_J^\blacktriangle - x \in_? I} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^\blacktriangledown) ⨟ x_J^\blacktriangledown - x \in_? I} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma, I)} q.$$

Other Rules $p \xrightarrow[\alpha]{\tau} q \in \Delta_B$**:** For the rules other than those above, we add the same rules.

4.5.2 Properties of Predicting MTPDA C

In order to show the predicting MTPDA C can simulate the predicting MTPDA B, we prove some important properties of C. The following lemma states that a global clock associated with a marked frame correctly records the value of the local clock z of the marked frame.

Lemma 4.11. Let us consider the following computations:

$$\langle q_{\text{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow^* \langle q_1, \nu_1, w_1 \langle (\gamma_1, J^\blacktriangle), \mu_1 \rangle \rangle,$$
$$\langle q_{\text{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow^* \langle q_2, \nu_2, w_2 \langle (\gamma_2, J^\blacktriangledown), \mu_2 \rangle \rangle.$$

For these computations, we have $\mu_1(z) = \nu_1(x_J^\blacktriangle)$ and $\mu_2(z) = \nu_2(x_J^\blacktriangledown)$.

Proof. To show this, we introduce a predicate for a configuration. A configuration $\langle q, \nu, w \rangle$ is *correctly storing local clocks*, $\mathit{CS}(\langle q, \nu, w \rangle)$, if the following conditions are satisfied:

- if there is a stack frame of the form $\langle (\gamma, J^\blacktriangle), \mu \rangle$ in the stack w, then $\nu(x_J^\blacktriangle) = \mu(z)$;
- if there is a stack frame of the form $\langle (\gamma, J^\blacktriangledown), \mu \rangle$ in the stack w, then $\nu(x_J^\blacktriangledown) = \mu(z)$.

We can easily verify that the predicate CS is an invariant for the transitions $\to$ and $\overset{\delta}{\rightsquigarrow}$. Since the initial configuration $\langle q_{\text{init}}, \mathbf{0}, \epsilon \rangle$ satisfies the predicate, the statement is derived because of $\mathit{CS}(\langle q_1, \nu_1, w_1 \langle (\gamma_1, J^\blacktriangle), \mu_1 \rangle \rangle)$ and $\mathit{CS}(\langle q_2, \nu_2, w_2 \langle (\gamma_2, J^\blacktriangledown), \mu_2 \rangle \rangle)$. □

The following lemma states that the value of the local clock z of a non-marked frame $\langle (\gamma, J), \mu \rangle$ belongs to J while the frame is located top on a stack.

Lemma 4.12. Let us consider the following computation:

$$\langle q_{\text{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow^* \langle q, \nu, w \langle (\gamma, J), \mu \rangle \rangle \Rightarrow^* \langle q_F, \nu_F, \epsilon \rangle.$$

For this computation, we have $\mu(z) - \nu(x) \in J$.

To prove this lemma, we need some additional lemmas. First, we show a lemma similar to Lemma 4.8. A stack w of the predicting MTPDA C is *well-formed* if it satisfies the following:

$$w \in \mathcal{F}(I_\uparrow) \cdot \mathcal{F}(I) \cdot \mathcal{F}(I_\downarrow)$$

where $\mathcal{F}$ is defined as follows:

$$\mathcal{F}(J) \triangleq \{\epsilon\} \quad \cup \quad \Upsilon(J^\blacktriangle) \cdot \Upsilon(J)^* \quad \cup \quad \Upsilon(J^\blacktriangle) \cdot \Upsilon(J)^* \cdot \Upsilon(J^\blacktriangledown).$$

Recall that $\Upsilon(J)$ is defined as $\Upsilon(J) \triangleq (\Gamma \times \{J\}) \times (\mathcal{Z} \to \mathbb{R}_{\geq 0})$.

Proposition 4.6. Let us consider a transition $\langle q, \nu, w \rangle \Rightarrow \langle q', \nu', w' \rangle$. If w is well-formed, then w' is also well-formed.

Proof. We proceed by case analysis on the transition. Here we consider transitions with `push` and `pop` because all the other transition rules do not change the symbols of a stack and the statement clearly holds.

First, we consider the case of a transition with `push`; there are the following types of transitions (for the sake of simplicity, we only write stack symbols):

$$\begin{array}{rcll}
\epsilon & \to & (\gamma, J^\blacktriangle), & \\
w(\gamma, J^\blacktriangle) & \to & w(\gamma, J^\blacktriangle)(\gamma', J^\blacktriangledown), & \\
w(\gamma, J) & \to & w(\gamma, J)(\gamma', J^\blacktriangledown), & \\
w(\gamma, J^\blacktriangledown) & \to & w(\gamma, J)(\gamma', J^\blacktriangledown), & \\
w(\gamma, K^\blacktriangle) & \to & w(\gamma, K^\blacktriangle)(\gamma', J^\blacktriangle), & (J \sqsubset K) \\
w(\gamma, K) & \to & w(\gamma, K)(\gamma', J^\blacktriangle), & (J \sqsubset K) \\
w(\gamma, K^\blacktriangledown) & \to & w(\gamma, K^\blacktriangledown)(\gamma', J^\blacktriangle). & (J \sqsubset K)
\end{array}$$

It is clear that the well-formedness is preserved for each case.

Next, we consider the case of a transition with pop; there are the following types of transitions:

$$w(\gamma, J^{\blacktriangle}) \to w, \quad w(\gamma, J^{\blacktriangledown}) \to w, \quad w(\gamma, J) \to w.$$

Again, it is clear that the well-formedness is preserved for each case. □

This proposition immediately derives that any reachable configuration, which can be reached from the initial configuration, is well-formed.

Lemma 4.13. Let us consider the following reachable configuration:

$$\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \langle q, \nu, w\rangle.$$

The stack w is well-formed.

Proof. From the definition, the empty stack ϵ is well-formed; therefore, by repeatedly applying Proposition 4.6, we know w is well-formed. □

Furthermore, this lemma leads to the following technical property.

Lemma 4.14. Let us consider the following computation:

$$\pi = \langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \langle q', \nu', w\langle(\gamma, J), \mu'\rangle\rangle.$$

We can decompose the above computation π as follows:

$$\begin{aligned}
&\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \\
&\langle p, \nu, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots \langle(\gamma_j, J^{\blacktriangledown}), \mu_j\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma_j, J^{\blacktriangledown})\,;\, x^{\blacktriangledown}_J - x \in ?J\,;\, \mathsf{pop}(\gamma_j, J^{\blacktriangledown})} \\
&\langle q, \nu, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots\rangle \Rightarrow^* \\
&\langle q', \nu', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu'_i\rangle \ldots \langle(\gamma, J), \mu'\rangle\rangle = \langle q', \nu', w\langle(\gamma, J), \mu'\rangle\rangle.
\end{aligned}$$

Proof. For the sake of simplicity, we omit valuations on the local clocks of a stack in this proof. By Lemma 4.13, the computation π must take the following form:

$$\pi = \langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \langle q', \nu', \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J)\rangle.$$

The following transition rule can only make the frame $\langle(\gamma, J), \mu'\rangle$ by our construction of the predicting MTPDA C:

$$p \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^{\blacktriangledown})\,;\, \mathsf{rew}(\gamma, J)\,;\, \mathsf{push}(\gamma, J^{\blacktriangledown})\,;\, \mathsf{reset}(x^{\blacktriangledown}_J)} q.$$

Therefore, we can again refine the computation π as follows:

$$\pi = \begin{aligned}
&\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \\
&\langle q'', \nu'', \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J^{\blacktriangledown})\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^{\blacktriangledown})\,;\, \mathsf{rew}(\gamma, J)\,;\, \mathsf{push}(\gamma, J^{\blacktriangledown})\,;\, \mathsf{reset}(x^{\blacktriangledown}_J)} \\
&\langle q''', \nu''', \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J)(\gamma', J^{\blacktriangledown})\rangle \underset{\sharp}{\Rightarrow}^* \\
&\langle q', \nu', \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J)\rangle.
\end{aligned}$$

Let us focus on the sequence of transitions $\underset{\sharp}{\Rightarrow}^*$. To reach $\langle q', \nu', \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J)\rangle$ from $\langle q''', \nu''', \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J)(\gamma', J^{\blacktriangledown})\rangle$, we must eventually pop a stack frame associated with $J^{\blacktriangledown}$. Therefore, we can refine $\underset{\sharp}{\Rightarrow}^*$ as follows:

$$\begin{aligned}
&\langle q''', \nu''', \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J)(\gamma', J^{\blacktriangledown})\rangle \Rightarrow^* \\
&\langle p, \nu, \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J) \ldots (\gamma_j, J^{\blacktriangledown})\rangle \xrightarrow[\alpha']{\mathsf{check}(\gamma_j, J^{\blacktriangledown})\,;\, x^{\blacktriangledown}_J - x \in ?J\,;\, \mathsf{pop}(\gamma_j, J^{\blacktriangledown})} \\
&\langle p, \nu, \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J) \ldots\rangle \Rightarrow^* \langle q', \nu', \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J)\rangle.
\end{aligned}$$

Combining these transitions, we finally have the adequate decomposition:

$$\begin{aligned}
&\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \\
&\langle p, \nu, \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J) \ldots (\gamma_j, J^{\blacktriangledown})\rangle \xrightarrow[\alpha']{\mathsf{check}(\gamma_j, J^{\blacktriangledown}) \fatsemi\ x_J^{\blacktriangledown} - x \in_? J \fatsemi\ \mathsf{pop}(\gamma_j, J^{\blacktriangledown})} \\
&\langle p, \nu, \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J) \ldots\rangle \Rightarrow^* \langle q', \nu', \ldots (\gamma_i, J^{\blacktriangle}) \ldots (\gamma, J)\rangle.
\end{aligned}$$

□

Now we go back to and prove Lemma 4.12 based on Lemma 4.14.

Lemma 4.12. Let us consider the following computation:

$$\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \langle q, \nu, w \, \langle(\gamma, J), \mu\rangle\rangle \Rightarrow^* \langle q_F, \nu_F, \epsilon\rangle.$$

For this computation, we have $\mu(z) - \nu(x) \in J$.

Proof. Applying Lemma 4.14 to the computation, we have the following decomposition:

$$\begin{aligned}
&\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \\
&\langle p, \nu, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots \langle(\gamma_j, J^{\blacktriangledown}), \mu_j\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma_j, J^{\blacktriangledown}) \fatsemi\ x_J^{\blacktriangledown} - x \in_? J \fatsemi\ \mathsf{pop}(\gamma_j, J^{\blacktriangledown})} \\
&\langle p', \nu, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots\rangle \Rightarrow^* \\
&\langle q', \nu'', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i''\rangle\rangle \xrightarrow[\alpha']{\mathsf{check}(\gamma_i, J^{\blacktriangle}) \fatsemi\ x_J^{\blacktriangle} - x \in_? J \fatsemi\ \mathsf{pop}(\gamma_i, J^{\blacktriangle})} \\
&\langle q', \nu'', \ldots\rangle \Rightarrow^* \langle q_F, \nu_F, \epsilon\rangle.
\end{aligned}$$

From the decomposition, we know $\nu(x_J^{\blacktriangledown}) - \nu(x) \in J$ and $\nu''(x_J^{\blacktriangle}) - \nu''(x) \in J$. These and Lemma 4.11 lead to $\mu_j(z) - \nu(x) \in J$ and $\mu_i''(z) - \nu''(x) \in J$. Finally, from the same argument as Lemma 4.10, we have $\mu(z) - \nu(x) \in J$. □

4.5.3 Equivalence of the predicting MTPDA B and C

In order to show $L(B) = L(C)$, we define the following relation between configurations of B and C:

$$\langle q_B, \nu, w_B\rangle \approx \langle q_C, \eta, w_C\rangle \overset{\text{def}}{\iff} q_B = q_C \ \wedge\ \nu = \eta \restriction \mathcal{X} \ \wedge\ w_B = \psi(w_C)$$

where $\psi : \Gamma_C^* \to (\Gamma \times \{I_\downarrow, I, I_\uparrow\})^*$ is a projection defined as follows:

$$\psi((\gamma, J)) = (\gamma, J), \quad \psi((\gamma, J^{\blacktriangle})) = (\gamma, J), \quad \psi((\gamma, J^{\blacktriangledown})) = (\gamma, J).$$

To show $L_\epsilon(B) = L_\epsilon(C)$, we first show $L_\epsilon(C) \subseteq L_\epsilon(B)$ and then show $L_\epsilon(B) \subseteq L_\epsilon(C)$.

Lemma 4.15. $L_\epsilon(C) \subseteq L_\epsilon(B)$.

Proof. Let $\boldsymbol{d}$ be a configuration of C such that:

$$\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_C^* \boldsymbol{d} \Rightarrow_C^* \langle q_F, \eta_F, \epsilon\rangle.$$

By induction on the length of the sequence of transitions $\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_C^* \boldsymbol{d}$, we show the following:

$$\begin{array}{ccccccccccc}
\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle & \overset{\delta_1}{\rightsquigarrow} & \exists \boldsymbol{c}_1 & \xrightarrow[\alpha_1]{\tau_1} & \exists \boldsymbol{c}_1' & \overset{\delta_2}{\rightsquigarrow} \cdots \overset{\delta_n}{\rightsquigarrow} & \exists \boldsymbol{c}_n & \xrightarrow[\alpha_n]{\tau_n} & \exists \boldsymbol{c}_n' & & \\
\wr\wr & & \wr\wr & & \wr\wr & & \wr\wr & & \wr\wr & & \\
\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle & \overset{\delta_1}{\rightsquigarrow} & \boldsymbol{d}_1 & \xrightarrow[\alpha_1]{\tau_1'} & \boldsymbol{d}_1' & \overset{\delta_2}{\rightsquigarrow} \cdots \overset{\delta_n}{\rightsquigarrow} & \boldsymbol{d}_n & \xrightarrow[\alpha_n]{\tau_n'} & \boldsymbol{d}_n' & = \boldsymbol{d} \Rightarrow^* & \langle q_F, \eta_F, \epsilon\rangle.
\end{array}$$

Since this is clear about the base case $\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_C^0 \langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle$, we consider the induction step by case analysis.

Case timed transition $\overset{\delta}{\rightsquigarrow}$. We consider the following case:

$$\begin{array}{ccccc}
\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle & \Rightarrow^*_B & \langle p, \nu, w \rangle & & \\
\wr & & \wr & & \\
\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle & \Rightarrow^*_C & \langle q, \eta, w' \rangle & \overset{\delta}{\rightsquigarrow} & \langle q, \eta + \delta, w' + \delta \rangle
\end{array}$$

By the definition of $\approx$, it is clear that $\langle q, \nu + \delta, w + \delta \rangle \approx \langle q, \eta + \delta, w' + \delta \rangle$.

Hereafter, we consider several non-trivial cases of discrete transitions. For all the other cases, the above diagram is easily shown.

Case pop. First, we consider the following subcase:

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow^*_C \langle p, \eta, w'\langle (\gamma, J^{\blacktriangle}), \mu \rangle \rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^{\blacktriangle}) \fatsemi\, x^{\blacktriangle}_J - x \in_? J \fatsemi\, \mathsf{pop}(\gamma, J^{\blacktriangle})}_C \langle q, \eta, w' \rangle.$$

By the induction hypothesis, we have the following:

$$\begin{array}{ccc}
\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle & \Rightarrow^*_B & \langle p, \nu, w\langle (\gamma, J), \mu \rangle \rangle \\
\wr & & \wr \\
\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle & \Rightarrow^*_C & \langle p, \eta, w'\langle (\gamma, J^{\blacktriangle}), \mu \rangle \rangle
\end{array}$$

Since $\eta(x^{\blacktriangle}_J) - \nu(x) \in J$ and Lemma 4.11 leads to $\mu(z) = \eta(x^{\blacktriangle}_J)$, $\mu(z) - \nu(x) \in J$ holds. Therefore, the following holds:

$$\begin{array}{ccc}
\langle p, \nu, w\langle (\gamma, J), \mu \rangle \rangle & \xrightarrow[\alpha]{\mathsf{check}(\gamma, J) \fatsemi\, z - x \in_? J \fatsemi\, \mathsf{pop}(\gamma, J)}_B & \langle p, \nu, w \rangle \\
\wr & & \wr \\
\langle p, \eta, w'\langle (\gamma, J^{\blacktriangle}), \mu \rangle \rangle & \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^{\blacktriangle}) \fatsemi\, x^{\blacktriangle}_J - x \in_? J \fatsemi\, \mathsf{pop}(\gamma, J^{\blacktriangle})}_C & \langle q, \eta, w' \rangle.
\end{array}$$

Next, we consider the following subcase:

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow^*_C \langle p, \eta, w'\langle (\gamma, J^{\blacktriangledown}), \mu \rangle \rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^{\blacktriangledown}) \fatsemi\, x^{\blacktriangledown}_J - x \in_? J \fatsemi\, \mathsf{pop}(\gamma, J^{\blacktriangledown})}_C \langle q, \eta, w' \rangle.$$

This case is shown in the same way as above; therefore, we omit the proof.

Finally, we consider the following subcase:

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow^*_C \langle p, \eta, w'\langle (\gamma, J), \mu \rangle \rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, J) \fatsemi\, \mathsf{pop}(\gamma, J)}_C \langle q, \eta, w' \rangle.$$

By the induction hypothesis, we have the following:

$$\begin{array}{ccc}
\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle & \Rightarrow^*_B & \langle p, \nu, w\langle (\gamma, J), \mu \rangle \rangle \\
\wr & & \wr \\
\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle & \Rightarrow^*_C & \langle p, \eta, w'\langle (\gamma, J), \mu \rangle \rangle
\end{array}$$

Applying Lemma 4.12 to the sequence of transitions $\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow^*_C \langle p, \eta, w'\langle (\gamma, J), \mu \rangle \rangle \Rightarrow^*_C \langle q_F, \eta_F, \epsilon \rangle$, we have $\mu(z) - \eta(x) \in J$. Since $\eta(x) = \nu(x)$, $\mu(z) - \nu(x) \in J$ holds and thus we have the following:

$$\begin{array}{ccc}
\langle p, \nu, w\langle (\gamma, J), \mu \rangle \rangle & \xrightarrow[\alpha]{\mathsf{check}(\gamma, J) \fatsemi\, z - x \in_? J \fatsemi\, \mathsf{pop}(\gamma, J)}_B & \langle p, \nu, w \rangle \\
\wr & & \wr \\
\langle p, \eta, w'\langle (\gamma, J), \mu \rangle \rangle & \xrightarrow[\alpha]{\mathsf{check}(\gamma, J) \fatsemi\, \mathsf{pop}(\gamma, J)}_C & \langle q, \eta, w' \rangle.
\end{array}$$

Case for transition rules derived from $p \xrightarrow[\alpha]{z-x\in?I} q \in \Delta_B$: First, we consider the following subcase:

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_C^* \langle p, \eta, w'\langle(\gamma, J^{\blacktriangle}), \mu\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^{\blacktriangle})\fatsemi\ x_J^{\blacktriangle}-x\in?I}_C \langle q, \eta, w'\langle(\gamma, J^{\blacktriangle}), \mu\rangle\rangle.$$

The induction hypothesis leads to the following:

$$\begin{array}{ccc} \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle & \Rightarrow_B^* & \langle p, \nu, w\langle(\gamma, J), \mu\rangle\rangle \\ \wr\wr & & \wr\wr \\ \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle & \Rightarrow_C^* & \langle p, \eta, w'\langle(\gamma, J^{\blacktriangle}), \mu\rangle\rangle \end{array}$$

By $\eta(x_J^{\blacktriangle}) - \eta(x) \in I$ and Lemma 4.11, we have $\mu(z) - \nu(x) \in I$ and the following transition holds:

$$\begin{array}{ccc} \langle p, \nu, w\langle(\gamma, J), \mu\rangle\rangle & \xrightarrow[\alpha]{z-x\in?I}_B & \langle p, \nu, w\langle(\gamma, J), \mu\rangle\rangle \\ \wr\wr & & \wr\wr \\ \langle p, \eta, w'\langle(\gamma, J^{\blacktriangle}), \mu\rangle\rangle & \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^{\blacktriangle})\fatsemi\ x_J^{\blacktriangle}-x\in?I}_C & \langle q, \eta, w'\langle(\gamma, J^{\blacktriangle}), \mu\rangle\rangle. \end{array}$$

Next, we consider the following subcase:

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_C^* \langle p, \eta, w'\langle(\gamma, J^{\blacktriangledown}), \mu\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^{\blacktriangledown})\fatsemi\ x_J^{\blacktriangledown}-x\in?I}_C \langle q, \eta, w'\langle(\gamma, J^{\blacktriangledown}), \mu\rangle\rangle.$$

This case is shown in the same way as above; therefore, we omit the proof.

Finally, we consider the following subcase:

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_C^* \langle p, \eta, w'\langle(\gamma, I), \mu\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, I)}_C \langle q, \eta, w'\langle(\gamma, I), \mu\rangle\rangle.$$

The induction hypothesis leads to the following:

$$\begin{array}{ccc} \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle & \Rightarrow_B^* & \langle p, \nu, w\langle(\gamma, I), \mu\rangle\rangle \\ \wr\wr & & \wr\wr \\ \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle & \Rightarrow_C^* & \langle p, \eta, w'\langle(\gamma, I), \mu\rangle\rangle. \end{array}$$

Applying Lemma 4.12 to the sequence of transitions $\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_C^* \langle p, \eta, w'\langle(\gamma, I), \mu\rangle\rangle \Rightarrow_C^* \langle q_F, \eta_F, \epsilon\rangle$, we have $\mu(z) - \eta(x) \in I$. This also implies $\mu(z) - \nu(x) \in I$; therefore we have the following:

$$\begin{array}{ccc} \langle p, \nu, w\langle(\gamma, I), \mu\rangle\rangle & \xrightarrow[\alpha]{z-x\in?I}_B & \langle p, \nu, w\langle(\gamma, I), \mu\rangle\rangle \\ \wr\wr & & \wr\wr \\ \langle p, \eta, w'\langle(\gamma, I), \mu\rangle\rangle & \xrightarrow[\alpha]{\mathsf{check}(\gamma, I)}_C & \langle q, \eta, w'\langle(\gamma, I), \mu\rangle\rangle. \end{array}$$

□

Lemma 4.16. $L_\epsilon(B) \subseteq L_\epsilon(C)$.

Proof. Let $\boldsymbol{c}$ be a configuration of B such that:

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_B^* \boldsymbol{c} \Rightarrow_B^* \langle q_F, \nu_F, \epsilon\rangle.$$

By induction on the length of the sequence of transitions $\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_B^* \boldsymbol{c}$, we show the following:

$$\begin{array}{ccccccccccc} \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle & \overset{\delta_1}{\rightsquigarrow} & \boldsymbol{c}_1 & \xrightarrow[\alpha_1]{\tau_1} & \boldsymbol{c}_1' & \overset{\delta_2}{\rightsquigarrow} \cdots \overset{\delta_n}{\rightsquigarrow} & \boldsymbol{c}_n & \xrightarrow[\alpha_n]{\tau_n} & \boldsymbol{c}_n' = \boldsymbol{c} \Rightarrow_B^* \langle q_F, \nu_F, \epsilon\rangle \\ \wr\wr & & \wr\wr & & \wr\wr & & \wr\wr & & \wr\wr \\ \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle & \overset{\delta_1}{\rightsquigarrow} & {}^{\exists}\boldsymbol{d}_1 & \xrightarrow[\alpha_1]{\tau_1'} & {}^{\exists}\boldsymbol{d}_1' & \overset{\delta_2}{\rightsquigarrow} \cdots \overset{\delta_n}{\rightsquigarrow} & {}^{\exists}\boldsymbol{d}_n & \xrightarrow[\alpha_n]{\tau_n'} & {}^{\exists}\boldsymbol{d}_n' \end{array}$$

Since the base case $\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_B^0 \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle$ is clear, we consider the induction step by case analysis.

Case timed transition $\overset{\delta}{\rightsquigarrow}$. We consider the following case:

$$\begin{array}{ccccc} \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle & \Rightarrow_B^* & \langle p, \nu, w \rangle & \overset{\delta}{\rightsquigarrow} & \langle q, \nu + \delta, w + \delta \rangle \\ \wr\wr & & \wr\wr & & \\ \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle & \Rightarrow_C^* & \langle q, \eta, w' \rangle & & \end{array}$$

By the definition of $\approx$, it is clear that $\langle q, \nu + \delta, w + \delta \rangle \approx \langle q, \eta + \delta, w' + \delta \rangle$.

Case push. We consider the following case:

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow_B^* \langle p, \nu, w \langle (\gamma, J), \mu \rangle \rangle \xrightarrow[\alpha]{\mathsf{push}(\gamma', J')}_B \langle q, \nu, w \langle (\gamma, J), \mu \rangle \langle (\gamma', J'), \mathbf{0} \rangle \rangle.$$

By Lemma 4.8, we have $J' \sqsubset J$ or $J' = J$. First, we consider the case $J' \sqsubset J$.

Subcase $\langle p, \nu, w\langle (\gamma, J), \mu \rangle \rangle \approx \langle p, \eta, w' \langle (\gamma, J^{\blacktriangle}), \mu \rangle \rangle$: By the induction hypothesis and the construction of C, we have the following:

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow_C^* \langle p, \eta, w' \langle (\gamma, J^{\blacktriangle}), \mu \rangle \rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^{\blacktriangle}) ⨾ \ \mathsf{push}(\gamma', J'^{\blacktriangle}) ⨾ \ \mathsf{reset}(x_{J'}^{\blacktriangle})}_C \langle q, \eta', w' \langle (\gamma, J^{\blacktriangle}), \mu \rangle \langle (\gamma', J'^{\blacktriangle}), \mathbf{0} \rangle \rangle.$$

Now $\nu = \eta' \restriction \mathcal{X}$ holds from $\nu = \eta \restriction \mathcal{X}$ and $\eta \restriction \mathcal{X} = \eta' \restriction \mathcal{X}$, and it leads to the following:

$$\langle q, \nu, w \langle (\gamma, J), \mu \rangle \langle (\gamma', J'), \mathbf{0} \rangle \rangle \approx \langle q, \eta', w' \langle (\gamma, J^{\blacktriangle}), \mu \rangle \langle (\gamma', J'^{\blacktriangle}), \mathbf{0} \rangle \rangle.$$

Subcases $\begin{cases} \langle p, \nu, w \langle (\gamma, J), \mu \rangle \rangle \approx \langle p, \eta, w' \langle (\gamma, J), \mu \rangle \rangle, \\ \langle p, \nu, w \langle (\gamma, J), \mu \rangle \rangle \approx \langle p, \eta, w' \langle (\gamma, J^{\blacktriangledown}), \mu \rangle \rangle \end{cases}$ **:** Since we can show these cases in the same way as the above case, we omit the proof.

Next, we consider the case $J' = J$.

Subcase $\langle p, \nu, w\langle (\gamma, J), \mu \rangle \rangle \approx \langle p, \eta, w' \langle (\gamma, J^{\blacktriangle}), \mu \rangle \rangle$: By the induction hypothesis and the construction of C, we have the following:

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow_C^* \langle p, \eta, w' \langle (\gamma, J^{\blacktriangle}), \mu \rangle \rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^{\blacktriangle}) ⨾ \ \mathsf{push}(\gamma', J^{\blacktriangledown}) ⨾ \ \mathsf{reset}(x_J^{\blacktriangledown})}_C \langle q, \eta', w' \langle (\gamma, J^{\blacktriangle}), \mu \rangle \langle (\gamma', J^{\blacktriangledown}), \mathbf{0} \rangle \rangle.$$

Now $\nu = \eta' \restriction \mathcal{X}$ holds from $\nu = \eta \restriction \mathcal{X}$ and $\eta \restriction \mathcal{X} = \eta' \restriction \mathcal{X}$, and it leads to the following:

$$\langle q, \nu, w \langle (\gamma, J), \mu \rangle \langle (\gamma', J), \mathbf{0} \rangle \rangle \approx \langle q, \eta', w' \langle (\gamma, J^{\blacktriangle}), \mu \rangle \langle (\gamma', J^{\blacktriangledown}), \mathbf{0} \rangle \rangle.$$

Subcases $\begin{cases} \langle p, \nu, w \langle (\gamma, J), \mu \rangle \rangle \approx \langle p, \eta, w' \langle (\gamma, J), \mu \rangle \rangle \textbf{ or} \\ \langle p, \nu, w \langle (\gamma, J), \mu \rangle \rangle \approx \langle p, \eta, w' \langle (\gamma, J^{\blacktriangledown}), \mu \rangle \rangle \end{cases}$ **:** These cases can be shown in the same way as the above case; therefore, we omit the proof.

Case pop. We consider the following case:

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow_B^* \langle p, \nu, w \langle (\gamma, J), \mu \rangle \rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, J) ⨾ \ z - x \in_? J ⨾ \ \mathsf{pop}(\gamma, J)}_B \langle q, \nu, w \rangle.$$

This implies $\mu(z) - \nu(x) \in J$.

Subcase $\langle p, \nu, w\langle (\gamma, J), \mu \rangle \rangle \approx \langle p, \eta, w' \langle (\gamma, J^{\blacktriangle}), \mu \rangle \rangle$: For this subcase, we know $\eta(x_J^{\blacktriangle}) - \eta(x) \in J$ because $\nu(x) = \eta(x)$ and Lemma 4.11 implies $\mu(z) = \eta(x_J^{\blacktriangle})$. Therefore, we have the following transition

$$\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow_C^* \langle p, \eta, w' \langle (\gamma, J^{\blacktriangle}), \mu \rangle \rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^{\blacktriangle}) ⨾ \ x_J^{\blacktriangle} - x \in_? J ⨾ \ \mathsf{pop}(\gamma, J^{\blacktriangle})}_C \langle q, \eta, w' \rangle.$$

Subcase $\begin{cases} \langle p, \nu, w \langle (\gamma, J), \mu \rangle \rangle \approx \langle p, \eta, w' \langle (\gamma, J), \mu \rangle \rangle \textbf{ or} \\ \langle p, \nu, w \langle (\gamma, J), \mu \rangle \rangle \approx \langle p, \eta, w' \langle (\gamma, J^{\blacktriangledown}), \mu \rangle \rangle \end{cases}$ **:** These cases can be shown in the same way as the above case; therefore, we omit the proof.

Case $z - x \in_? I$. We consider the following case:

$$\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_B^* \langle p, \nu, w\langle(\gamma, J), \mu\rangle\rangle \xrightarrow[\alpha]{z-x\in_? I}_B \langle q, \nu, w\langle(\gamma, J), \mu\rangle\rangle.$$

This means $\mu(z) - \nu(x) \in I$.

Subcase $\langle p, \nu, w\langle(\gamma, J), \mu\rangle\rangle \approx \langle p, \eta, w'\langle(\gamma, J^\blacktriangle), \mu\rangle\rangle$: Since $\mu(z) = \eta(x_J^\blacktriangle)$ by Lemma 4.11 and $\nu(x) = \eta(x)$, the predicting MTPDA C has the following transition:

$$\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_C^* \langle p, \eta, w'\langle(\gamma, J^\blacktriangle), \mu\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, J^\blacktriangle)\,⨾\; x_J^\blacktriangle - x \in_? I}_C \langle q, \eta, w'\langle(\gamma, J^\blacktriangle), \mu\rangle\rangle.$$

Subcase $\langle p, \nu, w\langle(\gamma, J), \mu\rangle\rangle \approx \langle p, \eta, w'\langle(\gamma, J^\blacktriangledown), \mu\rangle\rangle$: This case is shown by the same argument as the above.

Subcase $\langle p, \nu, w\langle(\gamma, J), \mu\rangle\rangle \approx \langle p, \eta, w'\langle(\gamma, J), \mu\rangle\rangle$: By the induction hypothesis and Lemma 4.13, we have the following:

$$\begin{array}{lll}
\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_B^* & \langle q', \nu', \ldots \langle(\_, J), \_\rangle \ldots \langle(\gamma, J), \_\rangle \ldots \langle(\_, J), \_\rangle \ldots\rangle & \Rightarrow_B^* \langle p, \nu, w\langle(\gamma, J), \mu\rangle\rangle \\
\quad \wr\wr & \qquad\qquad\qquad \wr\wr & \\
\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_C^* & \langle q', \eta', \ldots \langle(\_, J^\blacktriangle), \_\rangle \ldots \langle(\gamma, J), \mu'\rangle \ldots \langle(\_, J^\blacktriangledown), \_\rangle \ldots\rangle & \Rightarrow_C^* \langle p, \eta, w'\langle(\gamma, J), \mu\rangle\rangle.
\end{array}$$

We have $\mu(z) - \nu(x) \in J$ by applying Lemma 4.10 to the following computation:

$$\langle q', \nu', \ldots \langle(\_, J), \_\rangle \ldots \langle(\gamma, J), \_\rangle \ldots \langle(\_, J), \_\rangle \ldots\rangle \Rightarrow_C^* \langle p, \nu, w\langle(\gamma, J), \mu\rangle\rangle \Rightarrow_C^* \langle q_F, \nu_F, \epsilon\rangle.$$

Since $\mu(z) - \nu(x) \in I$ and $J \in \{I_\uparrow, I, I_\downarrow\}$, $J = I$ must hold.

Finally, we have the following transition from the definition of the predicting MTPDA C:

$$\langle p, \eta, w'\langle(\gamma, I), \mu\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, I)}_C \langle q, \eta, w'\langle(\gamma, I), \mu\rangle\rangle.$$

□

Combining these lemmas, we have the following result.

Theorem 4.1. Let A be an MTPDA that does not have transitions with $\mathsf{reset}(z)$ and $z - x \in_? I$ be an action that compares the local clock z and global clock x. There is an MTPDA C that satisfies the following conditions:

- $L_\epsilon(A) = L_\epsilon(C)$.
- There are no transition rules that contain an action $z - x \in_? I$.

4.6 Removing Transition Rules with $z - x \in_? I$ from MTPDA

Extending the construction of the previous section, we will show the following theorem in the present section.

Theorem 4.2. Let A be an MTPDA and $z - x \in_? I$ be an action that compares the local clock z and global clock x. There is an MTPDA D that satisfies the following conditions:

- $L_\epsilon(A) = L_\epsilon(D)$.
- There are no transition rules that contain an action $z - x \in_? I$.

In the previous section, we impose the restriction that there are no transitions with the action $\mathsf{reset}(z)$ with respect to a fixed action $z - x \in_? I$. We needed this restriction to prove the important lemmas about a stack of predicting MTPDA, Lemma 4.9 and 4.10; indeed, if we permit transitions with the action $\mathsf{reset}(z)$, then Lemma 4.9 does not hold.

In the present section, we use new auxiliary global clocks $\{\mathtt{C}_{I_\downarrow}, \mathtt{C}_I, \mathtt{C}_{I_\uparrow}\}$ to deal with the action $\mathsf{reset}(z)$. Our strategy is simple: if the action $\mathsf{reset}(z)$ is taken when the stack top has a prediction J, then we assign the auxiliary global clock $\mathtt{C}_J$ (if we already assign $\mathtt{C}_J$ to some frame, then we dissolve the assignment and then we assign the clock to the top frame).

Let A be an MTPDA and B be the predicting MTPDA obtained by A:

$$\begin{aligned} A &= (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta), \\ B &= (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma \times \{I_\downarrow, I, I_\uparrow\}, \mathcal{X}, \mathcal{Z}, \Delta_B). \end{aligned}$$

We construct the following MTPDA D that does not have transition rules with $z - x \in_? I$:

$$D = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma_D, \mathcal{X}_D, \mathcal{Z}, \Delta_D)$$

where Γ_D and $\mathtt{C}_D$ is defined as follows:

$$\begin{aligned} \Gamma_D &\triangleq \Gamma \times \{J^\blacktriangle, J, J^\blacktriangledown, J^{\blacktriangle,\circ}, J^\circ, J^{\blacktriangledown,\circ} : J \in \{I_\downarrow, I, I_\uparrow\}\}, \\ \mathcal{X}_D &\triangleq \mathcal{X} \;\cup\; \{x_J^\blacktriangle, x_J^\blacktriangledown : J \in \{I_\downarrow, I, I_\uparrow\}\} \;\cup\; \{\mathtt{C}_{I_\downarrow}, \mathtt{C}_I, \mathtt{C}_{I_\uparrow}\}. \end{aligned}$$

- The stack symbol $J^\blacktriangle$ means that we store the elapsed time in the global clock $x_J^\blacktriangle$ after a frame that has $J^\blacktriangle$ pushed. So does the stack symbol $J^\blacktriangledown$. (However, as we will see later, we do not use the symbol $J^\blacktriangledown$.)
- The stack symbol J° means that we store the value of the local clock z in the global clock $\mathtt{C}_J$.
- The stack symbol $J^{\blacktriangle,\circ}$ means that we store (1) the elapsed time in the global clock $x_J^\blacktriangle$ after a frame that has $J^\blacktriangle$ pushed and (2) the value of the local clock z in the global clock $\mathtt{C}_J^\blacktriangle$. So does the stack symbol $J^{\blacktriangledown,\circ}$.

On the basis of the construction of the previous section, we define the set of transition rules Δ_D as follows.

Case $p \xrightarrow[\alpha]{\mathsf{reset}(z)} q \in \Delta_B$**:** On the predicting MTPDA D, when we reset the local clock z, we mark a frame by adding $\cdot^\circ$ and reset the global clock $\mathtt{C}$ to reflect the value of the local clock z of the current stack top frame. We carry out this by adding the following transition rules to Δ_D:

$$p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J)\,⨟\,\mathsf{rew}(\gamma,J^\circ)\,⨟\,\tau} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^\circ)\,⨟\,\tau} q,$$
$$p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^\blacktriangle)\,⨟\,\mathsf{rew}(\gamma,J^{\blacktriangle,\circ})\,⨟\,\tau} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\blacktriangle,\circ})\,⨟\,\tau} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\blacktriangledown,\circ})\,⨟\,\tau} q$$

where $\tau \triangleq \mathsf{reset}(z) ⨟ \mathsf{reset}(\mathtt{C}_J)$.

Case $p \xrightarrow[\alpha]{\mathsf{push}(\gamma,J)} q \in \Delta_B$**:** We add transition rules to Δ_D as follows:

- The following one corresponds to pushing a frame onto the empty stack:

$$p \xrightarrow[\alpha]{\mathsf{check}(\epsilon)\,⨟\,\mathsf{push}(\gamma,J^{\blacktriangle,\circ})\,⨟\,\mathsf{reset}(x_J^\blacktriangle)\,⨟\,\mathsf{reset}(\mathtt{C}_J)} q.$$

- The following rules correspond to pushing a frame onto the stack whose top symbol has J:

$$
\begin{gathered}
p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\blacktriangle}) \fatsemi \tau} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\blacktriangle,\circ}) \fatsemi \mathsf{rew}(\gamma,J^{\blacktriangle}) \fatsemi \tau} q, \\
p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J) \fatsemi \tau} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\circ}) \fatsemi \mathsf{rew}(\gamma,J) \fatsemi \tau} q, \\
p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\blacktriangledown,\circ}) \fatsemi \mathsf{rew}(\gamma,J) \fatsemi \tau} q
\end{gathered}
$$

  where $\tau = \mathsf{push}(\gamma, J^{\blacktriangledown,\circ}) \fatsemi \mathsf{reset}(x_J^{\blacktriangledown}) \fatsemi \mathsf{reset}(\mathtt{C}_J)$.

- The following rules correspond to pushing a frame onto the stack whose top symbol has K and $J \sqsubset K$:

$$
\begin{gathered}
p \xrightarrow[\alpha]{\mathsf{check}(\gamma,K^{\blacktriangle}) \fatsemi \tau} q, p \xrightarrow[\alpha]{\mathsf{check}(\gamma,K^{\blacktriangle,\circ}) \fatsemi \tau} q, \\
p \xrightarrow[\alpha]{\mathsf{check}(\gamma,K) \fatsemi \tau} q, p \xrightarrow[\alpha]{\mathsf{check}(\gamma,K^{\circ}) \fatsemi \tau} q, \\
p \xrightarrow[\alpha]{\mathsf{check}(\gamma,K^{\blacktriangledown,\circ}) \fatsemi \tau} q
\end{gathered}
$$

  where $\tau = \mathsf{push}(\gamma, J^{\blacktriangle,\circ}) \fatsemi \mathsf{reset}(x_J^{\blacktriangle}) \fatsemi \mathsf{reset}(\mathtt{C}_J)$.

Case $p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J) \fatsemi z-x\in_? J \fatsemi \mathsf{pop}(\gamma,J)} q \in \Delta_B$: We add transition rules to Δ_D as follows:

$$
\begin{gathered}
p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\blacktriangle}) \fatsemi x_J^{\blacktriangle}-x\in_? J \fatsemi \mathsf{pop}(\gamma,J^{\blacktriangle})} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\blacktriangle,\circ}) \fatsemi x_J^{\blacktriangle}-x\in_? J \fatsemi \mathsf{pop}(\gamma,J^{\blacktriangle,\circ})} q, \\
p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J) \fatsemi \mathsf{pop}(\gamma,J)} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\circ}) \fatsemi \mathsf{pop}(\gamma,J^{\circ})} q, \\
p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\blacktriangledown,\circ}) \fatsemi x_J^{\blacktriangledown}-x\in_? J \fatsemi \mathsf{pop}(\gamma,J^{\blacktriangledown,\circ})} q.
\end{gathered}
$$

Case $p \xrightarrow[\alpha]{z-x\in_? I} q \in \Delta_B$: We add the following transition rules:

$$
\begin{gathered}
p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\blacktriangle,\circ}) \fatsemi \mathtt{C}_J-x\in_? I} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\blacktriangledown,\circ}) \fatsemi \mathtt{C}_J-x\in_? I} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma,J^{\circ}) \fatsemi \mathtt{C}_J-x\in_? I} q, \\
p \xrightarrow[\alpha]{\mathsf{check}(\gamma,I^{\blacktriangle})} q, \quad p \xrightarrow[\alpha]{\mathsf{check}(\gamma,I)} q.
\end{gathered}
$$

Other Rules $p \xrightarrow[\alpha]{\tau} q \in \Delta_B$: For the rules other than those above, we add the same rules to Δ_D.

4.6.1 Properties of Predicting MTPDA D

We define the well-formedness of a stack of the predicting MTPDA D and show the property corresponding to Lemma 4.13. A stack w of D is *well-formed* if it satisfies the following condition:

$$w \in \mathcal{F}(I_{\uparrow}) \cdot \mathcal{F}(I) \cdot \mathcal{F}(I_{\downarrow})$$

where $\mathcal{F}(J)$ is defined as follows:

$$\mathcal{F}(J) = \{\epsilon\} \ \cup \ \Upsilon(J^{\blacktriangle,\circ}) \ \cup \ \Upsilon(J^{\blacktriangle}) \cdot \Upsilon(J)^* \ \cup \ \Upsilon(J^{\blacktriangle}) \cdot \Upsilon(J)^* \cdot (\Upsilon(J^{\circ}) \cup \Upsilon(J^{\blacktriangledown,\circ})).$$

Recall that $\Upsilon(J)$ is defined as $\Upsilon(J) \triangleq (\Gamma \times \{J\}) \times (\mathcal{Z} \to \mathbb{R}_{\geq 0})$.

Lemma 4.17. If a configuration $\langle q, \eta, w \rangle$ of D is reachable from the initial configuration (i.e., $\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon \rangle \Rightarrow_D^* \langle q, \eta, w \rangle$), then w is a well-formed stack.

Proof. This lemma can be shown in the same way as Lemma 4.13. □

If the top frame of a stack is one of $\langle(\gamma, J^{\blacktriangle,\circ}), \mu\rangle$, $\langle(\gamma, J^{\blacktriangledown,\circ}), \mu\rangle$, or $\langle(\gamma, J^{\circ}), \mu\rangle$, then the value of the local clock z is recorded in the corresponding auxiliary clock.

Lemma 4.18. Let us consider the following computations of D:

$$\begin{aligned}&\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_D^* \langle q_1, \eta_1, \ldots \langle(\gamma_1, J^{\blacktriangle,\circ}), \mu_1\rangle\rangle,\\ &\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_D^* \langle q_2, \eta_2, \ldots \langle(\gamma_2, J^{\blacktriangledown,\circ}), \mu_2\rangle\rangle,\\ &\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_D^* \langle q_3, \eta_3, \ldots \langle(\gamma_3, J^{\circ}), \mu_3\rangle\rangle.\end{aligned}$$

For these computations, we have $\mu_1(z) = \eta_1(\mathsf{C}_J)$, $\mu_2(z) = \eta_2(\mathsf{C}_J)$, and $\mu_3(z) = \eta_3(\mathsf{C}_J)$.

Proof. Each case can be easily shown by induction on the length of the computation. □

Next, we consider the case that the top frame a stack is of the form $\langle(\gamma, J), \mu\rangle$. For this case, we have a property that are similar to Lemma 4.12. To show the property, we prove the technical lemma corresponding to Lemma 4.14.

Lemma 4.19. Let us consider the following computation π for a configuration $\boldsymbol{c'}$:

$$\pi = \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_D^* \langle q', \eta', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i'\rangle \ldots \langle(\gamma, J), \mu'\rangle\rangle = \boldsymbol{c'}.$$

We can decompose π as follows:

$$\begin{aligned}&\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^*\\ &\langle p, \eta, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots \langle(\gamma_j, J^{\blacktriangledown,\circ}), \mu_j\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma_j, J^{\blacktriangledown,\circ}) \fatsemi\ x_J^{\blacktriangledown} - x \in_? J \fatsemi\ \mathsf{pop}(\gamma_j, J^{\blacktriangledown,\circ})}\\ &\langle q, \eta, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots\rangle \underset{\sharp}{\Rightarrow}^*\\ &\langle q', \eta', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i'\rangle \ldots \langle(\gamma, J), \mu'\rangle\rangle.\end{aligned}$$

Especially, among the part of the computation $\underset{\sharp}{\Rightarrow}^*$, we do not reset $\mathsf{reset}(z)$ against the frame that contains the stack symbol (γ, J).

Proof. We can decompose π as follows in the same argument as Lemma 4.14:

$$\begin{aligned}&\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^*\\ &\langle \ell, \eta, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots \langle(\gamma_J, J^{\blacktriangledown,\circ}), \_\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma_J, J^{\blacktriangledown,\circ}) \fatsemi\ x_J^{\blacktriangledown} - x \in_? J \fatsemi\ \mathsf{pop}(\gamma_J, J^{\blacktriangledown,\circ})}\\ &\langle q, \eta, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots\rangle \underset{\Diamond}{\Rightarrow}^* \langle q', \eta', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i'\rangle \ldots \langle(\gamma, J), \mu'\rangle\rangle.\end{aligned}$$

However, this does not ensure that there are no transitions with $\mathsf{reset}(z)$ in $\underset{\Diamond}{\Rightarrow}^*$. If there are no such transitions in $\underset{\Diamond}{\Rightarrow}^*$, then the proof is finished. Therefore, now we assume that $\underset{\Diamond}{\Rightarrow}^*$ has such a transition and focus on the transition:

$$\begin{aligned}&\langle q, \eta, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots\rangle \Rightarrow^*\\ &\langle \ell, \nu, \ldots \langle(\gamma_i, J^{\blacktriangle}), \_\rangle \ldots \langle(\gamma, J), \_\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma, J) \fatsemi\ \mathsf{rew}(\gamma, J^{\circ}) \fatsemi\ \mathsf{reset}(z) \fatsemi\ \mathsf{reset}(\mathsf{C}_J)}\\ &\langle p', \nu', \ldots \langle(\gamma_i, J^{\blacktriangle}), \_\rangle \ldots \langle(\gamma, J^{\circ}), \_\rangle\rangle \underset{\clubsuit}{\Rightarrow}^* \langle q', \eta', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i'\rangle \ldots \langle(\gamma, J), \mu'\rangle\rangle.\end{aligned}$$

It should be noted that there are no transitions with $\mathsf{reset}(z)$ in $\underset{\clubsuit}{\Rightarrow}^*$. To rewrite J° to J, the sequence of transitions $\underset{\clubsuit}{\Rightarrow}^*$ must be decomposed as follows:

$$\begin{aligned}&\langle p', \nu', \ldots \langle(\gamma_i, J^{\blacktriangle}), \_\rangle \ldots \langle(\gamma, J^{\circ}), \_\rangle\rangle \Rightarrow^* \xrightarrow{\mathsf{check}(\gamma, J^{\circ}) \fatsemi\ \mathsf{rew}(\gamma, J) \fatsemi\ \mathsf{push}(\gamma', J^{\blacktriangledown,\circ}) \fatsemi\ \mathsf{reset}(x_J^{\blacktriangledown}) \fatsemi\ \mathsf{reset}(\mathsf{C}_J)} \Rightarrow^*\\ &\langle p, \eta, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots \langle(\gamma_j, J^{\blacktriangledown,\circ}), \mu_j\rangle\rangle \xrightarrow{\mathsf{check}(\gamma_j, J^{\blacktriangledown,\circ}) \fatsemi\ x_J^{\blacktriangledown} - x \in_? J \fatsemi\ \mathsf{pop}(\gamma_j, J^{\blacktriangledown,\circ})}\\ &\langle q, \eta, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots\rangle \underset{\sharp}{\Rightarrow}^* \langle q', \eta', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i'\rangle \ldots \langle(\gamma, J), \mu'\rangle\rangle.\end{aligned}$$

Combining these transitions, we have the following desired decomposition:

$$\begin{array}{l}
\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \\
\langle p, \eta, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots \langle(\gamma_j, J^{\blacktriangledown,\circ}), \mu_j\rangle\rangle \xrightarrow{\mathsf{check}(\gamma_j, J^{\blacktriangledown,\circ})\,;\, x_J^{\blacktriangledown} - x \in_? J\,;\, \mathsf{pop}(\gamma_j, J^{\blacktriangledown,\circ})} \\
\langle q, \eta, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots\rangle \underset{\sharp}{\Rightarrow^*} \langle q', \eta', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i'\rangle \ldots \langle(\gamma, J), \mu'\rangle\rangle
\end{array}$$

where there are no transition with $\mathsf{reset}(z)$ in $\underset{\sharp}{\Rightarrow^*}$. □

Lemma 4.20. Let us consider the following computation:

$$\pi = \langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_D^* \langle q', \eta', \ldots \langle(\gamma, J), \mu'\rangle\rangle \Rightarrow_D^* \langle q_F, \eta_F, \epsilon\rangle.$$

For this computation, we have $\mu'(z) - \eta'(x) \in J$.

Proof. Applying Lemma 4.19 to π, we have the following decomposition:

$$\begin{array}{l}
\langle q_{\mathrm{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \\
\langle p, \eta, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots \langle(\gamma_j, J^{\blacktriangledown,\circ}), \mu_j\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma_j, J^{\blacktriangledown,\circ})\,;\, x_J^{\blacktriangledown} - x \in_? J\,;\, \mathsf{pop}(\gamma_j, J^{\blacktriangledown,\circ})} \\
\langle q, \eta, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots\rangle \underset{\sharp}{\Rightarrow^*} \\
\langle q', \eta', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i'\rangle \ldots \langle(\gamma, J), \mu'\rangle\rangle \Rightarrow^* \\
\langle q'', \eta'', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i''\rangle\rangle \xrightarrow{\mathsf{check}(\gamma_i, J^{\blacktriangle})\,;\, x_J^{\blacktriangle} - x \in_? J\,;\, \mathsf{pop}(\gamma_i, J^{\blacktriangle})} \Rightarrow^* \langle q_F, \eta_F, \epsilon\rangle.
\end{array}$$

By the definition of pop, we have $\eta(x_J^{\blacktriangledown}) - \eta(x) \in J$ and $\eta''(x_J^{\blacktriangle}) - \eta''(x) \in J$. Therefore, we prove the following to show $\mu'(z) - \eta'(x) \in J$:

$$\eta(x_J^{\blacktriangledown}) - \eta(x) \leq \mu'(z) - \eta'(x) \leq \eta''(x_J^{\blacktriangle}) - \eta''(x).$$

We assume that δ is the elapsed time among $\langle q, \eta, \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \ldots \langle(\gamma, J), \mu\rangle \ldots\rangle \underset{\sharp}{\Rightarrow^*} \langle q', \eta', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i'\rangle \ldots \langle(\gamma, J), \mu'\rangle\rangle$.

$$\begin{array}{rcll}
\eta(x_J^{\blacktriangledown}) - \eta(x) & = & (\eta + \delta)(x_J^{\blacktriangledown}) - (\eta + \delta)(x) & \\
& \leq & (\eta + \delta)(x_J^{\blacktriangledown}) - \eta'(x) & \\
& \leq & (\mu(z) + \delta) - \eta'(x) & [\because \eta(x_J^{\blacktriangledown}) \leq \mu(z)] \\
& = & \mu'(z) - \eta'(x). & \left[\because \begin{array}{l}\text{we do not perform } \mathsf{reset}(z) \\ \text{against the frame containing } \langle\gamma, J\rangle\end{array}\right]
\end{array}$$

We assume that δ' is the elapsed time among $\langle q', \eta', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i'\rangle \ldots \langle(\gamma, J), \mu'\rangle\rangle \Rightarrow^* \langle q'', \eta'', \ldots \langle(\gamma_i, J^{\blacktriangle}), \mu_i''\rangle\rangle$.

$$\begin{array}{rcll}
\mu'(z) - \eta'(x) & = & (\mu' + \delta')(z) - (\eta' + \delta')(x) & \\
& \leq & (\mu' + \delta')(z) - \eta''(x) & \\
& \leq & (\mu_i' + \delta')(z) - \eta''(x) & \\
& \leq & (\eta' + \delta')(x_J^{\blacktriangle}) - \eta''(x) & [\because \mu_i'(z) \leq \eta'(x_J^{\blacktriangle})] \\
& = & \eta''(x_J^{\blacktriangle}) - \eta''(x). &
\end{array}$$

□

Finally, we consider the case that the top frame a stack is of the form $\langle(\gamma, J^{\blacktriangle}), \mu\rangle$. The lemmas corresponding to Lemma 4.21 and 4.22 hold. Since both lemmas can be shown by the same argument as the corresponding lemmas, we omit the proof.

Lemma 4.21. Let us consider the following computation π for a configuration $\boldsymbol{c}'$:

$$\pi = \langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_D^* \langle q', \eta', \dots \langle(\gamma_i, J^{\blacktriangle}), \mu_i'\rangle\rangle = \boldsymbol{c}'.$$

We can decompose π as follows:

$$\begin{aligned}
&\langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow^* \\
&\langle p, \eta, \dots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \dots \langle(\gamma_j, J^{\blacktriangledown,\circ}), \mu_j\rangle\rangle \xrightarrow[\alpha]{\mathsf{check}(\gamma_j, J^{\blacktriangledown,\circ})\fatsemi\ x_J^{\blacktriangledown} - x \in_? J \fatsemi\ \mathsf{pop}(\gamma_j, J^{\blacktriangledown,\circ})} \\
&\langle q, \eta, \dots \langle(\gamma_i, J^{\blacktriangle}), \mu_i\rangle \dots\rangle \underset{\sharp}{\Rightarrow}^* \\
&\langle q', \eta', \dots \langle(\gamma_i, J^{\blacktriangle}), \mu_i'\rangle\rangle.
\end{aligned}$$

Especially, among the part of the computation $\underset{\sharp}{\Rightarrow}^*$, we do not reset $\mathsf{reset}(z)$ against the frame that contains the stack symbol (γ, J).

Lemma 4.22. Let us consider the following computation:

$$\pi = \langle q_{\text{init}}, \mathbf{0}, \epsilon\rangle \Rightarrow_D^* \langle p, \eta', \dots \langle(\gamma, J^{\blacktriangle}), \mu'\rangle\rangle \Rightarrow_D^* \langle q_F, \eta_F, \epsilon\rangle.$$

For this computation, we have $\mu'(z) - \eta'(x) \in J$.

4.6.2 Language Equivalence between Predicting MTPDA B and D

In order to show $L_\epsilon(B) = L_\epsilon(D)$, we define the following relation between configurations of B and D:

$$\langle q_B, \nu_B, w_B\rangle \approx \langle q_D, \nu_D, w_D\rangle \overset{\text{def}}{\iff} q_B = q_D \ \wedge\ \nu_B = \nu_D \restriction \mathcal{X} \ \wedge\ w_B = \psi(w_D)$$

where $\psi : \Gamma_D^* \to (\Gamma \times \{I_\downarrow, I, I_\uparrow\})^*$ is a projection defined as follows:

$$\begin{aligned}
&\psi((\gamma, J)) = (\gamma, J), \quad \psi((\gamma, J^{\blacktriangle})) = (\gamma, J), \quad \psi((\gamma, J^{\blacktriangledown})) = (\gamma, J), \\
&\psi((\gamma, J^{\circ})) = (\gamma, J), \quad \psi((\gamma, J^{\blacktriangle,\circ})) = (\gamma, J), \quad \psi((\gamma, J^{\blacktriangledown,\circ})) = (\gamma, J).
\end{aligned}$$

On the basis of this relation, we can show $L_\epsilon(B) \subseteq L_\epsilon(D)$ and $L_\epsilon(B) \supseteq L_\epsilon(D)$ in the similar way to the proofs of Lemma 4.16 and 4.15 because we already showed Lemma 4.17, 4.18, 4.20, and 4.22 that correspond to Lemma 4.13, 4.11, and 4.12.

Therefore, we obtain the following result that states, for a given MTPDA A, we can remove a diagonal constraint of the form $z - x \in_? I$ of A while preserving its language.

Theorem 4.2. Let A be an MTPDA and $z - x \in_? I$ be an action that compares the local clock z and global clock x. There is an MTPDA D that satisfies the following conditions:

- $L_\epsilon(A) = L_\epsilon(D)$.
- There are no transition rules that contain an action $z - x \in_? I$.

4.7 Untiming Theorem of MTPDA

Summarizing all the above discussion, we prove the untiming theorem of MTPDA.

Theorem 4.3. Let A be an MTPDA.

There is a pushdown timed automaton E such that $L_\epsilon(A) = L_\epsilon(E)$.

Proof. Applying Lemma 4.6 to the MTPDA A, we have an MTPDA B such that

- $L_\epsilon(A) = L_\epsilon(B)$ and

- There are no transitions that have an action $x - z \in_? I$ or $z - z' \in_? I$ where x is a global clock and z, z' are local clocks.

Applying Theorem 4.2 to B until there are no transition rules with an action of the form $z - x \in_? I$, we have a predicting MTPDA C such that

- $L_\epsilon(B) = L_\epsilon(C)$ and
- There are no transitions that have an action $z - x \in_? I$, $x - z \in_? I$, or $z - z' \in_? I$ where x is a global clock and z, z' are local clocks.

From the MTPDA $C = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta_C)$, we construct the following 0-MTPDA D:

$$D = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \emptyset, \Delta_D)$$

where Δ_D is defined by removing local clock resetting from Δ_C:

$$\Delta_D \triangleq \left\{ p \xrightarrow[\alpha]{\tau} q \in \Delta_C : \tau \neq \mathsf{reset}(z), z \in \mathcal{Z} \right\}.$$

It is clear that $L_\epsilon(C) = L_\epsilon(D)$.

Since pushdown timed automata do not allow diagonal constraints of the form $x_1 - x_2 \in_? I$, we should remove such actions from D. To this end, we can use the same construction to remove diagonal constraints from timed automata; therefore, we obtain a pushdown timed automaton E such that $L_\epsilon(A) = L_\epsilon(E)$. □

Combining the above theorem and Proposition 4.2, we have the following form of the untiming theorem of MTPDA.

Corollary 4.1. Let A be an MTPDA. There is a pushdown timed automaton E such that $L(A) = L(E)$.

The untiming theorem also implies the following time complexity result for MTPDA.

Corollary 4.2. The emptiness problem and location reachability problem of MTPDA are EXPTIME-complete.

Proof. Since we can reduce the emptiness problem to the location reachability problem in a linear time and vice versa, it suffices to show the EXPTIME-completeness of the emptiness problem.

First, we show the EXPTIME-hardness. This is immediately shown by Corollary 3.1 of Chapter 3 because any PTA is an MTPDA.

Next, we show the emptiness problem is in EXPTIME. Let $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta)$ be an MTPDA. Let K be the numbers of transition rules of the form $p \xrightarrow[\alpha]{z-x\in_? I} q$ where $z \in \mathcal{Z}$ and $x \in \mathcal{X}$. Applying Lemma 4.6 and 4.1, we can construct an MTPDA A_0 such that:

- if A_0 has a transition rule with $c_1 - c_2 \in_? I$, then c_1 is a local clock and c_2 is a global clock;
- the numbers of states and stack symbols of A_0 are exponential in the size of A;
- the numbers of global clocks and local clocks of A_0 are linear in the size of A.

In order to remove transition rules with actions of the form $z - x \in_? I$, we apply Theorem 4.2 about $O(K)$-times. Each applying the theorem to an MTPDA causes exponential increases in the numbers of states and stack symbols and a linear increase in the number of global clocks. Therefore, after we remove all the transition rules with actions of the form $z - x \in_? I$, then the numbers of states and stack symbols of the obtained PTA is exponential in the size of the original MTPDA A and the number of global clocks is linear in the size of A.

Since the emptiness problem of a PTA is solved time linar in the numbers of its states and stack symbols and exponential in the numbers of clocks, the language emptiness problem of the untimed PTA can be solved in exponential time with respect to the size of the input MTPDA A. $\square$

Chapter 5

Synchronized Recursive Timed Automata

This chapter presents a class of timed pushdown automata, *synchronized recursive timed automata (SRTA)*, and we study its expressiveness and decidability. This chapter is based on our previous work [UM15, UM18].

Expressiveness. In comparison with existing classes of timed pushdown automata—PTA, DTPDA, and MTPDA—, SRTA have novel constraints, *fractional constraints*—formulae of the form $\mathit{frac}(x) = 0$ and $\mathit{frac}(x) < \mathit{frac}(y)$ ($\mathit{frac}(x)$ is the fractional part of a clock x)—for checking the fractional parts of clocks. Owing to fractional constraints, the class of SRTA is more expressive than the existing classes of PTA, DTPDA, and MTPDA. Indeed, an SRTA accepts the following timed language L_{SRTA} which cannot be recognized by any PTA:

$$L_{\mathrm{SRTA}} \triangleq \left\{(a, t_1)(a, t_2) \dots (a, t_n)(b, t'_n) \dots (b, t'_2)(b, t'_1) : t'_i - t_i \in \mathbb{N}\right\}.$$

We will formally show that the above language can be accepted by an SRTA and not by any PTA.

Decidability of Reachability Problem. Even though SRTA extend MTPDA due to the presence of fractional constraints, the location reachability problem of SRTA is decidable and EXPTIME-complete. Furthermore, we show the decidability of the configuration reachability problem of SRTA. For a given SRTA and configuration $\langle q, w\rangle$, the configuration reachability problem $\mathbf{c}_{\mathrm{init}} \to^*_? \langle q, w\rangle$ decides whether we can reach the configuration $\langle q, w\rangle$ from the initial configuration $\mathbf{c}_{\mathrm{init}}$. Although the configuration reachability problem of pushdown automata is straightforwardly reduced to the location reachability problem of them, such a reduction does not hold on SRTA due to the unboundedness and denseness of real numbers. Indeed, the configuration reachability problem of TPDA was not considered by Abdulla et al. in [AAS12a].

Our decidability proof of the configuration reachability problem is organized as follows. The reader will find the detailed overview of our proof in Section 5.3. In Section 5.1, we introduce SRTA and give the standard semantics of SRTA. In order to reduce the configuration reachability problem of the standard semantics to that of a semantics defined as a pushdown system, we need to remove the entire stack modification of timed transitions $\langle q, \langle\gamma_1, \nu_1\rangle \dots \langle\gamma_n, \nu_n\rangle\rangle \overset{\delta}{\rightsquigarrow} \langle q, \langle\gamma_1, \nu_1 + \delta\rangle \dots \langle\gamma_n, \nu_n + \delta\rangle\rangle$ and the unboundedness and denseness of real numbers. In Section 5.4, to remove the entire stack modification of timed transitions, we use the technique called lazy time elapsing that was introduced by Abdulla et al. to show the decidability of the location reachability problem of DTPDA in [AAS12a]. In Section 5.5, we remove the unboundedness of real numbers by introducing collapsed real numbers. In Section 5.6, we remove the denseness of real numbers with the formalization of the region of Abdulla et al. Through Section 5.4 to 5.6, we can give a semantics defined as a pushdown system that corresponds to the standard semantics and it leads to the decidability of the configuration reachability problem of SRTA.

We compare the conventional region of timed automata given by Alur and Dill in [AD94] and the region designed by Abdulla et al. for DTPDA in Section 5.7. Through this comparison, we find out that a key technical lemma fails on the region of Alur and Dill.

Basic Notation

Bounded Intervals. In this chapter, unlike the previous chapters, we consider bounded intervals I defined as follows:

$$I ::= [a:b] \mid (a:b)$$

where $a, b \in \mathbb{N}$. Therefore, if we call I interval, then the (bounded) interval I is not an unbounded intervals such as $(a:\omega)$, $(-\omega:b]$, etc.

We use $\mathbb{I}$ to denote the set of bounded intervals and $\mathbb{I}^\omega$ to denote the set of intervals including unbounded intervals:

$$\mathbb{I} = \{(a:b), [a:b] : a, b \in \mathbb{N}\}, \qquad \mathbb{I}^\omega = \{(a:b), [a:b], (-\omega:a), (a:\omega) : a, b \in \mathbb{N}\}.$$

Pushdown Systems.

A pushdown system (PDS) is a triple $(Q, \Gamma, \hookrightarrow)$ where Q is a finite set of control locations, Γ is a (possibly infinite) stack alphabet, and $\hookrightarrow \subseteq (Q \times \Gamma^*) \times (Q \times \Gamma^*)$ is a set of transition rules. A configuration is a pair $\langle q, w\rangle$ of a location $q \in Q$ and a stack $w \in \Gamma^*$. A one-step transition $\langle q, w\,v\rangle \to \langle q', w\,v'\rangle$ is defined if $\langle q, v\rangle \hookrightarrow \langle q', v'\rangle$. We also write $w \to w'$ by omitting locations if the locations are irrelevant. A PDS is called a *finite* PDS if its set of transition rules is finite. Otherwise, it is called an *infinite* PDS. Note that our formalization allows multiple popping rather than single element popping at each single move. A PDS is called a *normalized* PDS if the set of transition rules $\hookrightarrow$ is a subset of $(Q \times \Gamma) \times (Q \times \Gamma^*)$.

For given configurations $\mathbf{c}_1$ and $\mathbf{c}_2$, the configuration reachability problem asks if $\mathbf{c}_1 \to^* \mathbf{c}_2$ holds. The configuration reachability problem of finite normalized PDS is in PTime [BEM97, FWW97]. Since we can translate a finite PDS to the corresponding finite normalized PDS while preserving the configuration reachability, the configuration reachability problem of finite PDS is also decidable.

5.1 Synchronized Recursive Timed Automata

We introduce synchronized recursive timed automata (SRTA) and define the standard semantics of SRTA called Stnd.

Clock Constraints. Let $\mathcal{C}$ be a finite set of clocks. The set $\Phi_{\mathcal{C}}$ of clock constraints is given by:

$$\varphi ::= c \in_? I \mid \mathit{frac}(x) = 0 \mid \mathit{frac}(c) \bowtie \mathit{frac}(c') \mid \varphi \wedge \varphi \mid \neg\varphi$$

where $c, c' \in \mathcal{C}$, $I \in \mathbb{I}$ is a bounded interval, and $\bowtie \in \{<, =, >\}$.

For a constraint $\varphi \in \Phi_{\mathcal{C}}$ and valuation $\nu : \mathcal{C} \to \mathbb{R}_{\geq 0}$, we write $\nu \models \varphi$ if φ holds when clocks are replaced by the values of ν: e.g., $\nu \models c \in_? I$ if $\nu(x) \in I$, $\nu \models \mathit{frac}(c) = 0$ if $\mathit{frac}(\nu(c)) = 0$. The fractional constraints $\mathit{frac}(c) = 0$ and $\mathit{frac}(c) \bowtie \mathit{frac}(c')$ are a novel feature against previous pushdown-extensions of timed automata.

Definition 5.1 (Synchronized Recursive Timed Automata)**.** A synchronized recursive timed automaton (SRTA) is a 7-tuple $\mathcal{A} = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ where

- Q is a finite set of control locations, q_{init} is the initial location, $F \subseteq Q$ is a set of accepting locations,
- Σ is a finite input alphabet, Γ is a finite set of stack symbols,
- $\mathcal{X}$ is a finite set of clocks, and
- $\Delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Act_{\text{SRTA}} \times Q$ is a finite set of *discrete transition rules.*
  - To denote a transition rule $\langle p, \alpha, \tau, q\rangle \in \Delta$, we also write $p \xrightarrow[\alpha]{\tau} q$.

Act_{SRTA} is the set of actions of SRTA defined as follows:

$$\tau \in Act_{\text{SRTA}} ::= \textsf{push}(\gamma) \mid \textsf{pop}(\gamma) \mid \textsf{dig}(x, y) \mid x \leftarrow I \mid \textsf{check}(\varphi)$$

where $\gamma \in \Gamma, x, y \in \mathcal{X}, I \in \mathbb{I}$, and $\varphi \in \Phi_{\mathcal{X}}$. ■

We define the standard semantics Stnd of SRTA as an infinite transition system.

Definition 5.2 (Semantics Stnd). A configuration is a pair $\langle q, w\rangle$ of a location q and a stack w in which each frame $\langle\gamma, \nu\rangle$ consists of a stack symbol γ and a concrete valuation $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$. The set of configurations of Stnd is $Q \times (\Gamma \times (\mathcal{X} \to \mathbb{R}_{\geq 0}))^*$.

For an action $\tau \in Act_{\text{SRTA}}$, we define a *discrete transition* $w \xrightarrow{\tau} w'$ for $w, w' \in (\Gamma \times (\mathcal{X} \to \mathbb{R}_{\geq 0}))^*$ by case analysis on τ as follows:

$$w\,\langle\gamma_1,\nu_1\rangle \xrightarrow{\textsf{push}(\gamma_2)} w\,\langle\gamma_1,\nu_1\rangle\,\langle\gamma_2,\mathbf{0}\rangle, \qquad w\,\langle\gamma_1,\nu_1\rangle\,\langle\gamma_2,\nu_2\rangle \xrightarrow{\textsf{pop}(\gamma_2)} w\,\langle\gamma_1,\nu_2\rangle,$$

$$\frac{\nu_2' = \nu_2[x := \nu_1(y)]}{w\,\langle\gamma_1,\nu_1\rangle\,\langle\gamma_2,\nu_2\rangle \xrightarrow{\textsf{dig}(x,y)} w\,\langle\gamma_1,\nu_1\rangle\,\langle\gamma_2,\nu_2'\rangle}\ ,$$

$$\frac{r \in I \quad \nu' = \nu[x := r]}{w\,\langle\gamma,\nu\rangle \xrightarrow{x \leftarrow I} w\,\langle\gamma,\nu'\rangle}\ , \qquad \frac{\nu \models \varphi}{w\,\langle\gamma,\nu\rangle \xrightarrow{\textsf{check}(\varphi)} w\,\langle\gamma,\nu\rangle}\ .$$

We note that the **pop**-rule removes the top frame and puts ν_2 to the next frame as follows:

$$\begin{tikzcd}[column sep=tiny] w\langle & \gamma_1 \arrow[rrrrr, bend right] & ,\nu_1\rangle\langle\gamma_2, & \nu_2 \arrow[rrrr, bend left] & \rangle \to & w\langle & \gamma_1, & \nu_2\rangle. \end{tikzcd}$$

In addition to discrete transitions, we allow *timed transitions*:

$$\frac{|w| \geq 1 \quad \delta \in \mathbb{R}_{\geq 0}}{w \overset{\delta}{\rightsquigarrow} w + \delta}\ \textsf{time}$$

where $w + \delta$ is defined as follows:

$$(\langle\gamma_1,\nu_1\rangle\langle\gamma_2,\nu_2\rangle \ldots \langle\gamma_n,\nu_n\rangle) + \delta \triangleq \langle\gamma_1,\nu_1 + \delta\rangle\langle\gamma_2,\nu_2 + \delta\rangle \ldots \langle\gamma_n,\nu_n + \delta\rangle.$$

The operational semantics Stnd of the SRTA $\mathcal{A}$ is defined as a labeled infinite transition system $T_{\mathcal{A}} = (Q \times (\Gamma \times (\mathcal{X} \to \mathbb{R}_{\geq 0}))^*, \to, \rightsquigarrow)$ where the set of discrete transitions $\to$ and the set of timed transitions $\rightsquigarrow$ are defined as follows:

$$\frac{p \xrightarrow[\alpha]{\tau} q \in \Delta \quad w \xrightarrow{\tau} w'}{\langle p, w\rangle \xrightarrow[\alpha]{\tau} \langle q, w'\rangle,} \qquad \frac{\delta \in \mathbb{R}_{\geq 0}}{\langle q, w\rangle \overset{\delta}{\rightsquigarrow} \langle q, w + \delta\rangle.}$$

■

By exploiting the **dig** rule, we can implement useful transition rules, which naturally appeared in the existing models: timed recursive state machines and recursive timed automata [BMP10, TW10]. We call an SRTA equipped with the following extended push and pop rules *extended* SRTA:

$$\frac{\nu_2 = \mathbf{0}_{\mathcal{X}}[X \coloneqq \nu_1]}{w \langle \gamma_1, \nu_1 \rangle \to w \langle \gamma_1, \nu_1 \rangle \langle \gamma_2, \nu_2 \rangle} \ \mathsf{push}(\gamma_2, X)$$

$$\frac{\nu_2' = \nu_2[X \coloneqq \nu_1]}{w \langle \gamma_1, \nu_1 \rangle \langle \gamma_2, \nu_2 \rangle \to w \langle \gamma_1, \nu_2' \rangle} \ \mathsf{pop}(\gamma_2, X)$$

where X is a subset of $\mathcal{X}$ and $\nu[\{x_1, \dots, x_n\} \coloneqq \nu']$ is defined as follows:

$$\nu[\{x_1, x_2, \dots, x_n\} \coloneqq \nu'] \triangleq \nu[x_1 \coloneqq \nu'(x_1)][x_2 \coloneqq \nu'(x_2)] \cdots [x_n \coloneqq \nu'(x_n)].$$

For any extended SRTA, we can construct the corresponding normal SRTA by replacing each extended transition rule with appropriate **dig** rules.

Timed Languages of SRTA. We define the timed language of an SRTA in the same way as timed automata.

For an SRTA $\mathcal{A}$, its timed language, $L(\mathcal{A})$, is defined as follows:

$$L(\mathcal{A}) \triangleq \{\mathsf{tw}(\pi) : \pi = \langle q_{\text{init}}, \langle \bot, \mathbf{0} \rangle \rangle \rightsquigarrow \cdots \to \langle q, w \rangle, q \in F\}.$$

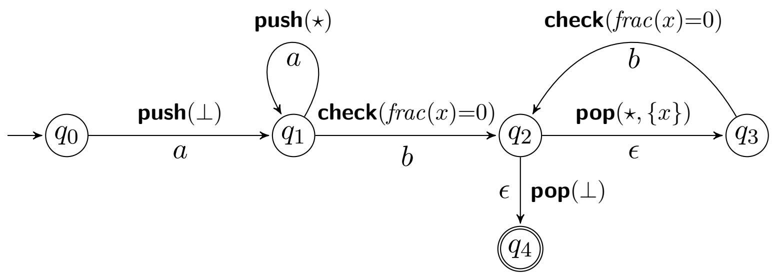
Remark: For the initial configuration $\langle q_{\text{init}}, \langle \bot, \mathbf{0} \rangle \rangle$, we use the special stack symbol $\bot$.

We consider the following timed language:

$$L_{\text{SRTA}} \triangleq \left\{(a, t_1)(a, t_2) \dots (a, t_n)(b, t_n') \dots (b, t_2')(b, t_1') : t_i' - t_i \in \mathbb{N},\ n \geq 1\right\}.$$

It should be noted that if we forget the time stamps from L_{SRTA} then the language $\{a^n b^n : n \geq 1\}$ is a typical context-free language.

To accept the above timed language, let us consider an extended SRTA $\mathcal{A}_{\text{SRTA}} = (\{q_0, \dots, q_4\}, q_0, \{q_4\}, \{a, b\}, \{\bot, \star\}, \{x\}, \Delta)$ where Δ is defined as follows:



SRTA $\mathcal{A}_{\text{SRTA}}$ accepts L_{SRTA} and the following acceptable computation represents the timed word $(a, 0.1)(a, 1.2)(b, 2.2)(b, 3.1) \in L_{\text{SRTA}}$:

$$\langle q_0, \langle \bot, 0 \rangle \rangle \overset{0.1}{\rightsquigarrow} \langle q_0, \langle \bot, 0.1 \rangle \rangle \xrightarrow{a} \langle q_1, \langle \bot, 0.1 \rangle \langle \bot, 0 \rangle \rangle \overset{1.1}{\rightsquigarrow} \langle q_1, \langle \bot, 1.2 \rangle \langle \bot, 1.1 \rangle \rangle \xrightarrow{a}$$

$$\langle q_1, \langle \bot, 1.2 \rangle \langle \bot, 1.1 \rangle \langle \star, 0 \rangle \rangle \overset{1.0}{\rightsquigarrow} \langle q_1, \langle \bot, 2.2 \rangle \langle \bot, 2.1 \rangle \langle \star, 1 \rangle \rangle \xrightarrow{b} \langle q_2, \langle \bot, 2.2 \rangle \langle \bot, 2.1 \rangle \langle \star, 1 \rangle \rangle \overset{0.5}{\rightsquigarrow}$$

$$\langle q_2, \langle \bot, 2.7 \rangle \langle \bot, 2.6 \rangle \langle \star, 1.5 \rangle \rangle \xrightarrow{\epsilon} \langle q_3, \langle \bot, 2.7 \rangle \langle \bot, 2.6 \rangle \rangle \overset{0.4}{\rightsquigarrow} \langle q_3, \langle \bot, 3.1 \rangle \langle \bot, 3 \rangle \rangle \xrightarrow{b}$$

$$\langle q_2, \langle \bot, 3.1 \rangle \langle \bot, 3 \rangle \rangle \overset{0}{\rightsquigarrow} \langle q_2, \langle \bot, 3.1 \rangle \langle \bot, 3 \rangle \rangle \xrightarrow{\epsilon} \langle q_4, \langle \bot, 3.1 \rangle \rangle$$

The fractional constraint $\textsf{\textbf{check}}(\mathit{frac}(x) = 0)$ checks if the fractional part of $t'_i - t_i$ is zero (i.e., $t'_i - t_i \in_? \mathbb{N}$) and is key to excluding runs τ such that $\mathsf{tw}(\tau) \notin L_{\mathrm{SRTA}}$. For example,

$$\begin{aligned}\tau = \langle q_0, \langle \bot, 0\rangle\rangle &\overset{0.1}{\rightsquigarrow} \langle q_0, \langle \bot, 0.1\rangle\rangle \xrightarrow{a} \\ &\langle q_1, \langle \bot, 0.1\rangle\langle \bot, 0\rangle\rangle \overset{0.2}{\rightsquigarrow} \langle q_1, \langle \bot, 0.3\rangle\langle \bot, 0.2\rangle\rangle \not\xrightarrow{b}\end{aligned}$$

~~$\langle q_2, \langle \bot, 0.3\rangle\langle \bot, 0.2\rangle\rangle$~~

and $\mathsf{tw}(\tau) = (a, 0.1)(b, 0.3) \notin L_{\mathrm{SRTA}}$.

5.2 Expressiveness of SRTA

In this section, we study the expressivenss of SRTA by comparing it with PTA and MTPDA. First, we show that SRTA are more expressive than PTA; and then, we show that SRTA can be seen as an extension of MTPDA.

5.2.1 SRTA is More Expressive than PTA: PTA $\subsetneq$ SRTA

We show **PTA** $\subseteq$ **SRTA** and then show the above timed language L_{SRTA} can not be recognized by any PTA.

Theorem 5.1. Let A be a PTA. There is an SRTA B such that $L(A) = L(B)$.

Proof. Since there are no global clocks in SRTA, let us see how we can encode them in SRTA by extended rules $\textsf{\textbf{push}}(\gamma, X)$ and $\textsf{\textbf{pop}}(\gamma, X)$.

Without loss of generality, we can assume that if $p \xrightarrow[\alpha]{x \in_? I} q$ in A, then $I = (a : b)$, $I = [a : b]$, or $I = (a : \omega)$. We denote the PTA A as $A = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ and construct the following SRTA B:

$$B = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma \cup \{\bot\}, \mathcal{X}, \Delta').$$

Our idea to simulate A by B is to represent the values of the global clocks $\mathcal{X}$ of A in the stack top frame of B. To formally state this, we define a correspondence relation between configurations of A and B as follows:

$$\begin{aligned}&\langle q, \nu, \epsilon\rangle \sim \langle q, \langle \bot, \nu\rangle\rangle, \\ &\langle q, \nu, \gamma_1\gamma_2 \ldots \gamma_n\rangle \sim \langle q, \langle \gamma_1, \_\rangle\langle \gamma_2, \_\rangle \ldots \langle \gamma_n, \nu\rangle\rangle.\end{aligned}$$

Now we define Δ' as follows so that the above relation forms a bisimulation between A and B:

$$\frac{p \xrightarrow[\alpha]{\mathsf{nop}} q \in \Delta}{p \xrightarrow[\alpha]{\mathit{frac}(x)=\mathit{frac}(x)} q \in \Delta'}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta}{p \xrightarrow[\alpha]{\textsf{\textbf{push}}(\gamma,\mathcal{X})} q \in \Delta'}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta}{p \xrightarrow[\alpha]{\textsf{\textbf{pop}}(\gamma,\emptyset)} q \in \Delta'}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{reset}(x)} q \in \Delta}{p \xrightarrow[\alpha]{x \leftarrow [0:0]} q \in \Delta'},$$

$$\frac{p \xrightarrow[\alpha]{x \in_? (a:b)} q \in \Delta}{p \xrightarrow[\alpha]{x \in_? (a:b)} q \in \Delta'}, \qquad \frac{p \xrightarrow[\alpha]{x \in_? [a:b]} q \in \Delta}{p \xrightarrow[\alpha]{x \in_? [a:b]} q \in \Delta'}, \qquad \frac{p \xrightarrow[\alpha]{x \in_? (a:\omega)} q \in \Delta}{p \xrightarrow[\alpha]{x \notin_? [0:a]} q \in \Delta'}.$$

(Here we assume that $\mathcal{X} = \{x, \ldots\}$ is not empty).

It should be noted that a transition rule $p \xrightarrow[\alpha]{\textsf{\textbf{push}}(\gamma,\mathcal{X})} q$ induces transitions of the following form:

$$\langle p, \langle \gamma_1, \nu_1\rangle\langle \gamma_2, \nu_2\rangle \ldots \langle \gamma_n, \nu_n\rangle\rangle \xrightarrow[\alpha]{\textsf{\textbf{push}}(\gamma,\mathcal{X})} \langle q, \langle \gamma_1, \nu_1\rangle\langle \gamma_2, \nu_2\rangle \ldots \langle \gamma_n, \nu_n\rangle\langle \gamma, \nu_n\rangle\rangle$$

and a transition rule $p \xrightarrow[\alpha]{\textsf{pop}(\gamma,\emptyset)} q$ induces transitions of the following form:

$$\langle p, \langle \gamma_1, \nu_1\rangle\langle\gamma_2,\nu_2\rangle \dots \langle\gamma_n,\nu_n\rangle\langle\gamma,\nu_{n+1}\rangle\rangle \xrightarrow[\alpha]{\textsf{pop}(\gamma,\emptyset)} \langle q, \langle \gamma_1, \nu_1\rangle\langle\gamma_2,\nu_2\rangle \dots \langle\gamma_n,\nu_{n+1}\rangle\rangle.$$

It can be easily verified that the above relation $\sim$ form a bisimulation between A and B for the set of transition rules Δ'; therefore, we have $L(A) = L(B)$. □

Theorem 5.2. The following SRTA language cannot be accepted by any PTA.

$$L_{\text{SRTA}} = \left\{(a_1,t_1)(a_2,t_2)\dots(a_n,t_n)(b_n,t'_n)\dots(b_2,t'_2)(b_1,t'_1) : n \geq 1,\ t'_i - t_i \in \mathbb{N}\right\}.$$

Proof. To prove this, we use the notations introduced in Section 3.4 of Chapter 3. We prove this by contradiction; we assume there is an m-PTA A_m such that $L_{\text{SRTA}} = L(A_m)$. Since $L_{\text{SRTA}} = L(A_m)$, there is a timed word such that

$$(a_{m+1},t_{m+1})\dots(a_1,t_1)(b_1,t'_1)\dots(b_{m+1},t'_{m+1}) \in L(A_n)$$

where $t'_i - t_i = i$ for any $1 \leq i \leq m+1$. This and Proposition 3.7 of Chapter 3 shows the presence of a timed word

$$(a_{m+1},u_{m+1})\dots(a_1,u_1)(b_1,u'_1)\dots(b_{n+1},u'_{m+1}) \in L(A_n)$$

such that $1 < u'_1 - u_1 < 2$. By the definition of L_{SRTA}, this timed word does not belong to L_{SRTA}. Therefore, $L(A_m) \neq L_{\text{SRTA}}$. □

Combining these results, we obtain the main result about the expressiveness of timed pushdown automata.

Corollary 5.1.

$$\textbf{PTA} = \textbf{DTPDA} = \textbf{MTPDA} \subsetneq \textbf{SRTA}.$$

5.2.2 Alternative View of SRTA

We show that SRTA can be seen as an extension of MTPDA.

Clock Constraints with Diagonal Constraints. Let $\mathcal{C}$ be a finite set of clocks. We extend the set of clock constraints $\Phi_{\mathcal{C}}$ with diagonal constraints.

The set $\Phi^{\text{diag}}_{\mathcal{C}}$ of clock constraints with diagonal constraints is given by:

$$\varphi \in \Phi^{\text{diag}}_{\mathcal{C}} ::= c \in_? I \mid \mathit{frac}(c) = 0 \mid \mathit{frac}(c) \bowtie \mathit{frac}(c') \mid c - c' \bowtie k \mid \varphi \wedge \varphi \mid \neg\varphi$$

where $c, c' \in \mathcal{C}$, I is an bounded interval, and $\bowtie \in \{<,=,>\}$, and $k \in \mathbb{Z}$. A constraint $c - c' \bowtie k$ correspond to a diagonal constraint of timed automata in Section 2.5.1 of Chapter 2.

Let $\varphi \in \Phi^{\text{diag}}_{\mathcal{C}}$ be a constraint and $\nu : \mathcal{C} \to \mathbb{R}_{\geq 0}$ be a valuation. We write $\nu \models \varphi$ if φ holds when clocks are replaced by values of ν. We write $\mathit{Var}(\varphi)$ for the set of clocks of φ: for example, if $\varphi \equiv c_1 \in_? (3:5) \wedge c_2 - c_3 < 3 \wedge \neg(\mathit{frac}(c_4) = 0)$, then $\mathit{Var}(\varphi) = \{c_1, c_2, c_3, c_4\}$.

Extended MTPDA Here we consider an extension of MTPDA, *extended MTPDA (EMTPDA)*. An EMTPDA A is a 8-tuple $A = (Q, q_{\mathrm{init}}, q_F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta)$ where each component except Δ is the same as that of MTPDA and $\Delta \subseteq Q \times \Sigma_\epsilon \times Act_{\mathrm{EMTPDA}} \times Q$ is the set of transition rules. The set of actions of EMTPDA is defined by the following grammar:

$$Act_{\mathrm{EMTPDA}} ::= \mathsf{push}(\gamma) \mid \mathsf{pop}(\gamma) \mid c \hookleftarrow c' \mid c \leftarrow I \mid \mathbf{check}(\varphi)$$

where $\gamma \in \Gamma$, $c, c' \in \mathcal{X} \cup \mathcal{Z}$, I is an bounded interval, and $\varphi \in \Phi^{\mathrm{diag}}_{\mathcal{X}\cup\mathcal{Z}}$. As with MTPDA, A configuration of the EMTPDA A is a triple $\langle q, \nu, \langle\gamma_1, \mu_1\rangle \ldots \langle\gamma_n, \mu_n\rangle\rangle$ where q is a control location, $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a valuation on global clocks, and $\langle\gamma_i, \mu_i\rangle \in \Gamma \times (\mathcal{Z} \to \mathbb{R}_{\geq 0})$ is a stack frame with a stack symbol and valuation on local clocks. The semantics of the transition rules of the form $p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q$ and $p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q$ is defined in the same way as MTPDA. We define the semantics of the other transition rules as follows:

Bounded Update $c \leftarrow I$:

$$\frac{p \xrightarrow[\alpha]{x \leftarrow I} q \in \Delta \quad x \in \mathcal{X} \quad r \in I}{\langle p, \nu, w\rangle \xrightarrow[\alpha]{x \leftarrow I} \langle q, \nu[x := r], w\rangle.} \qquad \frac{p \xrightarrow[\alpha]{z \leftarrow I} q \in \Delta \quad z \in \mathcal{Z} \quad r \in I}{\langle p, \nu, w\langle\gamma, \mu\rangle\rangle \xrightarrow[\alpha]{z \leftarrow I} \langle q, \nu, w\langle\gamma, \mu[z := r]\rangle\rangle,}$$

Checking Clocks $\mathbf{check}(\varphi)$:

$$\frac{p \xrightarrow[\alpha]{\mathbf{check}(\varphi)} q \in \Delta \quad Var(\varphi) \cap \mathcal{Z} = \emptyset \quad \nu \models \varphi.}{\langle p, \nu, w\rangle \xrightarrow[\alpha]{\mathbf{check}(\varphi)} \langle q, \nu, w\rangle,} \qquad \frac{p \xrightarrow[\alpha]{\mathbf{check}(\varphi)} q \in \Delta \quad Var(\varphi) \cap \mathcal{Z} \neq \emptyset \quad \nu \cup \mu \models \varphi}{\langle p, \nu, w\langle\gamma, \mu\rangle\rangle \xrightarrow[\alpha]{\mathbf{check}(\varphi)} \langle q, \nu, w\langle\gamma, \mu\rangle\rangle}$$

Copying Clocks $c_1 \hookleftarrow c_2$:

$$\frac{p \xrightarrow[\alpha]{x \hookleftarrow x'} q \in \Delta \quad x \in \mathcal{X} \quad x' \in \mathcal{X}}{\langle p, \nu, w\rangle \xrightarrow[\alpha]{x \hookleftarrow x'} \langle q, \nu[x := \nu(x')], w\rangle,} \qquad \frac{p \xrightarrow[\alpha]{x \hookleftarrow z} q \in \Delta \quad x \in \mathcal{X} \quad z \in \mathcal{Z}}{\langle p, \nu, w\langle\gamma, \mu\rangle\rangle \xrightarrow[\alpha]{x \hookleftarrow z} \langle q, \nu[x := \mu(z)], w\langle\gamma, \mu\rangle\rangle,}$$

$$\frac{p \xrightarrow[\alpha]{z \hookleftarrow x} q \in \Delta \quad z \in \mathcal{Z} \quad x \in \mathcal{X}}{\langle p, \nu, w\langle\gamma, \mu\rangle\rangle \xrightarrow[\alpha]{z \hookleftarrow x} \langle q, \nu, w\langle\gamma, \mu[z := \nu(x)]\rangle\rangle,} \qquad \frac{p \xrightarrow[\alpha]{z \hookleftarrow z'} q \in \Delta \quad z \in \mathcal{Z} \quad z' \in \mathcal{Z}}{\langle p, \nu, w\langle\gamma, \mu\rangle\rangle \xrightarrow[\alpha]{z \hookleftarrow z'} \langle q, \nu, w\langle\gamma, \mu[z := \mu(z')]\rangle\rangle.}$$

We define the language $L(A)$ of an EMTPDA A in the same way as MTPDA. Extended MTPDA can be seen as follows:

$$\begin{array}{rcl} \text{Extended MTPDA} & = & \text{MTPDA} + \text{fractional constraints } \mathit{frac}(c) = 0 \ \& \ \mathit{frac}(c) \bowtie \mathit{frac}(c') \\ & + & \text{bounded update } c \leftarrow I + \text{value copying } c \hookleftarrow c' \end{array}$$

Since we can simulate the **dig** actions of SRTA by the value copying mechanism of extended MTPDA, we have the following lemma.

Lemma 5.1. Let A be an SRTA. There is an extended MTPDA B such that $L(A) = L(B)$.

Proof. We denote the SRTA A by $A = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ where $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$.

We define the EMTPDA $B = (Q, q_{\mathrm{init}}, F, \Sigma, \Gamma, G_{\mathcal{X}}, L_{\mathcal{X}}, \Delta')$ where $G_{\mathcal{X}} = \{g_x : x \in \mathcal{X}\}$ and $L_{\mathcal{X}} = \{\ell_x : x \in \mathcal{X}\}$. Before we give the definition of Δ', we define a correspondence relation between configurations of A and B as follows:

$$\begin{array}{l} \langle q, \langle\gamma_1, \nu_1\rangle\rangle \sim \langle q, \nu_1, \langle\gamma_1, \text{-}\rangle\rangle, \\ \langle q, \langle\gamma_1, \nu_1\rangle\langle\gamma_2, \nu_2\rangle\rangle \sim \langle q, \nu_2, \langle\gamma_1, \text{-}\rangle\langle\gamma_2, \nu_1\rangle\rangle, \\ \langle q, \langle\gamma_1, \nu_1\rangle\langle\gamma_2, \nu_2\rangle \ldots \langle\gamma_{n-1}, \nu_{n-1}\rangle\langle\gamma_n, \nu_n\rangle\rangle \sim \langle q, \nu_n, \langle\gamma_1, \text{-}\rangle\langle\gamma_2, \nu_1\rangle \ldots \langle\gamma_n, \nu_{n-1}\rangle\rangle. \end{array}$$

Basically, we remember the clock valuation ν_n of the top frame of A in the global clock valuation of B and the clock valuation ν_{n-1} of the frame next to the top of A in the local clock valuation of the top frame of B, and so on. To have the above relation $\sim$ form a bisimulation, we define Δ' as follows:

$$\frac{p \xrightarrow[\alpha]{\textbf{push}(\gamma)} q \in \Delta}{p \xrightarrow[\alpha]{\mathsf{push}(\gamma) \fatsemi\ \ell_{x_1} \leftarrow g_{x_1} \fatsemi\ \cdots \fatsemi\ \ell_{x_k} \leftarrow g_{x_k} \fatsemi\ \mathsf{reset}(x_1) \fatsemi\ \cdots \fatsemi\ \mathsf{reset}(x_k)} q \in \Delta'},$$

$$\frac{p \xrightarrow[\alpha]{\textbf{dig}(x,y)} q \in \Delta}{p \xrightarrow[\alpha]{g_x\ \leftarrow\ \ell_y} q \in \Delta'}, \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta}{p \xrightarrow[\alpha]{\textbf{pop}(\gamma)} q \in \Delta'} \qquad \frac{p \xrightarrow[\alpha]{x \leftarrow I} q \in \Delta}{p \xrightarrow[\alpha]{g_x \leftarrow I} q \in \Delta'}, \qquad \frac{p \xrightarrow[\alpha]{\textbf{check}(\varphi)} q \in \Delta}{p \xrightarrow[\alpha]{\textbf{check}(\mathit{rename}(\varphi))} q \in \Delta'}$$

where the function $\mathit{rename} : \Psi^{\text{diag}}_{\mathcal{X}} \to \Psi^{\text{diag}}_{G_{\mathcal{X}}}$ renames each clock x of an input constraint to the corresponding clock g_x. To simplify the construction, we used atomic transition rules of the form $p \xrightarrow[\alpha]{\tau_1 \fatsemi\ \tau_2 \fatsemi\ \cdots\ \tau_n} q$; however, we can remove this by adding a sequence of transition rules and a fresh clock to ensure the atomicity of the sequence of transitions (see Lemma 4.1 of Chapter 4). $\square$

Conversely, we show the language class of EMTPDA is subsumed by that of SRTA. To this end, we consider an extension of SRTA, *full SRTA*, which are as expressive as SRTA. $\mathit{Act}_{\text{fullSRTA}}$ is the set of actions of a full SRTA defined as follows:

$$\begin{aligned} \tau \in \mathit{Act}_{\text{fullSRTA}} \quad ::= &\quad \textbf{push}(\gamma) \mid \textbf{pop}(\gamma) \mid \textbf{dig}(x, y) \mid x \leftarrow I \mid \textbf{check}(\varphi) \\ \mid &\quad \tau \fatsemi \tau \mid \textbf{top}(\gamma) \mid x \Lleftarrow x' \end{aligned}$$

where $\gamma \in \Gamma$, $x, x' \in \mathcal{X}$, I is an interval, and $\varphi \in \Phi^{\text{diag}}_{\mathcal{X}}$. On full SRTA, we newly allow the following types of transition rules:

Atomic transition An atomic transition rule $p \xrightarrow[\alpha]{\tau_1 \fatsemi\ \tau_2} q$ induces the following transition:

$$\frac{w \xrightarrow{\tau_1} w'' \quad w'' \xrightarrow{\tau_2} w'}{\langle p, w\rangle \xrightarrow[\alpha]{\tau_1 \fatsemi\ \tau_2} \langle q, w'\rangle}.$$

Adding this type of transition rules does not enlarge the expressiveness of SRTA because we can easily remove atomic transitions in the same way as Lemma 4.1 of Chapter 4.

Checking a stack symbol A transition rule $p \xrightarrow[\alpha]{\textbf{top}(\gamma)} q$ checks whether or not the stack symbol of the stack top frame is γ; therefore, the rule induces the following transition:

$$\frac{\gamma' = \gamma}{\langle p, w\langle \gamma', \mu\rangle\rangle \xrightarrow[\alpha]{\textbf{top}(\gamma)} \langle q, w\langle \gamma', \mu\rangle\rangle}.$$

Adding this type of transition rules does not enlarge the expressivenes of SRTA because we can remove such transition rules in the same way as Lemma 4.1 of Chapter 4.

Copying clocks in a single frame A transition rule $p \xrightarrow[\alpha]{x_1 \Lleftarrow x_2} q$, unlike the action **dig**, copies the value of the clock x_2 to the clock x_1 in the stack top frame; therefore, the rule induces the following transition:

$$\langle p, w\langle \gamma, \mu\rangle\rangle \xrightarrow[\alpha]{x_1 \Lleftarrow x_2} \langle q, w\langle \gamma, \mu[x_1 := \mu(x_2)]\rangle\rangle.$$

Adding this type of transition rules does not enlarge the expressivenes of SRTA because we can replace $p \xrightarrow[\alpha]{x_1 \Leftarrow\!\!\dashv x_2} q$ by $p \xrightarrow[\alpha]{\mathsf{push}(\gamma,\mathcal{X})\,⨾\,\mathsf{dig}(x_1,x_2)\,⨾\,\mathsf{pop}(\gamma,\emptyset)} q$.

Diagonal constraints On full SRTA, we allow diagonal constraints $x - y \bowtie k$. This does not enlarge the expressiveness of SRTA; we showed this result in our preliminary work [UM15]. The similar result was already shown on timed automata in [BDFP04]. This is in constast to the result of timed automata that the combination of *un*bounded updates and diagonal constraints enlarge the expressiveness of timed automata and leads to the undecidability of the reachability problem of timed automata [BDFP04].

Therefore, SRTA and full SRTA are equally expressive.

Lemma 5.2. Let A be an extended MTPDA. There is a full SRTA B such that $L(A) = L(B)$.

Proof. We denote the extended MTPDA by $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \mathcal{Z}, \Delta)$. We define the following full SRTA B:

$$B = (Q, q_{\text{init}}, F, \Sigma, \Gamma \cup \{\bot\}, \mathcal{X} \cup \mathcal{Z}, \Delta').$$

Before giving the definition of Δ', we define a correspondence relation between configurations of A and B:

$\langle q, \nu, \epsilon\rangle \sim \langle q, \langle \bot, \theta\rangle\rangle \iff \nu = \theta \restriction \mathcal{X}$,
$\langle q, \nu, \langle \gamma_1, \mu_1\rangle\rangle \sim \langle q, \langle \bot, \_\rangle\langle \gamma_1, \nu \cup \mu_1\rangle\rangle$,
$\langle q, \nu, \langle \gamma_1, \mu_1\rangle\langle \gamma_2, \mu_2\rangle \ldots \langle \gamma_n, \mu_n\rangle\rangle \sim \langle q, \langle \bot, \_\rangle\langle \gamma_1, \theta_1\rangle\langle \gamma_2, \theta_2\rangle \ldots \langle \gamma_n, \nu \cup \mu_n\rangle\rangle \iff \mu_i = (\theta_i \restriction \mathcal{X})$.

To have this relation $\sim$ form a bisimulation between A and B, we define the set of transition rules of B as follows:

$$\frac{p \xrightarrow[\alpha]{\mathsf{push}(\gamma)} q \in \Delta}{p \xrightarrow[\alpha]{\mathsf{push}(\gamma,\mathcal{X})} q \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma)} q \in \Delta}{p \xrightarrow[\alpha]{\mathsf{pop}(\gamma,\mathcal{X})} q \in \Delta',}$$

$$\frac{p \xrightarrow[\alpha]{c \leftarrow\!\!\dashv c'} q \in \Delta \quad \{c, c'\} \cap \mathcal{Z} \neq \emptyset \quad \gamma \in \Gamma}{p \xrightarrow[\alpha]{\mathsf{top}(\gamma)\,⨾\; c \Leftarrow\!\!\dashv c'} q \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{c \leftarrow\!\!\dashv c'} q \in \Delta \quad \{c, c'\} \cap \mathcal{Z} = \emptyset}{p \xrightarrow[\alpha]{c \Leftarrow\!\!\dashv c'} q \in \Delta',}$$

$$\frac{p \xrightarrow[\alpha]{c \leftarrow I} q \in \Delta \quad c \in \mathcal{Z} \quad \gamma \in \Gamma}{p \xrightarrow[\alpha]{\mathsf{top}(\gamma)\,⨾\; c \leftarrow I} q \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{c \leftarrow I} q \in \Delta \quad c \in \mathcal{X}}{p \xrightarrow[\alpha]{c \leftarrow I} q \in \Delta',}$$

$$\frac{p \xrightarrow[\alpha]{\mathsf{check}(\varphi)} q \in \Delta \quad Var(\varphi) \cap \mathcal{Z} \neq \emptyset \quad \gamma \in \Gamma}{p \xrightarrow[\alpha]{\mathsf{top}(\gamma)\,⨾\; \mathsf{check}(\varphi)} q \in \Delta',} \qquad \frac{p \xrightarrow[\alpha]{\mathsf{check}(\varphi)} q \in \Delta \quad Var(\varphi) \cap \mathcal{Z} = \emptyset}{p \xrightarrow[\alpha]{\mathsf{check}(\varphi)} q \in \Delta'.}$$

It can be easily verified that $\sim$ forms a bisimulation between the extended MTPDA A and the extended SRTA B; therefore, we have $L(A) = L(B)$. □

Combining these lemmas, we have the following.

Theorem 5.3. Extended MTPDA and (full) SRTA are equally expressive.

5.3 Overview of Decidability Proof of Configuration Reachability Problem

As an overview of the rest of the present paper, we outline our proof of the decidability of the configuration reachability problem of SRTA.

Configuration Reachability Problem.

For a configuration $\langle q, w\rangle$, the configuration reachability problem $\langle q_{\text{init}}, \langle \bot, \mathbf{0}\rangle\rangle \rightarrow^*_? \langle q, w\rangle$ decides if there is a run from the initial configuration $\langle q_{\text{init}}, \langle \bot, \mathbf{0}\rangle\rangle$ to $\langle q, w\rangle$.

The following is our main result:

Main Result (Corollary 5.2)

The configuration reachability problem of SRTA is decidable.

To show this, we build a semantics called the digitized semantics Digi that can be defined as a finite PDS through Section 5.4 and Section 5.6:

$$\begin{aligned} \textsc{Stnd} &\xrightarrow{\text{Sec.5.4}} \text{Lazy Semantics } \textsc{Lazy} \\ &\xrightarrow{\text{Sec.5.5}} \text{Collapsed Semantics } \textsc{Coll} \\ &\xrightarrow{\text{Sec.5.6}} \text{Digitized Semantics } \textsc{Digi}. \end{aligned}$$

These translations allow reducing the configuration reachability problem of the standard semantics to the configuration reachability problem of the digitized semantics. Since the configuration reachability problem of finite PDS is decidable, we obtain the decidability of the configuration reachability problem of SRTA. We note that our construction is based on the construction of Abdulla et al. in [AAS12a] and the finally obtained finite PDS is basically equivalent to their *symbolic pushdown automaton.*

Let us see the idea of each translation and explain the reason why our approach works well for the configuration reachability problem.

5.3.1 Idea of Each Semantics

Our aim is to reduce the reachability problem of the standard semantics Stnd to the corresponding reachability problem of the digitized semantics Digi. To this end, we remove the following three problems at each step:

1. The entire stack modification of timed transitions:
$$\langle \gamma_1, \nu_1\rangle\langle \gamma_2, \nu_2\rangle \dots \langle \gamma_n, \nu_n\rangle \overset{\delta}{\rightsquigarrow} \langle \gamma_1, \nu_1+\delta\rangle\langle \gamma_2, \nu_2+\delta\rangle \dots \langle \gamma_n, \nu_n+\delta\rangle.$$
2. The unboundedness of real numbers.
3. The denseness of real numbers.

Removing Entire Stack Modification: Lazy Semantics

To simulate the entire stack modification of timed transitions by pushdown systems that only allow to modify finitely (boundedly) many elements of a stack, we use the technique called *lazy time elapsing* that was developed by Abdulla et al. in [AAS12a, AAS12b].

To show the idea of the technique, let us consider the following transitions:

$$\nu_1\,\nu_2\,\nu_3 \overset{2.0}{\rightsquigarrow} \nu_1'\,\nu_2'\,\nu_3' \xrightarrow{\mathsf{dig}(x,x)} \nu_1'\,\nu_2'\,\nu_3'' \xrightarrow{\mathsf{pop}} \nu_1'\,\nu_3''$$

where

$$\begin{aligned} &\nu_1 = \{x \mapsto 0.5\}, \nu_2 = \{x \mapsto 2.0\}, \nu_3 = \{x \mapsto 1.5\}, \\ &\nu_i' = \nu_i + 2.0,\ \nu_3'' = \{x \mapsto 4.0\}. \end{aligned}$$

Our simulation idea is to

- keep the correct top and next to the top frames rather than the entire stack for the **check** and **dig** rules; and
- reconstruct a new frame that reflects the correct information when performing **pop** transitions.

We introduce the notion of clock marking and valuation pairing to describe this idea. We pair two valuations ν_i and ν_{i+1} and make the paired valuation μ_{i+1} as follows:

1. Mark ν_{i+1} and obtain the marked valuation $\dot{\nu}_{i+1} : \dot{\mathcal{X}} \to \mathbb{R}_{\geq 0}$ as $\dot{\nu}_{i+1}(\dot{x}) \triangleq \nu_{i+1}(x)$.
2. In the same way, we mark and obtain the marked valuation $\underset{\bullet}{\nu}_i : \underset{\bullet}{\mathcal{X}} \to \mathbb{R}_{\geq 0}$.
3. Finally, we obtain $\mu : \underset{\bullet}{\mathcal{X}} \cup \dot{\mathcal{X}} \to \mathbb{R}_{\geq 0}$ by gluing them $\mu_{i+1} \triangleq \underset{\bullet}{\nu}_i \cup \dot{\nu}_{i+1}$.

We relate the 1-height stack ν_1 to a 1-height stack μ_1 of a paired valuation such that $\nu_1(x) = \mu_1(\dot{x})$ as follows:

$$\nu_1 = \{x \mapsto 0.5\} \models \{\underset{\bullet}{x} \mapsto r;\ \dot{x} \mapsto 0.5\},$$

where the value of $\underset{\bullet}{x}$ is irrelevant. We also relate the 3-height stack $\nu_1\nu_2\nu_3$ to the following 3-height stack $\mu_1\mu_2\mu_3$:

$$\begin{array}{c} \nu_3 = \{x \mapsto 1.5\} \\ \hline \nu_2 = \{x \mapsto 2.0\} \\ \hline \nu_1 = \{x \mapsto 0.5\} \end{array} \models \begin{array}{c} \mu_3 = \{\underset{\bullet}{x} \mapsto 2.0;\ \dot{x} \mapsto 1.5\} \\ \hline \mu_2 = \{\underset{\bullet}{x} \mapsto 0.5;\ \dot{x} \mapsto 2.0\} \\ \hline \mu_1 = \{\underset{\bullet}{x} \mapsto \quad r;\ \dot{x} \mapsto 0.5\} \end{array},$$

where the top frame μ_3 has the correct information of the top frame ν_3 and next to the top frame ν_2, the frame μ_2 corresponds to the frames ν_2 and ν_1, and the frame μ_1 corresponds to the frame ν_1.

On the basis of valuation pairing, we simulate the entire stack modification $\nu_1\nu_2\nu_3 \overset{2.0}{\rightsquigarrow} \nu'_1\nu'_2\nu'_3$ by only evolving the top frame as follows ($\mu_1\mu_2\mu_3 \overset{2.0}{\rightsquigarrow} \mu_1\mu_2\mu'_3$):

$$\begin{array}{c} \{\underset{\bullet}{x} \mapsto 2.0;\ \dot{x} \mapsto 1.5\} \\ \hline \{\underset{\bullet}{x} \mapsto 0.5;\ \dot{x} \mapsto 2.0\} \\ \hline \{\underset{\bullet}{x} \mapsto \quad r;\ \dot{x} \mapsto 0.5\} \end{array} \overset{2.0}{\rightsquigarrow} \begin{array}{c} \{\underset{\bullet}{x} \mapsto 4.0;\ \dot{x} \mapsto 3.5\} \\ \hline \{\underset{\bullet}{x} \mapsto 0.5;\ \dot{x} \mapsto 2.0\} \\ \hline \{\underset{\bullet}{x} \mapsto \quad r;\ \dot{x} \mapsto 0.5\} \end{array}.$$

Although the obtained stack does not match the stack $\nu'_1\nu'_2\nu'_3$, the top frame μ'_3 matches the top frame ν'_3 and the next to the top frame ν'_2. Therefore, we can safely simulate the **dig** transition $\nu'_1\nu'_2\nu'_3 \xrightarrow{\mathsf{dig}(x,x)} \nu'_1\nu'_2\nu''_3$ as follows ($\mu_1\mu_2\mu'_3 \xrightarrow{\dot{x}:=\underset{\bullet}{x}} \mu_1\mu_2\mu''_3$):

$$\begin{array}{c} \{\underset{\bullet}{x} \mapsto 4.0;\ \dot{x} \mapsto 3.5\} \\ \hline \{\underset{\bullet}{x} \mapsto 0.5;\ \dot{x} \mapsto 2.0\} \\ \hline \{\underset{\bullet}{x} \mapsto \quad r;\ \dot{x} \mapsto 0.5\} \end{array} \xrightarrow{\dot{x}:=\underset{\bullet}{x}} \begin{array}{c} \{\underset{\bullet}{x} \mapsto 4.0;\ \dot{x} \mapsto 4.0\} \\ \hline \{\underset{\bullet}{x} \mapsto 0.5;\ \dot{x} \mapsto 2.0\} \\ \hline \{\underset{\bullet}{x} \mapsto \quad r;\ \dot{x} \mapsto 0.5\} \end{array}.$$

In order to simulate $\nu'_1\nu'_2\nu''_3 \xrightarrow{\mathsf{pop}} \nu'_1\nu''_3$, we evolve the next to the top frame μ_2 until $\mu_2 + \delta$ matches μ_3 or formally $(\mu_2 + \delta)(\dot{x}) = \mu_3(\underset{\bullet}{x})$ holds:

$$\begin{array}{c} \{\underline{\underset{\bullet}{x} \mapsto 4.0};\ \dot{x} \mapsto 4.0\} \\ \hline \{\underset{\bullet}{x} \mapsto 0.5;\ \overline{\dot{x} \mapsto 2.0}\} \\ \hline \{\underset{\bullet}{x} \mapsto \quad r;\ \dot{x} \mapsto 0.5\} \end{array} \to \begin{array}{c} \{\underline{\underset{\bullet}{x} \mapsto 4.0};\ \dot{x} \mapsto 4.0\} \\ \hline \{\underset{\bullet}{x} \mapsto 2.5;\ \overline{\dot{x} \mapsto 4.0}\} \\ \hline \{\underset{\bullet}{x} \mapsto \quad r;\ \dot{x} \mapsto 0.5\} \end{array}.$$

This operation reconstructs the frame $\{\underset{\bullet}{x} \mapsto 2.5; \dot{x} \mapsto 4.0\}$ that has the correct information of ν'_1 and ν'_2. We compose the frames μ''_3 and $\mu_2 + 2.0$ as follows:

$$\begin{array}{c} \{\underline{\underset{\bullet}{x} \mapsto 4.0};\ \dot{x} \mapsto 4.0\} \\ \hline \{\underset{\bullet}{x} \mapsto 2.5;\ \overline{\dot{x} \mapsto 4.0}\} \\ \hline \{\underset{\bullet}{x} \mapsto \quad r;\ \dot{x} \mapsto 0.5\}. \end{array} \to \begin{array}{c} \{\underset{\bullet}{x} \mapsto 2.5;\ \dot{x} \mapsto 4.0\} \\ \hline \{\underset{\bullet}{x} \mapsto \quad r;\ \dot{x} \mapsto 0.5\}. \end{array}$$

After simulating the **pop** transition, the top frame $\{\underset{\bullet}{x} \mapsto 2.5;\ \overset{\bullet}{x} \mapsto 4.0\}$ matches $\nu'_1\nu''_3$.

It is worth noting that, to simulate **pop** transitions, we need to operate the top and the next to the top frames at once. For this purpose, multiple pop transition rules $\langle q, \alpha\rangle \hookrightarrow \langle q, \beta\rangle$ where $\alpha \in \Gamma^+$ are allowed in our formalization of pushdown systems.

Considering stacks of paired valuations is enough to simulate the standard semantics by an infinite pushdown system. However, for technical reasons, we introduce *reference clocks* along with the lazy elapsing technique. We use a fresh clock called a *reference clock* $\mathtt{C}$. Such a clock is accessed and reset to 0.0 only when we push a new frame as follows:

$$\{\underset{\bullet}{\mathtt{C}} \mapsto r_1;\ \underset{\bullet}{x} \mapsto 0.5;\ \overset{\bullet}{\mathtt{C}} \mapsto r_2;\ \overset{\bullet}{x} \mapsto 2.0\} \xrightarrow{\textsf{push}} \frac{\{\underset{\bullet}{\mathtt{C}} \mapsto 0.0;\ \underset{\bullet}{x} \mapsto 2.0;\ \overset{\bullet}{\mathtt{C}} \mapsto 0.0;\ \overset{\bullet}{x} \mapsto 0.0\}}{\{\underset{\bullet}{\mathtt{C}} \mapsto r_1;\ \underset{\bullet}{x} \mapsto 0.5;\ \overset{\bullet}{\mathtt{C}} \mapsto 0.0;\ \overset{\bullet}{x} \mapsto 2.0\}}\,.$$

The introduction of reference clocks does not interfere with the above idea of the lazy elapsing technique. The reader can find the reason why we need reference clocks in Section 5.5.2, the remark after Lemma 5.7, and Section 5.6.5.

In Section 5.4, we show the simulation between the standard semantics and the lazy semantics in the both directions:

$$\textbf{Lemma 5.3}: \begin{tikzcd} \boldsymbol{s} \arrow[r, "\textsc{Stnd}"'] \arrow[d, phantom, "\Top"] & \boldsymbol{s'} \\ \boldsymbol{l} & \end{tikzcd} \Longrightarrow \begin{tikzcd} \boldsymbol{s} \arrow[r, "\textsc{Stnd}"'] \arrow[d, phantom, "\Top"] & \boldsymbol{s'} \arrow[d, phantom, "\Top"] \\ \boldsymbol{l} \arrow[r, "\textsc{Lazy}"'] & \exists \boldsymbol{l'}, \end{tikzcd}$$

$$\textbf{Lemma 5.4}: \begin{tikzcd} \boldsymbol{s} \arrow[d, phantom, "\Top"] & \\ \boldsymbol{l} \arrow[r, "\textsc{Lazy}"'] & \boldsymbol{l'} \end{tikzcd} \Longrightarrow \begin{tikzcd} \boldsymbol{s} \arrow[r, "\textsc{Stnd}"'] \arrow[d, phantom, "\Top"] & \exists \boldsymbol{s'} \arrow[d, phantom, "\Top"] \\ \boldsymbol{l} \arrow[r, "\textsc{Lazy}"'] & \boldsymbol{l'}. \end{tikzcd}$$

Removing the Unboundedness of Real Numbers: Collapsed Semantics Coll

To remove the unboundedness of real numbers, as with Section 2.4.3 of Chapter 2, we consider the collapsed valuations. For an SRTA $\mathcal{A}$, we introduce an upper-bound constant M:

$$\mathsf{M} \triangleq \max\{\ i, j : (i : j) \text{ or } [i : j] \text{ appears in an interval constraint of } \mathcal{A}\ \} + 1.$$

We also define the collapsed real numbers $\mathbb{C}$ as follows:

$$\mathbb{C} \triangleq \big([0..(\mathsf{M}-1)] \cup \{\infty\}\big) \times [0, 1)$$

and define the collapsing function $\mathcal{C} : \mathbb{R}_{\geq 0} \to \mathbb{C}$ as follows:

$$\mathcal{C}(r) \ \triangleq\ \begin{cases} (\infty\ , \mathit{frac}(r)) & \text{if } r \geq \mathsf{M} \\ (\lfloor r \rfloor, \mathit{frac}(r)) & \text{if } r < \mathsf{M} \end{cases}$$

In Section 5.5, we will show the simulation between the lazy and collapsed semantics in the both directions:

$$\textbf{Lemma 5.6}: \begin{tikzcd} \boldsymbol{l} \arrow[r, "\textsc{Lazy}"'] \arrow[d, phantom, "\Top"] & \boldsymbol{l'} \\ \boldsymbol{c} & \end{tikzcd} \Longrightarrow \begin{tikzcd} \boldsymbol{l} \arrow[r, "\textsc{Lazy}"'] \arrow[d, phantom, "\Top"] & \boldsymbol{l'} \arrow[d, phantom, "\Top"] \\ \boldsymbol{c} \arrow[r, "\textsc{Coll}"'] & \exists \boldsymbol{c'}, \end{tikzcd}$$

$$\textbf{Lemma 5.7}: \begin{tikzcd} \boldsymbol{l} \arrow[d, phantom, "\Top"] & \\ \boldsymbol{c} \arrow[r, "\textsc{Coll}"'] & \boldsymbol{c'} \end{tikzcd} \Longrightarrow \begin{tikzcd} \boldsymbol{l} \arrow[r, "\textsc{Lazy}"'] \arrow[d, phantom, "\Top"] & \exists \boldsymbol{l'} \arrow[d, phantom, "\Top"] \\ \boldsymbol{c} \arrow[r, "\textsc{Coll}"'] & \boldsymbol{c'}. \end{tikzcd}$$

Removing the Denseness of Real Numbers: Digitized Semantics Digi.

Finally, we remove the denseness of real numbers from the collapsed domain $\mathbb{C}$ by introducing digital valuations as with Section 2.3.1 of Chapter 2. Recall a digital valuation is obtained from a collapsed valuation by abstracting the fractional parts of the valuation into the corresponding ordering. For example, we abstract the following collapsed valuation

$$\mu = \{a \mapsto 1.0;\ b \mapsto 2.3;\ c \mapsto 3.7;\ d \mapsto \infty.3\}$$

into the digital valuation

$$\boldsymbol{d} = \{(a,1)\}_0\, \{(b,2),(d,\infty)\}\, \{(c,3)\} \quad (\mu \models \boldsymbol{d})$$

and $\boldsymbol{d}$ means the following:

- The term $\{(a,1)\}_0$ means that the integral part of a is 1 and the fractional part of a is 0.0 (i.e., the value of a is 1.0).
- The term $\{(b,2),(d,\infty)\}$ means that the fractional parts of b and d are the same. Furthermore, the fractional parts of b and d are strictly larger than 0.0 because they do not belong to $\{\ldots\}_0$.
- The order $\{(b,2),(d,\infty)\}\,\{(c,3)\}$ means that the fractional part of b and d is strictly smaller than that of c.

In the theory of timed automata (without the stack), the region is an appropriate abstraction of the collapsed valuations. Indeed, when considering timed automata rather than SRTA, we have the forward simulation between the collapsed and digitized semantics in the both directions as follows:

For TA:
$$\begin{tikzcd} \boldsymbol{c} \arrow[d, phantom, "\rotatebox{-90}{$\models$}"] & \\ \boldsymbol{d} \arrow[r, "\textsc{Digi}"'] & \boldsymbol{d'} \end{tikzcd} \Longrightarrow \begin{tikzcd} \boldsymbol{c} \arrow[r, "\textsc{Coll}"'] \arrow[d, phantom, "\rotatebox{-90}{$\models$}"] & \exists \boldsymbol{c'} \arrow[d, phantom, "\rotatebox{-90}{$\models$}"] \\ \boldsymbol{d} \arrow[r, "\textsc{Digi}"'] & \boldsymbol{d'}. \end{tikzcd}$$

Lemma 5.9:
$$\begin{tikzcd} \boldsymbol{c} \arrow[r, "\textsc{Coll}"'] \arrow[d, phantom, "\rotatebox{-90}{$\models$}"] & \boldsymbol{c'} \\ \boldsymbol{d} & \end{tikzcd} \Longrightarrow \begin{tikzcd} \boldsymbol{c} \arrow[r, "\textsc{Coll}"'] \arrow[d, phantom, "\rotatebox{-90}{$\models$}"] & \boldsymbol{c'} \arrow[d, phantom, "\rotatebox{-90}{$\models$}"] \\ \boldsymbol{d} \arrow[r, "\textsc{Digi}"'] & \exists \boldsymbol{d'}. \end{tikzcd}$$

However, on SRTA, the former diagram does not hold for the unavoidable nondeterminacy of **pop** transitions. For example, let us consider the following **pop**-transition:

$$\begin{tikzcd} \dfrac{\{\underset{\bullet}{x} \mapsto 1.0;\ \dot{x} \mapsto 2.2\}}{\{\dot{x} \mapsto 1.0;\ \underset{\bullet}{x} \mapsto 0.4\}} \arrow[r, "\textbf{pop}", "\textsc{Coll}"'] \arrow[d, phantom, "\rotatebox{-90}{$\models$}"] & \{\underset{\bullet}{x} \mapsto 0.4;\ \dot{x} \mapsto 2.2\} \\ \dfrac{\boldsymbol{d}_2 = \{(\underset{\bullet}{x},1)\}_0\{(\dot{x},2)\}}{\boldsymbol{d}_1 = \{(\dot{x},1)\}_0\{(\underset{\bullet}{x},0)\}} \arrow[r, "\textbf{pop}", "\textsc{Digi}"'] & \{\boldsymbol{d},\boldsymbol{d'},\boldsymbol{d''}\}\,. \end{tikzcd}$$

We need to decide the order of $\underset{\bullet}{x} \in \boldsymbol{d}_1$ and $\dot{x} \in \boldsymbol{d}_2$ when composing $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$. However, since there is no information about their fractional parts, we generate all the possibilities as follows:

$$\begin{aligned} \boldsymbol{d} &= \{\}_0\, \{(\underset{\bullet}{x},0),(\dot{x},2)\}\,, \\ \boldsymbol{d'} &= \{\}_0\, \{(\underset{\bullet}{x},0)\}\, \{(\dot{x},2)\}\,, \\ \boldsymbol{d''} &= \{\}_0\, \{(\dot{x},2)\}\, \{(\underset{\bullet}{x},0)\}\,. \end{aligned}$$

On the other hand, the **pop** transition of the collapsed semantics behaves deterministically; therefore, the collapsed semantics cannot capture the nondeterministic behavior of the **pop** transition of the digitized semantics.

Although we give up the forward simulation, the following *backward* simulation holds:

Lemma 5.10 :
$$\begin{tikzcd} & \boldsymbol{c'} \arrow[d, phantom, "\rotatebox{-90}{$\models$}"] \\ \boldsymbol{d} \arrow[r, "\textsc{Digi}"'] & \boldsymbol{d'} \end{tikzcd} \Longrightarrow \begin{tikzcd} \exists \boldsymbol{c} \arrow[r, "\textsc{Coll}"'] \arrow[d, phantom, "\rotatebox{-90}{$\models$}"] & \boldsymbol{c'} \arrow[d, phantom, "\rotatebox{-90}{$\models$}"] \\ \boldsymbol{d} \arrow[r, "\textsc{Digi}"'] & \boldsymbol{d'}. \end{tikzcd}$$

The backward simulation naturally solves the above determinacy vs. nondeterminacy problem. Furthermore, as we will see below, the backward simulation is key to establishing the decidability of the configuration reachability problem.

5.3.2 Backward Simulation in Configuration Reachability Problem

We see how to use the backward simulation lemma (Lemma 5.10) in the configuration reachability problem. Let us consider the following configuration reachability problem:

$$\langle q_{\mathrm{init}}, \langle \bot, \{x \mapsto 0.0;\ y \mapsto 0.0\}\rangle\rangle \xrightarrow[\textsc{Stnd}]{}{}^{*} \langle q, \langle \bot, \{x \mapsto 2.71;\ y \mapsto 3.14\}\rangle\rangle. \tag{$\star$}$$

Hereafter we confirm that the above problem is equivalent to find a digital valuation $\boldsymbol{d}$ that satisfies $\{x \mapsto 2.71; y \mapsto 3.14\} \models \boldsymbol{d}$ and the following:

$$\langle q_{\mathrm{init}}, \langle \bot, \{(\underset{\bullet}{x}, 0), (\underset{\bullet}{y}, 0), (\dot{x}, 0), (\dot{y}, 0)\}_0\rangle\rangle \xrightarrow[\textsc{Digi}]{}{}^{*} \langle q, \langle \bot, \boldsymbol{d}\rangle\rangle. \tag{$\sharp$}$$

For simplicity, we do not consider reference clocks in this example and assume $\mathsf{M} \geq 4$.

Let us assume the following digital valuation $\boldsymbol{d}$ satisfies ($\sharp$):

$$\boldsymbol{d} = \{\}_0\, \{(\dot{y}, 3)\}\, \{(\dot{x}, 2)\}\, \{(\underset{\bullet}{x}, \infty), (\underset{\bullet}{y}, \infty)\}.$$

We consider the following collapsed valuation μ such that $\mu \models \boldsymbol{d}$:

$$\mu = \left\{\begin{array}{c} \dot{x} \mapsto 2.71;\ \dot{y} \mapsto 3.14; \\ \hdashline \underset{\bullet}{x} \mapsto \infty.9;\ \underset{\bullet}{y} \mapsto \infty.9 \end{array}\right\} \models \boldsymbol{d}.$$

The backward simulation lemma, Lemma 5.10, ensures the following run:

$$\langle q_{\mathrm{init}}, \langle \bot, \left\{\begin{array}{c} \dot{x} \mapsto 0.0;\ \dot{y} \mapsto 0.0; \\ \hdashline \underset{\bullet}{x} \mapsto 0.0;\ \underset{\bullet}{y} \mapsto 0.0 \end{array}\right\}\rangle\rangle \xrightarrow[\textsc{Coll}]{}{}^{*} \langle q, \langle \bot, \mu\rangle\rangle.$$

Sequentially applying the forward simulation lemmas, Lemma 5.7 and 5.4, to this run, we obtain the above run ($\star$). Therefore, to solve ($\star$), it suffices to find a digital valuation $\boldsymbol{d}$ such that $\{x \mapsto 2.71; y \mapsto 3.14\} \models \boldsymbol{d}$ and solve ($\sharp$). Indeed, we will use the same argument in the proof of our main theorem (Theorem 5.4).

In the above argument, using the backward simulation is key and we cannot replace it by the forward simulation of the digitized semantics by the collapsed semantics. (As we have seen above, we cannot forwardly simulate the semantics Digi by the semantics Coll due to the nondeterminacy of **pop**-transitions.) Even if we could apply the forward simulation to the run ($\sharp$), then we may obtain the following run:

$$\langle q_{\mathrm{init}}, \langle \bot, \left\{\begin{array}{c} \dot{x} \mapsto 0.0;\ \dot{y} \mapsto 0.0; \\ \hdashline \underset{\bullet}{x} \mapsto 0.0;\ \underset{\bullet}{y} \mapsto 0.0 \end{array}\right\}\rangle\rangle \xrightarrow[\textsc{Coll}]{}{}^{*} \langle q, \langle \bot, \mu' = \left\{\begin{array}{c} \dot{x} \mapsto 2.34;\ \dot{y} \mapsto 3.09; \\ \hdashline \underset{\bullet}{x} \mapsto \infty.9;\ \underset{\bullet}{y} \mapsto \infty.9 \end{array}\right\}\rangle\rangle$$

where $\mu' \models \boldsymbol{d}$. Since the forward simulation only ensures a run to $\langle q, \langle \bot, \mu'\rangle\rangle$ where $\mu' \models \boldsymbol{d}$, we cannot show the existence of a run to $\langle q, \langle \bot, \mu\rangle\rangle$ where $\mu(\dot{x}) = 2.71$ and $\mu(\dot{y}) = 3.14$. If we apply Lemma 5.7 and 5.8 to this run. then we only obtain the following run that differs from ($\star$):

$$\langle q_{\mathrm{init}}, \langle \bot, \{x \mapsto 0.0;\ y \mapsto 0.0\}\rangle\rangle \xrightarrow[\textsc{Stnd}]{}{}^{*} \langle q, \langle \bot, \{x \mapsto 2.34;\ y \mapsto 3.09\}\rangle\rangle.$$

5.3.3 Comparing Proof of Abdulla et al. and Ours

We review the proof of *Lemma 4* of Abdulla et al. in [AAS12a], which enables us to reduce the location reachability problem of the standard semantics to the location reachability problem of the digitized semantics. The proof structure of their lemma can be summarized schematically as the following diagram by using our notation:

$$\begin{tikzcd}
\boldsymbol{W} \arrow[r, "\textsc{Digi}"] \arrow[d, phantom, "\wr"] & \boldsymbol{W'} \arrow[d, phantom, "\wr"] \\
{}^{\exists}C \arrow[d, phantom, "\cong"] & {}^{\forall}C' \arrow[l, dashed] \arrow[d, phantom, "\cong"] \\
{}^{\forall}w \arrow[r, "\textsc{Stnd}"'] \arrow[r, dashed, bend left] & {}^{\exists}w'
\end{tikzcd}.$$

This diagram says that if $\boldsymbol{W} \xrightarrow[\textsc{Digi}]{}^{*} \boldsymbol{W'}$ and $C' \approx \boldsymbol{W'}$, then there is C such that for all stack $w \cong C$ there exists $w' \cong C'$ such that $w \xrightarrow[\textsc{Stnd}]{}^{*} w'$. Let us explain the definition of the relation $\approx$ and $\cong$. Informally, $C \approx \boldsymbol{W}$ means that a valuation C is obtained by flattening a stack $\boldsymbol{W}$. Let us consider the following stack:

$$\boldsymbol{W} = \begin{array}{c} \boldsymbol{d}_3 = \{(y,1)\}_0 \{(x,2)\} \\ \hline \boldsymbol{d}_2 = \{\}_0 \{(x,3)(y,4)\} \\ \hline \boldsymbol{d}_1 = \{\}_0 \{(x,1)\} \{(y,5)\} \end{array}.$$

The following is one of the valuations obtained by flattening $\boldsymbol{W}$:

$$C = \left\{ \begin{array}{l} y^{(3)} \mapsto 1.0;\ x^{(1)} \mapsto 1.1;\ x^{(3)} \mapsto 2.4; \\ x^{(2)} \mapsto 3.6;\ y^{(2)} \mapsto 4.6;\ y^{(1)} \mapsto 5.9 \end{array} \right\}.$$

The tag of each clock $x^{(i)}$ or $y^{(j)}$ points to the frame where the clock comes from: for example, the clock $y^{(3)}$ and $x^{(1)}$ comes from the digital valuation $\boldsymbol{d}_3$ and $\boldsymbol{d}_1$ of $\boldsymbol{W}$, respectively. Informally, $w \cong C$ means that a stack w matches a valuation C or w is isomorphic to C. For the above flatten valuation C, there exists the unique stack w that matches C:

$$w = \begin{array}{c} \nu_3 = \{y \mapsto 1.0;\ x \mapsto 2.4\} \\ \hline \nu_2 = \{x \mapsto 3.6;\ y \mapsto 4.6\} \\ \hline \nu_1 = \{x \mapsto 1.1;\ y \mapsto 5.9\} \end{array}$$

because we can reconstruct the stack w from the tag information of C.

They directly bridged the two semantics, the standard semantics and digitized semantics. As a result, their simulation requires an elaborated form; we find out that their elaborate simulation is called a *backward-forward simulation* in Lynch and Vaandrager [LV95, LV96]. It is a source of complications in their proof to simultaneously handle the backward direction (choosing C from C') and the forward direction (finding w' from $w \in C$). In addition, the relation $\approx$ is not defined by a componentwise manner and it is another source of their elaborated proof.

In contrast, we clearly solve these problems as Lemmas 5.4, 5.7, and 5.10 by considering the intermediate semantics Lazy and Coll.

$$\begin{tikzcd}
\boldsymbol{W} \arrow[r, "\textsc{Digi}"] \arrow[d, phantom, "\models"] & \boldsymbol{W'} \arrow[d, phantom, "\models"] \\
{}^{\exists}\boldsymbol{w} \arrow[r, "\textsc{Coll}"'] \arrow[d, phantom, "\models"] & {}^{\forall}\boldsymbol{w'} \arrow[l, dashed, bend right, "\text{Lem}\,5.10"'] \arrow[d, phantom, "\models"] \\
{}^{\forall}\omega \arrow[r, "\textsc{Lazy}"'] \arrow[r, dashed, bend left, "\text{Lem}\,5.7"] \arrow[d, phantom, "\models"] & {}^{\exists}\omega' \arrow[d, phantom, "\models"] \\
{}^{\forall}w \arrow[r, "\textsc{Stnd}"'] \arrow[r, dashed, bend left, "\text{Lem}\,5.4"] & {}^{\exists}w'
\end{tikzcd}$$

This allows us to separate the above mixed simulation into three simple simulations and define correspondences $\models$ in a componentwise manner. Finally, these make the entire proof structure easy to understand.

5.4 Lazy Semantics: Removing Entire Stack Modification

We define notations to formalize the lazy time elapsing technique.

Definition 5.3 (Clock Marking)**.** Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ be a finite clock set. We use $\dot{\mathcal{X}}$ to denote the marked set $\{\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_n\}$ and $\underset{\cdot}{\mathcal{X}}$ to denote the marked set $\{\underset{\cdot}{x}_1, \underset{\cdot}{x}_2, \ldots, \underset{\cdot}{x}_n\}$.

Let $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be a valuation on $\mathcal{X}$. We write $\dot{\nu}$ to denote the marked valuation $\dot{\nu} : \dot{\mathcal{X}} \to \mathbb{R}_{\geq 0}$ defined by $\dot{\nu}(\dot{x}) \triangleq \nu(x)$. We also write $\underset{\cdot}{\nu}$ for the marked valuation on $\underset{\cdot}{\mathcal{X}}$.

For a constraint φ on $\mathcal{X}$, we use $\dot{\varphi}$ to denote the corresponding marked constraint on $\dot{\mathcal{X}}$. For example, if $\varphi = \big(\mathit{frac}(x) = 0 \land \mathit{frac}(x) < \mathit{frac}(y)\big)$, then $\dot{\varphi} = \big(\mathit{frac}(\dot{x}) = 0 \land \mathit{frac}(\dot{x}) < \mathit{frac}(\dot{y})\big)$. ■

Definition 5.4 (Compatibility and Composition)**.** Let μ_1 and μ_2 be valuations on $\underset{\cdot}{\mathcal{X}} \cup \dot{\mathcal{X}}$.

The valuation μ_1 is *compatible* with the valuation μ_2 if $\mu_1(\dot{x}) = \mu_2(\underset{\cdot}{x})$ for all $x \in \mathcal{X}$ and we write $\mu_1 \mathbin{/\!/} \mu_2$.

If a valuation μ_1 is compatible with a valuation μ_2, the composed valuation $\mu_1 \odot \mu_2$ is defined as follows:

$$(\mu_1 \odot \mu_2)(\underset{\cdot}{x}) = \mu_1(\underset{\cdot}{x}), \quad (\mu_1 \odot \mu_2)(\dot{x}) = \mu_2(\dot{x}).$$

■

As we have explained in the previous section, we need a reference clock to justify a simulation between the collapsed and digitized semantics.

Definition 5.5. Let $\mathcal{X}$ be a finite clock set. We write $\mathcal{X}_{\mathtt{C}}$ to denote the clock set $\mathcal{X} \cup \{\mathtt{C}\}$ extended by a reference clock $\mathtt{C}$.

Let ν_1 and ν_2 be valuations on $\mathcal{X}$ and μ be a valuation on $\underset{\cdot}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}}$. If $\big(\underset{\cdot}{\nu}_1 \cup \dot{\nu}_2\big) = \mu \restriction (\underset{\cdot}{\mathcal{X}} \cup \dot{\mathcal{X}})$, then we write $\langle \nu_1, \nu_2 \rangle \models \mu$. ■

For example, let us simulate the following transitions with our notations:

$$\nu_1 \xrightarrow{\textsf{push}} \nu_1\nu_2 \overset{2.0}{\rightsquigarrow} \nu_1'\nu_2' \xrightarrow{\textsf{pop}} \nu_2'$$

where

$$\nu_1 = \{x \mapsto 1.5\},\ \nu_2 = \{x \mapsto 0.0\},\ \nu_i' = \nu_i + 2.$$

We start from the following paired valuation μ_1:

$$\mu_1 = \left\{\begin{array}{c} \dot{x} \mapsto 1.5;\ \dot{\mathtt{C}} \mapsto r_2 \\ \hdashline \underset{\cdot}{x} \mapsto r_x;\ \underset{\cdot}{\mathtt{C}} \mapsto r_1 \end{array}\right\}.$$

A 1-height stack μ corresponds to a 1-height stack ν if $\dot{\nu} = \mu \restriction \dot{\mathcal{X}}$ holds. Therefore, the above 1-height stack μ_1 corresponds to the 1-height stack ν_1.

We simulate the first transition $\nu_1 \xrightarrow{\textsf{push}} \nu_1\nu_2$ as follows ($\mu_1 \to \mu_1'\mu_2$):

$$\left\{\begin{array}{c} \dot{x} \mapsto 1.5;\ \dot{\mathtt{C}} \mapsto r_2 \\ \hdashline \underset{\cdot}{x} \mapsto r_x;\ \underset{\cdot}{\mathtt{C}} \mapsto r_1 \end{array}\right\} \to \frac{\left\{\begin{array}{c} \dot{x} \mapsto 0.0;\ \dot{\mathtt{C}} \mapsto 0.0 \\ \hdashline \underset{\cdot}{x} \mapsto 1.5;\ \underset{\cdot}{\mathtt{C}} \mapsto 0.0 \end{array}\right\}}{\left\{\begin{array}{c} \dot{x} \mapsto 1.5;\ \dot{\mathtt{C}} \mapsto 0.0 \\ \hdashline \underset{\cdot}{x} \mapsto r_x;\ \underset{\cdot}{\mathtt{C}} \mapsto r_1 \end{array}\right\}}.$$

When pushing a new frame, we reset the clock $\dot{\mathtt{C}}$ of the current top frame to 0.0. This resetting is important to establish the backward simulation lemma, Lemma 5.10, in Section 5.6.5. We assign the values of the marked clocks $\dot{y}$ of the current top frame to the corresponding marked clocks $\underset{\cdot}{y}$ of the new frame to be pushed. As the result, the new top frame μ_2 reflects the information of the top ν_2 and next to the top ν_1 frames in the standard semantics.

$$\frac{\mu_1' = \mu_1[\dot{\mathtt{C}} := 0] \quad \mu_1' \mathbin{//} \mu_2 \quad \mu_2(\dot{x}) = 0 \ (\forall x \in \mathcal{X}_{\underset{\bullet}{\mathtt{C}}})}{\langle \gamma_1, \mu_1 \rangle \xhookrightarrow{\mathsf{push}(\gamma_2)} \langle \gamma_1, \mu_1' \rangle \langle \gamma_2, \mu_2 \rangle}\ \mathsf{push}(\gamma_2)$$

$$\frac{\mu_1 \leq \mu_1' \quad \mu_1' \mathbin{//} \mu_2}{\langle \gamma_1, \mu_1 \rangle\, \langle \gamma_2, \mu_2 \rangle \xhookrightarrow{\mathsf{pop}(\gamma_2)} \langle \gamma_1, \mu_1' \odot \mu_2 \rangle}\ \mathsf{pop}(\gamma_2) \qquad \frac{\mu \models \dot{\varphi}}{\langle \gamma, \mu \rangle \xhookrightarrow{\mathsf{check}(\varphi)} \langle \gamma, \mu \rangle}\ \mathsf{check}(\varphi)$$

$$\frac{\mu_2' = \mu_2[\dot{x} := \underset{\bullet}{y}]}{\langle \gamma_1, \mu_1 \rangle \langle \gamma_2, \mu_2 \rangle \xhookrightarrow{\mathsf{dig}(x,y)} \langle \gamma_1, \mu_1 \rangle \langle \gamma_2, \mu_2' \rangle}\ \mathsf{dig}(x, y) \qquad \frac{r \in I \quad \mu' = \mu[\dot{x} := r]}{\langle \gamma, \mu \rangle \xhookrightarrow{x \leftarrow I} \langle \gamma, \mu' \rangle}\ x \leftarrow I$$

Figure 5.1: Definition of actions on Lazy semantics

We simulate the second transition $\nu_1\nu_2 \overset{2.0}{\rightsquigarrow} \nu_1'\nu_2'$ as follows ($\mu_1'\mu_2 \to \mu_1'\mu_2'$):

$$\frac{\left\{\begin{array}{c} \dot{x} \mapsto 0.0;\ \dot{\mathtt{C}} \mapsto 0.0 \\ \hdashline \underset{\bullet}{x} \mapsto 1.5;\ \underset{\bullet}{\mathtt{C}} \mapsto 0.0 \end{array}\right\}}{\left\{\begin{array}{c} \dot{x} \mapsto 1.5;\ \dot{\mathtt{C}} \mapsto 0.0 \\ \hdashline \underset{\bullet}{x} \mapsto r_x;\ \underset{\bullet}{\mathtt{C}} \mapsto r_1 \end{array}\right\}} \to \frac{\left\{\begin{array}{c} \dot{x} \mapsto 2.0;\ \dot{\mathtt{C}} \mapsto 2.0 \\ \hdashline \underset{\bullet}{x} \mapsto 3.5;\ \underset{\bullet}{\mathtt{C}} \mapsto 2.0 \end{array}\right\}}{\left\{\begin{array}{c} \dot{x} \mapsto 1.5;\ \dot{\mathtt{C}} \mapsto 0.0 \\ \hdashline \underset{\bullet}{x} \mapsto r_x;\ \underset{\bullet}{\mathtt{C}} \mapsto r_1 \end{array}\right\}}.$$

When performing timed transitions, we only modify the top frame in a stack as above.

We simulate the last transition $\nu_1'\nu_2' \xrightarrow{\mathsf{pop}} \nu_2'$. First, we adjust μ_1' to be matched with the real frame ν_1', and it is formalized by evolving μ_1' until $\mu_1' + \delta \mathbin{//} \mu_2'$. For this case, $\delta = 2.0$ and this corresponds to adding the time (2.0) elapsed after μ_1' was covered by the new top frame μ_2. Next, we compose the two valuations $\mu_1' + 2.0$ and μ_2' as follows:

$$\frac{\left\{\begin{array}{c} \dot{x} \mapsto 2.0;\ \dot{\mathtt{C}} \mapsto 2.0; \\ \hdashline \underset{\bullet}{x} \mapsto 3.5;\ \underset{\bullet}{\mathtt{C}} \mapsto 2.0 \end{array}\right\}}{\left\{\begin{array}{c} \dot{x} \mapsto 3.5;\ \dot{\mathtt{C}} \mapsto 2.0; \\ \hdashline \underset{\bullet}{x} \mapsto r_x + 2.0;\ \underset{\bullet}{\mathtt{C}} \mapsto r_1 + 2.0 \end{array}\right\}} \overset{\odot}{\Rightarrow} \left\{\begin{array}{c} \dot{x} \mapsto 2.0;\ \dot{\mathtt{C}} \mapsto 2.0; \\ \hdashline \underset{\bullet}{x} \mapsto r_x + 2.0;\ \underset{\bullet}{\mathtt{C}} \mapsto r_1 + 2.0 \end{array}\right\}.$$

Since $((\mu_1' + 2.0) \odot \mu_2')(\dot{x}) = 2.0 = \nu_2'(x)$, the composed valuation reflects the real top frame ν_2'.

5.4.1 Lazy Semantics Lazy

On the basis of the above description, we formalize the lazy semantics Lazy of SRTA.

Definition 5.6 (Lazy Semantics Lazy). Let $\mathcal{A} = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ be an SRTA.

We define the infinite-PDS $(Q, \Gamma \times (\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}} \to \mathbb{R}_{\geq 0}), \hookrightarrow_d \cup \hookrightarrow_t)$ where discrete transition rules $\hookrightarrow_d$ and time elapsing transition rules $\hookrightarrow_t$ are defined as follows:

- A discrete transition rule $\langle q, \omega \rangle \hookrightarrow_d \langle q', \omega' \rangle$ is defined if there is $q \xrightarrow[\alpha]{\tau} q' \in \Delta$ and $\omega \overset{\tau}{\hookrightarrow} \omega'$ is defined by following Fig. 5.1.
- Time elapsing transition rules $\langle q, \langle \gamma, \mu \rangle \rangle \hookrightarrow_t \langle q, \langle \gamma, \mu' \rangle \rangle$ are defined for all $q \in Q$ and $\gamma \in \Gamma$ if $\mu \leq \mu'$ holds.

■

Note that, on the lazy semantics, we cannot always perform **pop**-transitions $\omega\, \mu_1\mu_2 \xrightarrow{\mathsf{pop}} \omega\, \mu$ because it requires the existence of a valuation μ_1' such that $\mu_1 \leq \mu_1'$ and $\mu_1' \mathbin{//} \mu_2$. However, this is not an obstacle to simulating the standard semantics because the well-formedness of the stack defined below ensures to perform **pop**-transitions.

Definition 5.7. Let μ_1 and μ_2 be valuations on $\underset{\bullet}{\mathcal{X}} \cup \dot{\mathcal{X}}$. If there is a valuation μ_1' such that $\mu_1 \leq \mu_1'$ and $\mu_1' \mathbin{//} \mu_2$, then we write $\mu_1 \lessapprox \mu_2$.

For two valuations μ_1 and μ_2 such that $\mu_1 \lessapprox \mu_2$, we define $\mu_2 \ominus \mu_1 \in \mathbb{R}_{\geq 0}$ by $\mu_2 \ominus \mu_1 \triangleq \mu_2(\underset{\bullet}{x}) - \mu_1(\dot{x})$ where x is a clock of $\mathcal{X}$. It is well-defined because $\mu_2(\underset{\bullet}{x}) - \mu_1(\dot{x})$ does not depend on the choice of a clock $x \in \mathcal{X}$. ■

Definition 5.8 (Well-formed Stack)**.** A stack $\omega = \langle \gamma_1, \mu_1 \rangle \langle \gamma_2, \mu_2 \rangle \ldots \langle \gamma_n, \mu_n \rangle \in \big(\Gamma \times (\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}} \to \mathbb{R}_{\geq 0})\big)^+$ is *well-formed* $\mathsf{WF}(\omega)$ if, for all $i \in [1..(n-1)]$, the following holds:

- The marked clock $\dot{\mathtt{C}}$ satisfies $\mu_i(\dot{\mathtt{C}}) = 0$;
- $\mu_i \lessapprox \mu_{i+1}$.

■

The following two basic properties are easily shown by definitions.

Proposition 5.1. Let μ_1 and μ_2 be valuations such that $\mu_1 \lessapprox \mu_2$. There exists the unique valuation μ_1' such that $\mu_1 \leq \mu_1'$ and $\mu_1' \mathbin{//} \mu_2$.

Proof. It suffices to take μ_1' as $\mu_1' = \mu_1 + (\mu_2 \ominus \mu_1)$. For any other δ ($\delta \neq \mu_2 \ominus \mu_1$), it is clear that $\mu_1 + \delta$ is not compatible with μ_2. □

Below, for $\mu_1 \lessapprox \mu_2$, we write $\mu_1 \lhd \mu_2$ to denote the valuation μ_1' uniquely determined by the above proposition.

Proposition 5.2 (WF is an invariant)**.** Let ω be a well-formed stack $\mathsf{WF}(\omega)$.

If $\langle q, \omega \rangle \to \langle q', \omega' \rangle$, then ω' is also a well-formed stack $\mathsf{WF}(\omega')$.

Proof. We consider the following nontrivial case induced by a **pop** transition:

$$\langle q, \omega\mu_1\mu_2\mu_3 \rangle \to \langle q', \omega\mu_1((\mu_2 \lhd \mu_3) \odot \mu_3) \rangle.$$

Since $\mathsf{WF}(\omega\mu_1\mu_2\mu_3)$, we have $\mu_1 \lessapprox \mu_2$. This and $\mu_2 \leq (\mu_2 \lhd \mu_3)$ imply $\mu_1 \lessapprox (\mu_2 \lhd \mu_3)$. It is clear that $(\mu_2 \lhd \mu_3)(x) = ((\mu_2 \lhd \mu_3) \odot \mu_3)(x)$ for any $x \in \underset{\bullet}{\mathcal{X}}_{\mathtt{C}}$. Therefore, we obtain $\mu_1 \lessapprox ((\mu_2 \lhd \mu_3) \odot \mu_3)$. □

We define the notations of stack correspondence and configuration correspondence between the standard and lazy semantics.

Definition 5.9 (Stack and Configuration Correspondence)**.** Let $w \in \big(\Gamma \times (\mathcal{X} \to \mathbb{R}_{\geq 0})\big)^+$ be a stack of the semantics Stnd and $\omega \in \big(\Gamma \times (\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}} \to \mathbb{R}_{\geq 0})\big)^+$ be a well-formed stack $\mathsf{WF}(\omega)$ of the semantics Lazy.

The *stack correspondence* relation is inductively defined as follows:

- $\langle \gamma, \nu \rangle \models \langle \gamma, \mu \rangle$ if $\dot{\nu} = \mu \restriction \dot{\mathcal{X}}$.
- $w\langle \gamma_1, \nu_1 \rangle \langle \gamma_2, \nu_2 \rangle \models \omega\langle \gamma_1, \mu_1 \rangle \langle \gamma_2, \mu_2 \rangle$ if
  - $\langle \nu_1, \nu_2 \rangle \models \mu_2$;
  - $w\langle \gamma_1, \nu_1 \rangle \models \omega\langle \gamma_1, \mu_1 \lhd \mu_2 \rangle$.

Let $\langle q, w \rangle$ and $\langle q', \omega \rangle$ be configurations of the semantics Stnd and Lazy, respectively. If $q = q'$ and $w \models \omega$, two configurations correspond and we write $\langle q, w \rangle \sim \langle q', \omega \rangle$. ■

For a given well-formed stack ω of paired valuations, there exists the unique stack w such that $w \models \omega$. This is shown by the following property.

Proposition 5.3 (Recover the concrete stack from a lazy one)**.** Let $\omega = \langle\gamma_1, \mu_1\rangle\langle\gamma_2, \mu_2\rangle \dots \langle\gamma_n, \mu_n\rangle$ be a well-formed stack $\mathsf{WF}(\omega)$.

The stack $w = \langle\gamma_1, \nu_1\rangle\langle\gamma_2, \nu_2\rangle \dots \langle\gamma_n, \nu_n\rangle$ defined as follows is the unique stack that satisfies $w \models \omega$:

- $\dot{\nu}_n = \mu_n \upharpoonright \dot{\mathcal{X}}$.
- For all $i \in [1..(n-1)]$, $\dot{\nu}_i = \mu_i \upharpoonright \dot{\mathcal{X}} + \sum_{j=i}^{n-1} (\mu_{j+1} \ominus \mu_j)$.

Proof. We proceed by induction on n.

Case $\omega = \langle\gamma_1, \mu_1\rangle$: The valuation ν_1 that satisfies $\dot{\nu}_1 = \mu_1 \upharpoonright \dot{\mathcal{X}}$ is the unique valuation such that $\langle\gamma_1, \nu_1\rangle \models \langle\gamma_1, \mu_1\rangle$.

Case $\omega = \langle\gamma_1, \mu_1\rangle \dots \langle\gamma_n, \mu_n\rangle\langle\gamma_{n+1}, \mu_{n+1}\rangle$:

Since ω is a well-formed stack, a stack $\omega_n = \langle\gamma_1, \mu_1\rangle \dots \langle\gamma_n, \mu_n + (\mu_{n+1} \ominus \mu_n)\rangle$ is also well formed and the induction hypothesis leads to the unique stack $w_n = \langle\gamma_1, \nu_1\rangle \dots \langle\gamma_1, \nu_n\rangle$ that satisfies $w_n \models \omega_n$ and the following:

$$\begin{aligned}
\dot{\nu}_n &= \mu_n \upharpoonright \dot{\mathcal{X}} + (\mu_{n+1} \ominus \mu_n) \\
&= \mu_n \upharpoonright \dot{\mathcal{X}} + \sum_{j=n}^{n} (\mu_{j+1} \ominus \mu_j), \\
\dot{\nu}_i &= \mu_i \upharpoonright \dot{\mathcal{X}} + (\mu_{n+1} \ominus \mu_n) + \sum_{j=i}^{n-1} (\mu_{j+1} \ominus \mu_j) \\
&= \mu_i \upharpoonright \dot{\mathcal{X}} + \sum_{j=i}^{n} (\mu_{j+1} \ominus \mu_j).
\end{aligned}$$

Let ν_{n+1} be the unique valuation such that $\dot{\nu}_{n+1} = \mu_{n+1} \upharpoonright \dot{\mathcal{X}}$. It is clear that $w_n\langle\gamma_{n+1}, \nu_{n+1}\rangle \models \omega$. The uniqueness of the $w_n\langle\gamma_{n+1}, \nu_{n+1}\rangle$ comes from the uniqueness of ν_{n+1} and w_n.

□

The above proposition immediately implies the following property that is key to connecting the two semantics Stnd and Lazy.

Proposition 5.4. Let $w\langle\gamma, \nu\rangle$ and $\omega\langle\gamma, \mu\rangle$ be stacks such that $w\langle\gamma, \nu\rangle \models \omega\langle\gamma, \mu\rangle$. For any $\delta \in \mathbb{R}_{\geq 0}$, $(w\langle\gamma, \nu\rangle) + \delta \models \omega\langle\gamma, \mu + \delta\rangle$.

We show the correspondence $\sim$ forms a bisimulation between Stnd and Lazy.

Lemma 5.3. Let $\langle q, w\rangle$ and $\langle q, \omega\rangle$ be configurations such that $w \models \omega$.

- If there is a timed transition $\langle q, w\rangle \overset{\delta}{\rightsquigarrow} \langle q, w'\rangle$ for some $\delta \in \mathbb{R}_{\geq 0}$, then there exists ω' such that $\langle q, \omega\rangle \to \langle q, \omega'\rangle$ and $w' \models \omega'$.
- If there is a discrete transition $\langle q, w\rangle \xrightarrow{\alpha} \langle q', w'\rangle$ for some $\alpha \in \Sigma \cup \{\epsilon\}$, then there exists ω' such that $\langle q, \omega\rangle \to \langle q', \omega'\rangle$ and $w' \models \omega'$.

$$\begin{tikzcd}
\langle q, w\rangle \arrow[r, "\textsc{Stnd}"'] \arrow[d, phantom, "\wr"] & \langle q', w'\rangle & & \langle q, w\rangle \arrow[r, "\textsc{Stnd}"'] \arrow[d, phantom, "\wr"] & \langle q', w'\rangle \arrow[d, phantom, "\wr"] \\
\langle q, \omega\rangle & & \Longrightarrow & \langle q, \omega\rangle \arrow[r, "\textsc{Lazy}"'] & \exists\langle q', \omega'\rangle.
\end{tikzcd}$$

Proof. We only consider the case of timed transitions. The other cases **push**, **dig**, $x \leftarrow I$, and **check**(φ) are trivial and the nontrivial case **pop** is shown in the same way as that of timed transitions.

Let us assume $\langle q, w\langle \gamma, \nu\rangle\rangle \overset{\delta}{\rightsquigarrow} \langle q, (w\langle \gamma, \nu\rangle){+}\delta\rangle$. From the assumption, we have $\langle q, w\langle \gamma, \nu\rangle\rangle \sim \langle q, \omega\langle \gamma, \mu\rangle\rangle$ for some valuation μ.

It suffices to show $(w\langle \gamma, \nu\rangle) + \delta \models \omega\langle \gamma, \mu + \delta\rangle$. This is clear from Proposition 5.4. $\square$

Lemma 5.4. Let $\langle q, w\rangle$ and $\langle q, \omega\rangle$ be configurations such that $w \models \omega$. We have the following:

$$\begin{tikzcd}
\langle q, w\rangle \arrow[d, phantom, "\wr"] & & & \langle q, w\rangle \arrow[r, "", "\textsc{Stnd}"'] \arrow[d, phantom, "\wr"] & \exists\langle q', w'\rangle \arrow[d, phantom, "\wr"] \\
\langle q, \omega\rangle \arrow[r, "", "\textsc{Lazy}"'] & \langle q', \omega'\rangle & \Longrightarrow & \langle q, \omega\rangle \arrow[r, "", "\textsc{Lazy}"'] & \langle q', \omega'\rangle .
\end{tikzcd}$$

Proof. We only consider the case of **pop** transitions. The other cases **push**, **dig**, $x \leftarrow I$, and **check**(φ) are trivial and the nontrivial case, timed transitions, is shown in the same way as the **pop** transitions.

We consider the following transition:

$$\langle q, \omega\langle \gamma_1, \mu_1\rangle\langle \gamma_2, \mu_2\rangle\langle \gamma_3, \mu_3\rangle\rangle \xrightarrow[\textsc{Lazy}]{\textbf{pop}} \\ \langle q', \omega\langle \gamma_1, \mu_1\rangle\langle \gamma_2, (\mu_2 \lhd \mu_3) \odot \mu_3\rangle\rangle.$$

(The following easy case

$$\langle q, \langle \gamma_1, \mu_1\rangle\langle \gamma_2, \mu_2\rangle\rangle \xrightarrow[\textsc{Lazy}]{\textbf{pop}} \langle q', \langle \gamma_1, (\mu_1 \lhd \mu_2) \odot \mu_2\rangle\rangle$$

is shown in the same argument for the above case.)

From the assumption, for some valuations ν_1, ν_2, and ν_3, we have following:

$$(\star) : \ \langle q, w\langle \gamma_1, \nu_1\rangle\langle \gamma_2, \nu_2\rangle\langle \gamma_3, \nu_3\rangle\rangle \sim \langle q, \omega\langle \gamma_1, \mu_1\rangle\langle \gamma_2, \mu_2\rangle\langle \gamma_3, \mu_3\rangle\rangle.$$

This implies the following:

$\sharp_1$ $\langle \nu_2, \nu_3\rangle \models \mu_3$.

$\sharp_2$ $w\langle \gamma_1, \nu_1\rangle\langle \gamma_2, \nu_2\rangle \sim \omega\langle \gamma_1, \mu_1\rangle\langle \gamma_2, \mu_2 \lhd \mu_3\rangle$.

Our aim is to show the following:

$$\langle q', w\langle \gamma_1, \nu_1\rangle\langle \gamma_2, \nu_3\rangle\rangle \sim \langle q', \omega\langle \gamma_1, \mu_1\rangle\langle \gamma_2, (\mu_2 \lhd \mu_3) \odot \mu_3\rangle\rangle.$$

First, we show $\langle \nu_1, \nu_3\rangle \models (\mu_2 \lhd \mu_3) \odot \mu_3$.

- Since $((\mu_2 \lhd \mu_3) \odot \mu_3)(\dot{x}) = \mu_3(\dot{x})$ and $\mu_3(\dot{x}) = \nu_3(x)$ (this comes from $\sharp_1$), we have $((\mu_2 \lhd \mu_3) \odot \mu_3)(\dot{x}) = \nu_3(x)$.
- Since $((\mu_2 \lhd \mu_3) \odot \mu_3)(\underset{\bullet}{x}) = (\mu_2 \lhd \mu_3)(\underset{\bullet}{x})$ and $(\mu_2 \lhd \mu_3)(\underset{\bullet}{x}) = \nu_1(x)$ (this comes from $\sharp_2$), we have $((\mu_2 \lhd \mu_3) \odot \mu_3)(\underset{\bullet}{x}) = \nu_1(x)$.

Next, we show the following:

$$w\langle \gamma_1, \nu_1\rangle \models \omega\langle \gamma_1, \mu_1 \lhd ((\mu_2 \lhd \mu_3) \odot \mu_3)\rangle.$$

Since $\mu_1 \lhd ((\mu_2 \lhd \mu_3) \odot \mu_3) = \mu_1 \lhd (\mu_2 \lhd \mu_3)$ is clear from the definition of $\lhd$, it suffices to show $w\langle \gamma_1, \nu_1\rangle \models \omega\langle \gamma_1, \mu_1 \lhd (\mu_2 \lhd \mu_3)\rangle$ and it is immediately shown by $\sharp_2$. $\square$

5.5 Collapsed Semantics

We cannot formalize the lazy semantics as finite PDS for the unboundedness and denseness of real numbers. In this section, we remove the unboundedness of real numbers. We will remove the denseness of real numbers in the next section and give a finite PDS semantics.

5.5.1 Removing the Unboundedness

In order to remove the unboundedness of real numbers, we translate a sufficiently large real number into the corresponding collapsed real number.

Although we have already defined the notion of collapsed valuations in Chapter 2, for the sake of completeness, we again define them.

Definition 5.10 (Upper-bound constant and Collapsed domain)**.** Let $\mathcal{A}$ be an SRTA and $\mathcal{I}$ be the set of intervals that appear in $\mathcal{A}$. The upper-bound constant M for $\mathcal{A}$ is defined as follows:

$$\mathsf{M} \triangleq \max\{\, i, j : [i : j] \in \mathcal{I},\ (i : j) \in \mathcal{I}\} + 1.$$

The set of collapsed real numbers $\mathbb{C}$ is defined as follows:

$$\mathbb{C} \triangleq \big([0..(\mathsf{M}-1)] \cup \{\infty\}\big) \times [0, 1).$$

The collapsing function $\mathcal{C} : \mathbb{R}_{\geq 0} \to \mathbb{C}$ is defined as follows:

$$\mathcal{C}(r) \triangleq \begin{cases} (\infty\ , \mathit{frac}(r)) & \text{if } r \geq \mathsf{M}, \\ (\lfloor r \rfloor, \mathit{frac}(r)) & \text{if } r < \mathsf{M}. \end{cases}$$

For a concrete valuation $\nu : \mathcal{X} \to \mathbb{R}_{\geq 0}$, we define the *collapsed valuation of* ν by $\mathcal{C}(\nu)(x) \triangleq \mathcal{C}(\nu(x))$. ■

We write $v.r$ to denote (v, r). Moreover, $\lfloor v.r \rfloor$ and $\mathit{frac}(v.r)$ denote v and r, respectively. We use Greek letters $\lambda, \ldots$ to denote a collapsed valuation. Especially, we use $\Lambda, \ldots$ to denote a collapsed valuation on a marked clock set $\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}}$.

The following basic properties are immediately shown by the above definition.

Proposition 5.5. Let ν_1 and ν_2 be valuations on a finite clock set $\mathcal{X}$. If $\mathcal{C}(\nu_1) = \mathcal{C}(\nu_2)$,

Validity. $\nu_1 \models \varphi$ iff $\nu_2 \models \varphi$ for any constraint φ.

Copying. $\mathcal{C}(\nu_1[x := y]) = \mathcal{C}(\nu_2[x := y])$ for any $x, y \in \mathcal{X}$.

Updating. $\mathcal{C}(\nu_1[x := r]) = \mathcal{C}(\nu_2[x := r])$ for any $x \in \mathcal{X}$ and $r \in \mathbb{R}_{\geq 0}$.

Evolving. $\mathcal{C}(\nu_1 + \delta) = \mathcal{C}(\nu_2 + \delta)$ for any $\delta \in \mathbb{R}_{\geq 0}$.

On the basis of Proposition 5.5, we define operations for collapsed valuations as follows.

Definition 5.11. Let $\mathcal{X}$ be a finite clock set, ν and Λ be concrete and collapsed valuations on $\mathcal{X}$ such that $\mathcal{C}(\nu) = \Lambda$.

- For a constraint φ, we write $\Lambda \models \varphi$ if $\nu \models \varphi$.
- For a real number $r \in \mathbb{R}_{\geq 0}$, $\Lambda[x := r] \triangleq \mathcal{C}(\nu[x := r])$.
- For clocks $x, y \in \mathcal{X}$, $\Lambda[x := y] \triangleq \mathcal{C}(\nu[x := y])$.
- For a real number $\delta \in \mathbb{R}_{\geq 0}$, $\Lambda + \delta \triangleq \mathcal{C}(\nu + \delta)$.

■

The above definition is well-defined because Proposition 5.5 ensures that the result does not depend on the choice of a witness ν for Λ.

We define a (quasi) ordering $\lambda \preccurlyeq \lambda'$ for collapsed valuations that corresponds to the ordering $\leq$ on concrete valuations.

Definition 5.12. Let λ and λ' be collapsed valuations. We write $\lambda \preccurlyeq \lambda'$ if there are two concrete valuations ν and ν' such that $\nu \leq \nu'$, $\mathcal{C}(\nu) = \lambda$, and $\mathcal{C}(\nu') = \lambda'$. ■

We define the compatibility and composition in the same way as the lazy semantics.

Definition 5.13. Let Λ_1 and Λ_2 be collapsed valuations on a finite set of clocks $\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}}$.

- If $\Lambda_1(\dot{x}) = \Lambda_2(\underset{\bullet}{x})$ for any $x \in \mathcal{X}_{\mathtt{C}}$, then Λ_1 is *compatible* with Λ_2 and we write $\Lambda_1 \mathbin{/\!/} \Lambda_2$.
- If $\Lambda_1 \mathbin{/\!/} \Lambda_2$, then the *composed* collapsed valuation $\Lambda_1 \odot \Lambda_2 : \underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}} \to \mathbb{C}$ is defined as follows:
  - $(\Lambda_1 \odot \Lambda_2)(\underset{\bullet}{x}) \triangleq \Lambda_1(\underset{\bullet}{x})$ for all $x \in \mathcal{X}_{\mathtt{C}}$.
  - $(\Lambda_1 \odot \Lambda_2)(\dot{x}) \triangleq \Lambda_2(\dot{x})$ for all $x \in \mathcal{X}_{\mathtt{C}}$.

■

5.5.2 Why Do We Need Reference Clock

Let us consider the following pop transition of the lazy semantics:

$$\frac{\mu_2 = \left\{\begin{array}{c} \dot{x} \mapsto 3.5;\ \dot{\mathtt{C}} \mapsto 2.0; \\ \hdashline \underset{\bullet}{x} \mapsto 4.0;\ \underset{\bullet}{\mathtt{C}} \mapsto 2.0 \end{array}\right\}}{\mu_1 = \left\{\begin{array}{c} \dot{x} \mapsto 2.0;\ \dot{\mathtt{C}} \mapsto 0.0; \\ \hdashline \underset{\bullet}{x} \mapsto 0.5;\ \underset{\bullet}{\mathtt{C}} \mapsto 0.0 \end{array}\right\}} \xrightarrow{\textsf{pop}} \mu$$

where

$$\mu = \left\{\begin{array}{c} \dot{x} \mapsto 3.5;\ \dot{\mathtt{C}} \mapsto 2.0; \\ \hdashline \underset{\bullet}{x} \mapsto 2.5;\ \underset{\bullet}{\mathtt{C}} \mapsto 2.0 \end{array}\right\}.$$

The above run is simulated as follows under $\mathsf{M} = 3$ in the collapsed semantics:

$$\frac{\Lambda_2 = \left\{\begin{array}{c} \dot{x} \mapsto \infty.5;\ \dot{\mathtt{C}} \mapsto 2.0; \\ \hdashline \underset{\bullet}{x} \mapsto \infty.0;\ \underset{\bullet}{\mathtt{C}} \mapsto 2.0 \end{array}\right\}}{\Lambda_1 = \left\{\begin{array}{c} \dot{x} \mapsto 2.0;\ \dot{\mathtt{C}} \mapsto 0.0; \\ \hdashline \underset{\bullet}{x} \mapsto 0.5;\ \underset{\bullet}{\mathtt{C}} \mapsto 0.0 \end{array}\right\}} \Rightarrow \Lambda$$

where

$$\Lambda = \left\{\begin{array}{c} \dot{x} \mapsto \infty.5;\ \dot{\mathtt{C}} \mapsto 2.0; \\ \hdashline \underset{\bullet}{x} \mapsto 2.5;\ \underset{\bullet}{\mathtt{C}} \mapsto 2.0 \end{array}\right\}.$$

In order to compute Λ from Λ_1 and Λ_2, we evolve Λ_1 to Λ_1' ($\Lambda_1 \preccurlyeq \Lambda_1'$) until $\Lambda_1' \mathbin{/\!/} \Lambda_2$ and compose them as $\Lambda = \Lambda_1' \odot \Lambda_2$. For this case, Λ_1' is uniquely determined as $\Lambda_1' = \Lambda_1 + 2.0$ by the reference clocks $\dot{\mathtt{C}}$ of Λ_1 and $\underset{\bullet}{\mathtt{C}}$ of Λ_2. In general, the presence of reference clocks ensures the uniqueness of such Λ_1' (see Proposition 5.6 and Lemma 5.5).

On the other hand, if we drop reference clocks, then the above **pop** transition behaves nondeterministically. Let us consider the following stack that is obtained by removing the reference clocks from the above transition:

$$\frac{\lambda_2 = \left\{\underset{\bullet}{x} \mapsto \infty.0;\ \dot{x} \mapsto \infty.5\right\}}{\lambda_1 = \left\{\underset{\bullet}{x} \mapsto 0.5;\ \dot{x} \mapsto 2.0\right\}}.$$

Due to the absence of reference clocks, there are the following three possibilities about λ_1' such that $\lambda_1 \preccurlyeq \lambda_1'$ and $\lambda_1' \mathbin{/\!/} \lambda_2$:

- $\lambda_1 + 1.0 = \{\underset{\bullet}{x} \mapsto 1.5;\ \ \dot{x} \mapsto \infty.0\} \mathbin{/\!/} \lambda_2$.
- $\lambda_1 + 2.0 = \{\underset{\bullet}{x} \mapsto 2.5;\ \ \dot{x} \mapsto \infty.0\} \mathbin{/\!/} \lambda_2$.
- $\lambda_1 + 3.0 = \{\underset{\bullet}{x} \mapsto \infty.5;\ \dot{x} \mapsto \infty.0\} \mathbin{/\!/} \lambda_2$.

Thus, we have the following nondeterministic transition:

$$\lambda_1\lambda_2 \xrightarrow{\textsf{pop}} \{\lambda_1 + 1 \odot \lambda_2, \lambda_1 + 2 \odot \lambda_2, \lambda_1 + 3 \odot \lambda_2\}.$$

Since the valuation μ only justifies $(\lambda_1 + 2.0) \odot \lambda_2$ for $\mathcal{C}(\mu)(\underset{\bullet}{x}) = 2.5$, the lazy semantics without reference clocks cannot simulate the collapsed semantics.

We show a key property to ensure the uniqueness of **pop** transitions.

Proposition 5.6. Let λ, λ', λ'' be collapsed valuations such that $\lambda \preccurlyeq \lambda'$ and $\lambda \preccurlyeq \lambda''$.
If the following conditions hold, then $\lambda' = \lambda''$: there is a clock $\mathtt{C}$ such that

($\sharp_1$) $\lambda(\mathtt{C}) = 0.0$; and

($\sharp_2$) $\lambda'(\mathtt{C}) = \lambda''(\mathtt{C})$.

Proof. We proceed by case analysis on $\lfloor \lambda'(\mathtt{C}) \rfloor$:

Case $\lfloor \lambda'(\mathtt{C}) \rfloor \neq \infty$: The condition ($\sharp_2$) implies $\lambda'(\mathtt{C}) = \lambda''(\mathtt{C}) = (i, r)$ for some $i \neq \infty$ and $r \in [0, 1)$. This means that $\lambda' = \lambda + (i, r)$ and $\lambda'' = \lambda + (i, r)$. Hence $\lambda' = \lambda''$.

Case $\lfloor \lambda'(\mathtt{C}) \rfloor = \infty$: The condition ($\sharp_2$) implies $\lambda'(\mathtt{C}) = \lambda''(\mathtt{C}) = (\infty, r)$ for some $r \in [0, 1)$. In contrast to the above case, in general, there may exist two distinct real numbers δ_1 and δ_2 such that $\lambda' = \lambda + \delta_1$ and $\lambda'' = \lambda + \delta_2$. Here, we assume $\delta_1 > \delta_2$.

It suffices to show $\mathit{frac}(\lambda'(x)) = \mathit{frac}(\lambda''(x))$ and $\lfloor \lambda'(x) \rfloor = \lfloor \lambda''(x) \rfloor$ for any clock x.

***$\mathit{frac}(\lambda'(x)) = \mathit{frac}(\lambda''(x))$*:** We have $\delta_1 - \delta_2 \in \mathbb{N}$ because $\mathit{frac}(\lambda'(\mathtt{C})) = \mathit{frac}(\lambda''(\mathtt{C}))$. This implies $\mathit{frac}(\lambda'(x)) = \mathit{frac}(\lambda''(x))$ for any clock x.

$\lfloor \lambda'(x) \rfloor = \lfloor \lambda''(x) \rfloor$: The condition ($\sharp_1$) implies that any clocks are collapsed: $\lfloor \lambda'(x) \rfloor = \lfloor \lambda''(x) \rfloor = \infty$ for any clock x.

□

Remark: We cannot replace the two conditions of Proposition 5.6 by the following single condition:

($\sharp_3$) There is a clock $\mathtt{C}$ such that $\lambda'(\mathtt{C}) = \lambda''(\mathtt{C})$.

For example, let us consider the following valuations under $\mathsf{M} = 4$:

$$\begin{aligned}
\lambda &= \{x \mapsto 0.5;\ \mathtt{C} \mapsto 3.0\}, \\
\lambda' &= \lambda + 1 = \{x \mapsto 1.5;\ \mathtt{C} \mapsto \infty.0\}, \\
\lambda'' &= \lambda + 2 = \{x \mapsto 2.5;\ \mathtt{C} \mapsto \infty.0\}.
\end{aligned}$$

We have $\lambda'(\mathtt{C}) = \lambda''(\mathtt{C})$; however, $\lambda' \neq \lambda''$.

$$\frac{\Lambda_1' = \Lambda_1[\dot{\mathtt{C}} := 0] \quad \Lambda_1' \mathbin{/\!/} \Lambda_2 \quad \Lambda_2(\dot{x}) = 0\ (\forall x \in \mathcal{X}_{\mathtt{C}})}{\langle\gamma_1,\Lambda_1\rangle \xhookrightarrow{\mathsf{push}(\gamma_2)} \langle\gamma_1,\Lambda_1'\rangle\langle\gamma_2,\Lambda_2\rangle}\ \mathbf{push}(\gamma_2)$$

$$\frac{\Lambda_1 \preccurlyeq \Lambda_1' \quad \Lambda_1' \mathbin{/\!/} \Lambda_2}{\langle\gamma_1,\Lambda_1\rangle\,\langle\gamma_2,\Lambda_2\rangle \xhookrightarrow{\mathsf{pop}(\gamma_2)} \langle\gamma_1,\Lambda_1' \odot \Lambda_2\rangle}\ \mathbf{pop}(\gamma_2) \qquad \frac{\Lambda \models \dot{\varphi}}{\langle\gamma,\Lambda\rangle \xhookrightarrow{\mathsf{check}(\varphi)} \langle\gamma,\Lambda\rangle}\ \mathbf{check}(\varphi)$$

$$\frac{\Lambda_2' = \Lambda_2[\dot{x} := \underset{\bullet}{y}]}{\langle\gamma_1,\Lambda_1\rangle\langle\gamma_2,\Lambda_2\rangle \xhookrightarrow{\mathsf{dig}(x,y)} \langle\gamma_1,\Lambda_1\rangle\langle\gamma_2,\Lambda_2'\rangle}\ \mathbf{dig}(x,y) \qquad \frac{r \in I \quad \Lambda' = \Lambda[\dot{x} := r]}{\langle\gamma,\Lambda\rangle \xhookrightarrow{x \leftarrow I} \langle\gamma,\Lambda'\rangle}\ x \leftarrow I$$

Figure 5.2: Definition of actions on Coll semantics

5.5.3 Collapsed Semantics Coll

Collapsed valuations lead to the collapsed semantics Coll, which removes the unboundedness of real numbers from the lazy semantics Lazy.

Definition 5.14 (Collapsed Semantics)**.** Let $\mathcal{A} = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ be an SRTA.

We define the infinite-PDS $(Q, \Gamma \times (\underset{\bullet}{\mathcal{X}_{\mathtt{C}}} \cup \dot{\mathcal{X}}_{\mathtt{C}} \to \mathbb{C}), \hookrightarrow_d \cup \hookrightarrow_t)$ where discrete transition rules $\hookrightarrow_d$ and time elapsing transition rules $\hookrightarrow_t$ are defined as follows:

- A discrete transition rule $\langle q, \boldsymbol{w}\rangle \hookrightarrow_d \langle q', \boldsymbol{w'}\rangle$ is defined if there is $q \xrightarrow[\alpha]{\tau} q' \in \Delta$ and $\boldsymbol{w} \overset{\tau}{\hookrightarrow} \boldsymbol{w'}$ is defined by following Fig. 5.2.
- Time elapsing transition rules $\langle q, \langle\gamma,\Lambda\rangle\rangle \hookrightarrow_t \langle q, \langle\gamma,\Lambda'\rangle\rangle$ are defined for all $q \in Q$ and $\gamma \in \Gamma$ if $\Lambda \preccurlyeq \Lambda'$ holds.

■

In the same way as the lazy semantics Lazy, we define the notion of the well-formed stack and prove some properties of well-formed stacks.

Definition 5.15. Let Λ_1 and Λ_2 be collapsed valuations on $\underset{\bullet}{\mathcal{X}_{\mathtt{C}}} \cup \dot{\mathcal{X}}_{\mathtt{C}}$. If there is a collapsed valuation Λ_1' such that $\Lambda_1 \preccurlyeq \Lambda_1'$ and $\Lambda_1' \mathbin{/\!/} \Lambda_2$, then we write $\Lambda_1 \precapprox \Lambda_2$. ■

Definition 5.16 (Well-formed Stack)**.** A stack $\boldsymbol{w} = \langle\gamma_1,\Lambda_1\rangle\langle\gamma_2,\Lambda_2\rangle \ldots \langle\gamma_n,\Lambda_n\rangle$ is *well-formed* $\mathsf{WF}(\boldsymbol{w})$ if for all $i \in [1..(n-1)]$

- The marked clock $\dot{\mathtt{C}}$ satisfies $\Lambda_i(\dot{\mathtt{C}}) = 0$; and
- $\Lambda_i \precapprox \Lambda_{i+1}$.

■

As the lazy semantics, the well-formedness WF forms an invariant.

Proposition 5.7. Let $\langle q, \boldsymbol{w}\rangle$ be a configuration where $\boldsymbol{w}$ is a well-formed stack $\mathsf{WF}(\boldsymbol{w})$. If $\langle q, \boldsymbol{w}\rangle \to \langle q, \boldsymbol{w'}\rangle$, then $\boldsymbol{w'}$ is also a well-formed stack $\mathsf{WF}(\boldsymbol{w'})$.

Proof. This is shown by the same argument of the proof in Proposition 5.2. □

As we mentioned above, the presence of reference clocks is key to the following property and the determinacy of **pop** transitions. The following property, which corresponds to Proposition 5.1, is immediate from Proposition 5.6.

Lemma 5.5. If $\mathsf{WF}(\boldsymbol{w}\langle\gamma_1,\Lambda_1\rangle\langle\gamma_2,\Lambda_2\rangle)$, then there exists the *unique* Λ_1' such that $\Lambda \preccurlyeq \Lambda_1'$ and $\Lambda_1' \mathbin{/\!/} \Lambda_2$. We use $\Lambda_1 \lhd \Lambda_2$ for such the unique valuation.

We define the stack and configuration correspondence between LAZY and COLL.

Definition 5.17. Let $\omega = \mu_1 \dots \mu_n$ and $\boldsymbol{w} = \Lambda_1 \dots \Lambda_n$ be well-formed stacks $\mathsf{WF}(\omega), \mathsf{WF}(\boldsymbol{w})$ of the lazy and collapsed semantics, respectively.

The *stack correspondence* $\omega \models \boldsymbol{w}$ is defined if $\mathcal{C}(\mu_i) = \Lambda_i$ for all $i \in [1..n]$.

It is naturally extended to configurations: we write $\langle q, \omega\rangle \sim \langle q', \boldsymbol{w}\rangle$ if $q = q'$ and $\omega \models \boldsymbol{w}$. ■

We show the correspondence $\sim$ forms a bisimulation between LAZY and COLL.

Lemma 5.6. Let $\langle q, \omega\rangle$ and $\langle q, \boldsymbol{w}\rangle$ be configurations such that $\omega \models \boldsymbol{w}$.

$$\begin{tikzcd}
\langle q, \omega\rangle \arrow[r, "\textsc{Lazy}"'] \arrow[d, phantom, "\wr"] & \langle q', \omega'\rangle & & \langle q, \omega\rangle \arrow[r, "\textsc{Lazy}"'] \arrow[d, phantom, "\wr"] & \langle q', \omega'\rangle \arrow[d, phantom, "\wr"] \\
\langle q, \boldsymbol{w}\rangle & & \Longrightarrow & \langle q, \boldsymbol{w}\rangle \arrow[r, "\textsc{Coll}"'] & \exists\langle q', \boldsymbol{w'}\rangle.
\end{tikzcd}$$

Proof. Since the stack correspondence relation $\omega \models \boldsymbol{w}$ is defined in a componentwise way, the forward simulation is immediately shown for all the transition rules. □

Lemma 5.7. Let $\langle q, \omega\rangle$ and $\langle q, \boldsymbol{w}\rangle$ be configurations such that $\omega \models \boldsymbol{w}$.

$$\begin{tikzcd}
\langle q, \omega\rangle \arrow[d, phantom, "\wr"] & & & \langle q, \omega\rangle \arrow[r, "\textsc{Lazy}"'] \arrow[d, phantom, "\wr"] & \exists\langle q', \omega'\rangle \arrow[d, phantom, "\wr"] \\
\langle q, \boldsymbol{w}\rangle \arrow[r, "\textsc{Coll}"'] & \langle q', \boldsymbol{w'}\rangle & \Longrightarrow & \langle q, \boldsymbol{w}\rangle \arrow[r, "\textsc{Coll}"'] & \langle q', \boldsymbol{w'}\rangle
\end{tikzcd}$$

Proof. We consider the case of **pop**-transitions:

$$\langle q, \boldsymbol{w}\langle\gamma_1, \Lambda_1\rangle\langle\gamma_2, \Lambda_2\rangle\rangle \xrightarrow{\textsf{pop}} \langle q', \boldsymbol{w}\langle\gamma_1, \Lambda_1' \odot \Lambda_2\rangle\rangle$$

for some Λ_1' such that $\Lambda_1 \preccurlyeq \Lambda_1'$ and $\Lambda_1' \mathbin{/\!/} \Lambda_2$. All the other cases are not difficult. Using Lemma 5.5, Λ_1' is uniquely determined as $\Lambda_1 \lhd \Lambda_2$. From the assumption, for some valuations, we have the following:

$$(\star) \quad \langle q, \omega\langle\gamma_1, \mu_1\rangle\langle\gamma_2, \mu_2\rangle\rangle \sim \langle q, \boldsymbol{w}\langle\gamma_1, \Lambda_1\rangle\langle\gamma_2, \Lambda_2\rangle\rangle.$$

We show the following correspondence:

$$\omega\,\langle\gamma_1, (\mu_1 \lhd \mu_2) \odot \mu_2\rangle \models \boldsymbol{w}\langle\gamma_1, (\Lambda_1 \lhd \Lambda_2) \odot \Lambda_2\rangle.$$

It suffices to confirm $\mathcal{C}(\mu_2) = \Lambda_2$ and $\mathcal{C}(\mu_1 \lhd \mu_2) = \Lambda_1 \lhd \Lambda_2$. The former is trivial from $(\star)$. To show the latter equation, we use the following simple property:

$$\text{if } \mu_a \mathbin{/\!/} \mu_b, \text{ then } \mathcal{C}(\mu_a) \mathbin{/\!/} \mathcal{C}(\mu_b).$$

We use a real number δ such that $\mu_1 + \delta = \mu_1 \lhd \mu_2$. Since $\mathcal{C}(\mu_1 + \delta) = \Lambda_1 + \delta$, which comes from $\mathcal{C}(\mu_1) = \Lambda_1$, and $\mathcal{C}(\mu_2) = \Lambda_2$, we have $\Lambda_1 + \delta \mathbin{/\!/} \Lambda_2$ from the above property. This compatibility and Lemma 5.5 imply $\Lambda_1 + \delta = \Lambda_1 \lhd \Lambda_2$. Therefore, our goal is rewritten as $\mathcal{C}(\mu_1 + \delta) = \Lambda_1 + \delta$ and it is already shown. □

Remark: As we have seen in Section 5.5.2, we need reference clocks to ensure the uniqueness of **pop**-transitions on the collapsed semantics. However, this does not answer the question why we need to introduce reference clocks at the lazy semantics rather than collapsed semantics. The answer of that question is that the forward simulation does not hold between the lazy semantics without reference clocks and the collapsed semantics. Let us consider the following **pop**-transition under $\mathsf{M} = 2$:

$$\frac{\Lambda_2 = \left\{\begin{array}{c} \overset{\bullet}{\mathtt{C}} \mapsto 1.5;\ \overset{\bullet}{x} \mapsto \infty.5; \\ \hdashline \underset{\bullet}{\mathtt{C}} \mapsto 1.5;\ \underset{\bullet}{x} \mapsto \infty.5 \end{array}\right\}}{\Lambda_1 = \left\{\begin{array}{c} \overset{\bullet}{\mathtt{C}} \mapsto 0.0;\ \overset{\bullet}{x} \mapsto \infty.0; \\ \hdashline \underset{\bullet}{\mathtt{C}} \mapsto 0.0;\ \underset{\bullet}{x} \mapsto 0.0 \end{array}\right\}} \xrightarrow{\textsf{pop}} \Lambda$$

where $\Lambda = (\Lambda_1 + 1.5) \odot \Lambda_2$ and thus it forms the following valuation:

$$\Lambda = \left\{\underset{\bullet}{\mathtt{C}} \mapsto 1.5;\ \underset{\bullet}{x} \mapsto 1.5;\ \overset{\bullet}{\mathtt{C}} \mapsto 1.5;\ \overset{\bullet}{x} \mapsto \infty.5\right\}.$$

If we do not introduce reference clocks at the lazy semantics, the following stack realizes the above stack $\Lambda_1\Lambda_2$:

$$\frac{\mu_2 = \left\{\underset{\bullet}{x} \mapsto 2.5;\ \overset{\bullet}{x} \mapsto 2.5\right\}}{\mu_1 = \left\{\overset{\bullet}{x} \mapsto 2.0;\ \underset{\bullet}{x} \mapsto 0.0\right\}} \models \frac{\Lambda_2}{\Lambda_1.}$$

On the lazy semantics without reference clocks, we have the following **pop**-transition:

$$\mu_1\mu_2 \xrightarrow{\textsf{pop}} (\mu_1 + 0.5) \odot \mu_2 = \left\{\underset{\bullet}{x} \mapsto 0.5;\ \overset{\bullet}{x} \mapsto 2.5\right\}.$$

For the obtained valuations, $(\mu_1 + 0.5) \odot \mu_2 \not\models \Lambda$ because $((\mu_1 + 0.5) \odot \mu_2)(\underset{\bullet}{x}) = 0.5$ and $\Lambda(\underset{\bullet}{x}) = 1.5$. To prevent this problem, we need reference clocks at the lazy semantics rather than at the collapsed semantics.

5.5.4 Upper Bound for Configuration Reachability Problem

To show lemmas for the configuration reachability problem of SRTA, as with Section 2.4 and 2.4.1, we redefine the upper bound constant M obtained from an SRTA.

Let $\mathcal{A}$ be an SRTA and $\langle q_0, \langle \bot, \mathbf{0}\rangle\rangle \xrightarrow[\textsc{Stnd}]{}{}^{*}_{?} \langle q, w\rangle$ be a query of the configuration reachability problem of SRTA. For a stack $w = \langle \gamma_1, \nu_1\rangle\langle \gamma_2, \nu_2\rangle \dots \langle \gamma_n, \nu_n\rangle \in (\Gamma \times (\mathcal{X} \to \mathbb{R}_{\geq 0}))^*$ on the standard semantics, we define the maximum value $\max(w)$ of the real numbers in w as follows:

$$\max(w) \triangleq \max\{\nu_i(x) : i \in [1..n], x \in \mathcal{X}\}.$$

We again take an upper-bound constant $\mathsf{M} \in \mathbb{N}$ so that it could satisfy the following:

$$\begin{aligned} &\mathsf{M} \geq \max\{\, j : (i,j) \text{ or } [i,j] \text{ appears in } \mathcal{A}\} + 1, \\ &\mathsf{M} \geq \max(w) + 1. \end{aligned}$$

Under the new definition of upper-bound constant, we have the following adequate simulation properties for the configuration reachability problem. To formalize the statement, we define a stack correspondence between the standard and collapsed semantics in a natural way.

Definition 5.18. Let w and $\boldsymbol{w}$ be a stack of the standard and collapsed semantics, respectively. If there is a stack ω of the lazy semantics such that $w \models \omega$ and $\omega \models \boldsymbol{w}$, we write $w \models \boldsymbol{w}$. ■

Lemma 5.8. Let $\langle q, w\rangle$ be a configuration of the standard semantics. The following are equivalent:

(1) $\langle q_{\text{init}}, \langle \bot, \mathbf{0}_{\mathcal{X}}\rangle\rangle \xrightarrow[\textsc{Stnd}]{}{}^{*} \langle q, w\rangle$.

(2) $\langle q_{\text{init}}, \langle \bot, \mathbf{0}_{\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \overset{\bullet}{\mathcal{X}}_{\mathtt{C}}}\rangle\rangle \xrightarrow[\textsc{Coll}]{}{}^{*} \langle q, \boldsymbol{w}\rangle$ for some stack $\boldsymbol{w}$ such that $\mathsf{WF}(\boldsymbol{w})$ and $w \models \boldsymbol{w}$.

Proof. The direction (1) $\Rightarrow$ (2) is shown by using the forward simulation Lemmas 5.3 and 5.6 sequentially.

We consider the other direction (2) $\Rightarrow$ (1). The assumption $w \models \boldsymbol{w}$ implies the existence of a stack ω such that $w \models \omega \models \boldsymbol{w}$.

First, the forward simulation of Coll by Lazy (Lemma 5.7) implies the following transition:

$$\langle q_{\text{init}}, \langle \bot, \mathbf{0}_{\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}}} \rangle\rangle \xrightarrow[\textsc{Lazy}]{}{}^{*} \langle q, \omega' \rangle$$

where ω' is a well-formed stack such that $\omega' \models \boldsymbol{w}$.

Next, the forward simulation of Lazy by Stnd (Lemma 5.4) implies the following transition:

$$\langle q_{\text{init}}, \langle \bot, \mathbf{0} \rangle\rangle \xrightarrow[\textsc{Stnd}]{}{}^{*} \langle q, w' \rangle$$

where w' is a stack such that $w' \models \omega'$.

We need to show $w = w'$ from $w \models \omega \models \boldsymbol{w}$ and $w' \models \omega' \models \boldsymbol{w}$. However, since $\omega \neq \omega'$ in general, showing $w = w'$ is not trivial. Let us consider the following two stacks under $\mathsf{M} = 4$:

$$\omega = \cfrac{\mu_2 = \left\{ \begin{array}{c} \dot{x} \mapsto 2.0;\ \dot{\mathtt{C}} \mapsto 4.5 \\ \hdashline \underset{\bullet}{x} \mapsto 3.5;\ \underset{\bullet}{\mathtt{C}} \mapsto 2.5 \end{array} \right\}}{\mu_1 = \left\{ \begin{array}{c} \dot{x} \mapsto 1.0;\ \dot{\mathtt{C}} \mapsto 0.0 \\ \hdashline \underset{\bullet}{x} \mapsto 8.0;\ \underset{\bullet}{\mathtt{C}} \mapsto 4.0 \end{array} \right\}},$$

$$\omega' = \cfrac{\mu_2' = \left\{ \begin{array}{c} \dot{x} \mapsto 2.0;\ \dot{\mathtt{C}} \mapsto 6.5 \\ \hdashline \underset{\bullet}{x} \mapsto 3.5;\ \underset{\bullet}{\mathtt{C}} \mapsto 2.5 \end{array} \right\}}{\mu_1' = \left\{ \begin{array}{c} \dot{x} \mapsto 1.0;\ \dot{\mathtt{C}} \mapsto 0.0 \\ \hdashline \underset{\bullet}{x} \mapsto 6.0;\ \underset{\bullet}{\mathtt{C}} \mapsto 5.0 \end{array} \right\}}$$

where ω and ω' are different and realize the same stack $\mathcal{C}(\mu_1)\mathcal{C}(\mu_2)$ because $\mathcal{C}(\mu_1) = \mathcal{C}(\mu_1')$ and $\mathcal{C}(\mu_2) = \mathcal{C}(\mu_2')$. Although $\omega \neq \omega'$, by the definition of M, $\mu_1(\dot{x}) = \mu_1'(\dot{x})$ and $\mu_2(\underset{\bullet}{x}) = \mu_2'(\underset{\bullet}{x})$ and it implies $\mu_2 \ominus \mu_1 = \mu_2' \ominus \mu_1' = 2.5$. Therefore, Proposition 5.3 implies $w = w'$ for the unique stacks w and w' such that $w \models \omega$ and $w' \models \omega$:

$$w = w' = \cfrac{\{x \mapsto 2.0\}}{\{x \mapsto 3.5\}}.$$

We formalize the above argument in the general case to show $w = w'$:

1. If $\omega = \langle \gamma_1, \mu_1 \rangle \dots \langle \gamma_n, \mu_n \rangle \langle \gamma_{n+1}, \mu_{n+1} \rangle$, then we can show the following from $w \models \omega$ and Proposition 5.3:

   - $\mu_1(\dot{x}) < \mathsf{M}$ for all $x \in \mathcal{X}$.
   - $\mu_i(x) < \mathsf{M}$ for all $i \in [2..(n+1)]$ and $x \in \underset{\bullet}{\mathcal{X}} \cup \dot{\mathcal{X}}$.

2. If $\omega' = \langle \gamma_1, \mu_1' \rangle \dots \langle \gamma_n, \mu_n' \rangle \langle \gamma_{n+1}, \mu_{n+1}' \rangle$, then we can show the following from $\omega \models \boldsymbol{w}$ and $\omega' \models \boldsymbol{w}$:

   - $\mu_1(\dot{x}) = \mu_1'(\dot{x}) < \mathsf{M}$ for all $x \in \mathcal{X}$.
   - $\mu_i(x) = \mu_i'(x) < \mathsf{M}$ for all $i \in [2..(n+1)]$ and $x \in \underset{\bullet}{\mathcal{X}} \cup \dot{\mathcal{X}}$.

   These imply $\mu_{i+1} \ominus \mu_i = \mu_{i+1}' \ominus \mu_i'$ for $i \in [1..n]$.

3. Assume $w = \langle \gamma_1, \nu_1 \rangle \dots \langle \gamma_{n+1}, \nu_{n+1} \rangle$ and $w' = \langle \gamma_1, \nu_1' \rangle \dots \langle \gamma_{n+1}, \nu_{n+1}' \rangle$. Proposition 5.3 and $\mu_{i+1} \ominus \mu_i = \mu_{i+1}' \ominus \mu_i'$ for all $i \in [1..n]$ imply $\nu_i = \nu_i'$ for all $i \in [1..n]$. Thus, $w = w'$.

The fact $w = w'$ implies the following transition:

$$\langle q_{\text{init}}, \langle \bot, \mathbf{0} \rangle\rangle \xrightarrow[\textsc{Stnd}]{}{}^{*} \langle q, w \rangle.$$

$\square$

5.6 Digital Valuations and Finite-PDS Semantics

The collapsed semantics cannot be formalized as a finite PDS for the denseness of real numbers. We define digital valuations to remove the denseness and the digitized semantics DIGI as a finite PDS.

Although we have already defined the notion of digital valuations in Chapter 2, for the sake of completeness, we again define them and state their properties without proofs.

5.6.1 Digital Valuations

Our definition of digital valuations equals to that of regions given by Abdulla et al. in [AAS12a].

Definition 5.19 (Digital Valuations)**.** Let $\mathcal{X}$ be a finite clock set. A sequence of sets $\boldsymbol{d} = d_0\, d_1 \dots d_n$, where $d_i \subseteq \mathcal{X} \times \{0, 1, \dots, \mathsf{M} - 1, \infty\}$, is a *digital valuation* on $\mathcal{X}$ if $\boldsymbol{d}$ satisfies the following conditions:

- Every clock in $\mathcal{X}$ appears in $\boldsymbol{d}$ exactly once.
- Except d_0, all the sets d_i are not empty: $d_i \neq \emptyset$ for all $i \in [1..n]$.

For a clock $x \in \mathcal{X}$ and set d_i of $\boldsymbol{d}$, we write $x \in d_i$ if $(x, v) \in d_i$ for some $v \in \{0, 1, \dots, \mathsf{M} - 1, \infty\}$. ■

Intuitively, each digital valuation is obtained by forgetting the fractional parts of a collapsed valuation and only keeping the order of the fractional parts of it. On the basis of this intuition, we define a realization relation between collapsed and digital valuations.

Definition 5.20 (Realization)**.** Let $\mathcal{X}$ be a finite clock set, λ be a collapsed valuation on $\mathcal{X}$, and $\boldsymbol{d} = d_0 d_1 \dots d_n$ be a digital valuation on $\mathcal{X}$. We write $\lambda \models \boldsymbol{d}$ if the following hold:

- For all $x \in \mathcal{X}$, $(x, \lfloor \lambda(x) \rfloor) \in d_i$ for some i.
- For all $x \in \mathcal{X}$, $\mathit{frac}(\lambda(x)) = 0.0$ iff $x \in d_0$.
- $\mathit{frac}(\lambda(x)) < \mathit{frac}(\lambda(y))$ iff $x \in d_i$ and $y \in d_j$ for some $i < j$.

■

Proposition 5.8. The realization relation $\models$ is *functional*: for a collapsed valuation Λ, there exists the unique digital valuation $\mathcal{D}(\Lambda)$ such that $\Lambda \models \mathcal{D}(\Lambda)$.

For the special set d_0 that contains clocks whose fractional parts are 0.0, we use the notation $\{\dots\}_0$ as above.

We define the successor relation $\boldsymbol{d} \vdash \boldsymbol{d'}$ that corresponds to time elapsing on collapsed valuations.

Definition 5.21 (Successor)**.** Let $\boldsymbol{d}$ and $\boldsymbol{d'}$ be digital valuations. The valuation $\boldsymbol{d'}$ is the unique successor of $\boldsymbol{d}$ ($\boldsymbol{d} \vdash \boldsymbol{d'}$) if one of the following holds:

Case $\boldsymbol{d} = d_0 d_1 \dots d_n$ and $d_0 \neq \emptyset$:

$$d_0\, d_1 \dots d_n \vdash \emptyset\, d_0\, d_1 \dots d_n.$$

Case $\boldsymbol{d} = \emptyset\, d_1 \dots d_{n-1} d_n$:

$$\emptyset\, d_1 \dots d_{n-1}\, d_n \vdash d'_n\, d_1 \dots d_{n-1},$$

where d'_n satisfies the following: if $(x, k) \in d_n$,

$$\begin{cases} (x, k+1) \in d'_n & \text{if } k < \mathsf{M} - 1, \\ (x, \infty) \quad\; \in d'_n & \text{if } k = \mathsf{M} - 1 \text{ or } k = \infty. \end{cases}$$

We use $\vdash^*$ to denote the reflexive transitive closure of $\vdash$. ■

On the realization relation between collapsed and digital valuations, the similar properties to Proposition 5.5 hold.

Proposition 5.9. Let Λ_1 and Λ_2 be collapsed valuations on $\mathcal{X}$. If $\mathcal{D}(\Lambda_1) = \mathcal{D}(\Lambda_2)$,

Validity. $\Lambda_1 \models \varphi$ iff $\Lambda_2 \models \varphi$ for any constraint φ.

Copying. $\mathcal{D}(\Lambda_1[x := y]) = \mathcal{D}(\Lambda_2[x := y])$ for any $x, y \in \mathcal{X}$.

Integer Updating. $\mathcal{D}(\Lambda_1[x := n]) = \mathcal{D}(\Lambda_2[x := n])$ for any $x \in \mathcal{X}$ and $n \in \{0, 1, \ldots, \mathsf{M} - 1\}$.

Evolving. If $\Lambda_1 \preccurlyeq \Lambda_1'$, then there exists Λ_2' such that $\Lambda_2 \preccurlyeq \Lambda_2'$ and $\mathcal{D}(\Lambda_1') = \mathcal{D}(\Lambda_2')$.

On the basis of Proposition 5.9, we define basic operations on the digital valuations in the same way as those of the collapsed valuations.

Definition 5.22. Let $\boldsymbol{d}$ be a digital valuation and λ be a collapsed valuation that satisfies $\mathcal{D}(\lambda) = \boldsymbol{d}$.

- For a constraint φ, $\boldsymbol{d} \models \varphi$ if $\lambda \models \varphi$.
- For two clocks x, y, $\boldsymbol{d}[x := y] \triangleq \mathcal{D}(\lambda[x := y])$.
- For a natural number n such that $0 \leq n < \mathsf{M}$, $\boldsymbol{d}[x := n] \triangleq \mathcal{D}(\lambda[x := n])$.

■

In order to treat updating $x \leftarrow I$ for a clock x and interval I, we need to define nondeterministic updating $\boldsymbol{d}[x \leftarrow I]$ for a digital valuation $\boldsymbol{d}$.

Definition 5.23 (Nondeterministic Update)**.** Let $\boldsymbol{d}$ be a digital valuation. We define the *update* $\boldsymbol{d}[x \leftarrow I]$ for a clock x and an interval I as follows:

$$\boldsymbol{d}[x \leftarrow I] \triangleq \{\, \mathcal{D}(\, \Lambda[x := r]\,) : r \in I, \Lambda \models \boldsymbol{d} \,\}\,.$$

■

Example. Let $\boldsymbol{d} = \{(x, 0)\}_0 \{(y, 1)\}$ be a digital valuation. The updating $\boldsymbol{d}[x \leftarrow (2, 3)]$ generates the following three digital valuations:

$$\boldsymbol{d}[\, x \leftarrow (2, 3)\,] = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3\}\,,$$

$$\begin{aligned} \boldsymbol{d}_1 &= \{\}_0 \{(x, 2)\} \{(y, 1)\}\,, \\ \boldsymbol{d}_2 &= \{\}_0 \{(x, 2), (y, 1)\}\,, \\ \boldsymbol{d}_3 &= \{\}_0 \{(y, 1)\} \{(x, 2)\}\,. \end{aligned}$$

5.6.2 Forward and Backward Simulation of Time Elapsing

For time elapsing, we have two forward simulations and one backward simulation below.

Proposition 5.10. Let λ be a collapsed valuation and $\boldsymbol{d}$ be a digital valuation.

$$\begin{array}{ccc} \lambda & \preccurlyeq & \lambda' \\ \Top & & \\ \boldsymbol{d} & & \end{array} \quad\Longrightarrow\quad \begin{array}{ccc} \lambda & \preccurlyeq & \lambda' \\ \Top & & \Top \\ \boldsymbol{d} & \vdash^* & \exists \boldsymbol{d}'. \end{array}$$

Proposition 5.11. Let λ be a collapsed valuation and $\boldsymbol{d}$ be a digital valuation.

$$\begin{array}{ccc} \lambda & & \\ \Top & & \\ \boldsymbol{d} & \vdash^* & \boldsymbol{d}' \end{array} \quad\Longrightarrow\quad \begin{array}{ccc} \lambda & \preccurlyeq & \exists\lambda' \\ \Top & & \Top \\ \boldsymbol{d} & \vdash^* & \boldsymbol{d}'. \end{array}$$

Proposition 5.12. Let λ' be a collapsed valuation and $\boldsymbol{d}$ be a digital valuation.

$$\begin{array}{ccccccc} & & \lambda' & & \exists\lambda & \preccurlyeq & \lambda' \\ & & \text{⫪} & \Longrightarrow & \text{⫪} & & \text{⫪} \\ \boldsymbol{d} & \vdash^* & \boldsymbol{d'} & & \boldsymbol{d} & \vdash^* & \boldsymbol{d'}. \end{array}$$

Remark. Proposition 5.12 is crucial to the backward simulation lemma (Lemma 5.10) and peculiar to collapsed valuations. Indeed, this fails on the realization relation $\nu \models \boldsymbol{d}$ of concrete and digital valuations. Let us consider the following under $\mathsf{M} = 2$:

$$\begin{array}{ccc} & & \{x \mapsto 2.0\} \\ & & \text{⫪} \\ \{\}_0\,\{x \mapsto \infty\} & \xrightarrow[\text{DIGI}]{} & \{x \mapsto \infty\}_0. \end{array}$$

For the backward simulation, we need to find a valuation ν such that $\nu < \{x \mapsto 2.0\}$ and $\nu \models \{\}_0\,\{x \mapsto \infty\}$. However, since $\nu < \{x \mapsto 2.0\}$ requires $\nu(x) < 2.0$, $\nu \models \{\}_0\,\{x \mapsto \infty\}$ never holds.

5.6.3 Nondeterministic Composition

As the collapsed semantics COLL, we define the compatibility and composition.

Definition 5.24 (Compatibility and Composition)**.** Let $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ be digital valuations on a finite marked clock set $\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}}$.

- The valuation $\boldsymbol{D}_1$ is *compatible* with the valuation $\boldsymbol{D}_2$ ($\boldsymbol{D}_1 \mathbin{/\!/} \boldsymbol{D}_2$) if there are collapsed valuations Λ_1 and Λ_2 such that $\Lambda_1 \models \boldsymbol{D}_1$, $\Lambda_2 \models \boldsymbol{D}_2$, and $\Lambda_1 \mathbin{/\!/} \Lambda_2$.
- The composed valuations are defined by:

$$\boldsymbol{D}_1 \odot \boldsymbol{D}_2 \triangleq \left\{ \mathcal{D}(\Lambda) : \begin{array}{l} \Lambda_1 \models \boldsymbol{D}_1,\ \Lambda_2 \models \boldsymbol{D}_2, \\ \Lambda_1 \mathbin{/\!/} \Lambda_2,\ \Lambda = \Lambda_1 \odot \Lambda_2 \end{array} \right\}.$$

■

Nondeterminacy of **pop** *transitions.* Let us consider the following **pop** transition under $\mathsf{M} = 3$:

$$\frac{\boldsymbol{D}_2 = \left\{(\dot{\mathtt{C}}, 2), (\underset{\bullet}{\mathtt{C}}, 2), (\underset{\bullet}{x}, \infty)\right\}_0 \{(\dot{x}, \infty)\}}{\boldsymbol{D}_1 = \left\{(\underset{\bullet}{\mathtt{C}}, 0), (\dot{\mathtt{C}}, 0), (\dot{x}, 2)\right\}_0 \left\{(\underset{\bullet}{x}, 0)\right\}} \xrightarrow{\textsf{pop}} \cdots .$$

When performing the **pop** transition, first we compute $\boldsymbol{D}'_1$ such that $\boldsymbol{D}_1 \vdash^* \boldsymbol{D}'_1$ and $\boldsymbol{D}'_1 \mathbin{/\!/} \boldsymbol{D}_2$. For this case, $\boldsymbol{D}'_1$ is uniquely determined as follows:

$$\boldsymbol{D}'_1 = \{(\underset{\bullet}{\mathtt{C}}, 2), (\dot{\mathtt{C}}, 2), (\dot{x}, \infty)\}_0\{(\underset{\bullet}{x}, 2)\}.$$

Next we compute the new top frame by composing the two digital valuations $\boldsymbol{D}'_1$ and $\boldsymbol{D}_2$ as $\boldsymbol{D}'_1 \odot \boldsymbol{D}_2$:

$$\begin{array}{ll} & \dfrac{\boldsymbol{D}_2 = \left\{(\dot{\mathtt{C}}, 2), (\underset{\bullet}{\mathtt{C}}, 2), (\underset{\bullet}{x}, \infty)\right\}_0 \{(\dot{x}, \infty)\}}{\boldsymbol{D}'_1 = \{(\underset{\bullet}{\mathtt{C}}, 2), (\dot{\mathtt{C}}, 2), (\dot{x}, \infty)\}_0\{(\underset{\bullet}{x}, 2)\}} \\ \xrightarrow{\odot} & \{\boldsymbol{D}_3, \boldsymbol{D}'_3, \boldsymbol{D}''_3\} \end{array}$$

where

$$\begin{array}{rcl} \boldsymbol{D}_3 & = & \left\{(\underset{\bullet}{\mathtt{C}}, 2), (\dot{\mathtt{C}}, 2)\right\}_0 \{(\underset{\bullet}{x}, 2)\}\{(\dot{x}, \infty)\}, \\ \boldsymbol{D}'_3 & = & \left\{(\underset{\bullet}{\mathtt{C}}, 2), (\dot{\mathtt{C}}, 2)\right\}_0 \{(\underset{\bullet}{x}, 2), (\dot{x}, \infty)\}, \\ \boldsymbol{D}''_3 & = & \left\{(\underset{\bullet}{\mathtt{C}}, 2), (\dot{\mathtt{C}}, 2)\right\}_0 \{(\dot{x}, \infty)\}\{(\underset{\bullet}{x}, 2)\}. \end{array}$$

When composing the two digital valuations, we need to decide the order between $\underset{\bullet}{x}$ of $\boldsymbol{D}'_1$ and $\dot{x}$ of $\boldsymbol{D}_2$. However, since we dismiss the fractional values in digital valuations, there are no clues to decide the ordering, and so we generate all the possible orderings as above.

$$\frac{\boldsymbol{D}_1[\dot{\mathtt{C}} := 0] \mathbin{/\!/} \boldsymbol{D}_2 \quad \boldsymbol{D}_2 \models \dot{x} \in [0,0]\ (\forall x \in \mathcal{X}_{\mathtt{C}})}{\langle\gamma_1,\boldsymbol{D}_1\rangle \xhookrightarrow{\mathsf{push}(\gamma_2)} \langle\gamma_1,\boldsymbol{D}_1[\dot{\mathtt{C}} := 0]\rangle\,\langle\gamma_2,\boldsymbol{D}_2\rangle}$$

$$\frac{\boldsymbol{D}_1 \vdash^* \boldsymbol{D}'_1 \quad \boldsymbol{D}'_1 \mathbin{/\!/} \boldsymbol{D}_2 \quad \boldsymbol{D} \in \boldsymbol{D}'_1 \odot \boldsymbol{D}_2}{\langle\gamma_1,\boldsymbol{D}_1\rangle\,\langle\gamma_2,\boldsymbol{D}_2\rangle \xhookrightarrow{\mathsf{pop}(\gamma_2)} \langle\gamma_1,\boldsymbol{D}\rangle} \qquad \frac{\boldsymbol{D} \models \dot{\varphi}}{\langle\gamma,\boldsymbol{D}\rangle \xhookrightarrow{\mathsf{check}(\varphi)} \langle\gamma,\boldsymbol{D}\rangle}$$

$$\frac{\boldsymbol{D}'_2 = \boldsymbol{D}_2[\dot{x} := \underset{\bullet}{y}]}{\langle\gamma_1,\boldsymbol{D}_1\rangle\langle\gamma_2,\boldsymbol{D}_2\rangle \xhookrightarrow{\mathsf{dig}(x,y)} \langle\gamma_1,\boldsymbol{D}_1\rangle\langle\gamma_2,\boldsymbol{D}'_2\rangle} \qquad \frac{\boldsymbol{D}' \in \boldsymbol{D}[\dot{x} \leftarrow I]}{\langle\gamma,\boldsymbol{D}\rangle \xhookrightarrow{x \leftarrow I} \langle\gamma,\boldsymbol{D}'\rangle}$$

Figure 5.3: Definition of actions on DIGI Semantics

As we have seen in the previous section, pop transitions for well-formed stacks behave deterministically on the collapsed semantics. Hence, the collapsed semantics cannot capture the nondeterminacy of pop transitions of the digitized semantics.

5.6.4 Digitized Semantics

Digital valuations lead to the digitized semantics DIGI, which is defined as a finite PDS.

Definition 5.25 (Digitized Semantics DIGI)**.** Let $\mathcal{A} = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ be an SRTA.

We use $\mathbb{D}$ for the finite set of digital valuations on $\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}}$.

We define the finite PDS $(Q, \Gamma \times \mathbb{D}, \hookrightarrow_d \cup \hookrightarrow_t)$ where discrete transition rules $\hookrightarrow_d$ and time elapsing transition rules $\hookrightarrow_t$ are defined as follows:

- A discrete transition rule $\langle q, \boldsymbol{W}\rangle \hookrightarrow_d \langle q', \boldsymbol{W'}\rangle$ is defined if there is $q \xrightarrow[\alpha]{\tau} q' \in \Delta$ and $\boldsymbol{W} \overset{\tau}{\hookrightarrow} \boldsymbol{W'}$ is defined by following Fig. 5.3.
- Time elapsing transition rules $\langle q, \langle\gamma, \boldsymbol{D}\rangle\rangle \hookrightarrow_t \langle q, \langle\gamma, \boldsymbol{D'}\rangle\rangle$ are defined for all $q \in Q$ and $\gamma \in \Gamma$ if $\boldsymbol{D} \vdash^* \boldsymbol{D'}$ holds.

■

In the same way as the lazy and collapsed semantics, we define the stack well-formedness for the digitized semantics.

Definition 5.26. Let $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ be digital valuations on a finite marked clock set $\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}}$.

If there is a digital valuation $\boldsymbol{D}'_1$ such that $\boldsymbol{D}_1 \vdash^* \boldsymbol{D}'_1$ and $\boldsymbol{D}'_1 \mathbin{/\!/} \boldsymbol{D}_2$, then we write $\boldsymbol{D}_1 \lessapprox \boldsymbol{D}_2$. ■

Definition 5.27 (Well-formed Stack)**.** A stack $\boldsymbol{W} = \langle\gamma_1, \boldsymbol{D}_1\rangle\langle\gamma_2, \boldsymbol{D}_2\rangle \ldots \langle\gamma_n, \boldsymbol{D}_n\rangle$ is *well-formed* $\mathsf{WF}(\boldsymbol{W})$ if for all $i \in [1..(n-1)]$

- $\boldsymbol{D}_i \models \dot{\mathtt{C}} \in [0,0]$ holds for the clock $\dot{\mathtt{C}}$; and
- $\boldsymbol{D}_i \lessapprox \boldsymbol{D}_{i+1}$.

■

The well-formedness of the digitized semantics forms an invariant.

Proposition 5.13. Let $\boldsymbol{W}$ be a well-formed stack $\mathsf{WF}(\boldsymbol{W})$. If $\langle q, \boldsymbol{W}\rangle \to \langle q', \boldsymbol{W'}\rangle$, then $\mathsf{WF}(\boldsymbol{W'})$.

Proof. This is shown in the same argument of the proof in Proposition 5.2. □

In order to show the forward and backward simulation properties between the collapsed and digitized semantics, we define stack and configuration correspondences.

Definition 5.28. Let $\boldsymbol{w} = \Lambda_1, \ldots, \Lambda_n$ and $\boldsymbol{W} = \boldsymbol{D}_1, \ldots, \boldsymbol{D}_n$ be well-formed stacks $\mathsf{WF}(\boldsymbol{w})$ and $\mathsf{WF}(\boldsymbol{W})$ of the collapsed and digitized semantics, respectively.

The *stack correspondence* $\boldsymbol{w} \models \boldsymbol{W}$ holds if $\Lambda_i \models \boldsymbol{D}_i$ for all $i \in [1..n]$.

It is naturally extended to configurations: we write $\langle q, \boldsymbol{w}\rangle \sim \langle q', \boldsymbol{W}\rangle$ if $q = q'$ and $\boldsymbol{w} \models \boldsymbol{W}$. ■

The following forward simulation of Coll by Digi can be shown easily.

Lemma 5.9. Let $\langle q, \boldsymbol{w}\rangle$ and $\langle q, \boldsymbol{W}\rangle$ be configurations such that $\boldsymbol{w} \models \boldsymbol{W}$.

$$\begin{tikzcd}
\langle q, \boldsymbol{w}\rangle \arrow[r, "\textsc{Coll}"'] \arrow[d, phantom, "\wr"] & \langle q', \boldsymbol{w'}\rangle & & \langle q, \boldsymbol{w}\rangle \arrow[r, "\textsc{Coll}"'] \arrow[d, phantom, "\wr"] & \langle q', \boldsymbol{w'}\rangle \arrow[d, phantom, "\wr"] \\
\langle q, \boldsymbol{W}\rangle & & \Longrightarrow & \langle q, \boldsymbol{W}\rangle \arrow[r, "\textsc{Digi}"'] & \exists\langle q', \boldsymbol{W'}\rangle.
\end{tikzcd}$$

Proof. We consider the following nontrivial case:

$$\begin{aligned} & \langle q\ , \boldsymbol{w}\langle\gamma_1, \Lambda_1\rangle\langle\gamma_2, \Lambda_2\rangle\langle\gamma_3, \Lambda_3\rangle\rangle \\ \xrightarrow{\mathsf{pop}(\gamma_3)}\ & \langle q', \boldsymbol{w}\langle\gamma_1, \Lambda_1\rangle\langle\gamma_2, (\Lambda_2 \lhd \Lambda_3) \odot \Lambda_3\rangle\rangle. \end{aligned}$$

Our assumption can be rewritten as follows:

$$\begin{gathered} \langle q\ , \boldsymbol{w}\langle\gamma_1, \Lambda_1\rangle\langle\gamma_2, \Lambda_2\rangle\langle\gamma_3, \Lambda_3\rangle\rangle \\ \wr \\ \langle q\ , \boldsymbol{W}\langle\gamma_1, \boldsymbol{D}_1\rangle\langle\gamma_2, \boldsymbol{D}_2\rangle\langle\gamma_3, \boldsymbol{D}_3\rangle\rangle. \end{gathered}$$

Since $\Lambda_2 \preccurlyeq (\Lambda_2 \lhd \Lambda_3)$, we apply Proposition 5.10 to $\Lambda_2 \models \boldsymbol{D}_2$ and obtain $\boldsymbol{D}'_2$ such that $\boldsymbol{D}_2 \vdash^* \boldsymbol{D}'_2$ and $(\Lambda_2 \lhd \Lambda_3) \models \boldsymbol{D}'_2$. It is clear that $\boldsymbol{D}'_2 \mathbin{/\!/} \boldsymbol{D}_3$ from the definition of $/\!/$.

Since $\Lambda_2 \lhd \Lambda_3 \models \boldsymbol{D}'_2$ and $\Lambda_3 \models \boldsymbol{D}_3$, the definition of $\odot$ implies that there is a digital valuation $\boldsymbol{D}$ such that $\boldsymbol{D} \in \boldsymbol{D}'_2 \odot \boldsymbol{D}_3$ and $(\Lambda_2 \lhd \Lambda_3) \odot \Lambda_3 \models \boldsymbol{D}$. This leads to the following transition:

$$\begin{aligned} & \langle q\ , \boldsymbol{W}\langle\gamma_1, \boldsymbol{D}_1\rangle\langle\gamma_2, \boldsymbol{D}_2\rangle\langle\gamma_3, \boldsymbol{D}_3\rangle\rangle \\ \xrightarrow{\mathsf{pop}(\gamma_3)}\ & \langle q', \boldsymbol{W}\langle\gamma_1, \boldsymbol{D}_1\rangle\langle\gamma_2, \boldsymbol{D}\rangle\rangle. \end{aligned}$$

Now $(\Lambda_2 \lhd \Lambda_3) \odot \Lambda_3 \models \boldsymbol{D}$ concludes the proof. □

5.6.5 Backward Simulation

Compared to the forward simulation of Coll by Digi, the counterpart does not hold for the nondeterminacy of **pop** rules in Digi as we have stated in Section 5.6.3. However, we can show the backward simulation by Proposition 5.12.

Lemma 5.10. Let $\langle q', \boldsymbol{w}'\rangle$ and $\langle q', \boldsymbol{W}'\rangle$ be configurations such that $\boldsymbol{w}' \models \boldsymbol{W}'$.

$$\begin{tikzcd}
& \langle q', \boldsymbol{w'}\rangle \arrow[d, phantom, "\wr"] & & \exists\langle q, \boldsymbol{w}\rangle \arrow[r, "\textsc{Coll}"'] \arrow[d, phantom, "\wr"] & \langle q', \boldsymbol{w'}\rangle \arrow[d, phantom, "\wr"] \\
\langle q, \boldsymbol{W}\rangle \arrow[r, "\textsc{Digi}"'] & \langle q', \boldsymbol{W'}\rangle & \Longrightarrow & \langle q, \boldsymbol{W}\rangle \arrow[r, "\textsc{Digi}"'] & \langle q', \boldsymbol{W'}\rangle.
\end{tikzcd}$$

Proof. We consider the two cases **push** and **time** (Here we write $\boldsymbol{c}_1 \xrightarrow{\mathsf{time}} \boldsymbol{c}_2$ to denote a transition caused by a time elapsing transition rule $\langle q, \langle\gamma, \boldsymbol{D}\rangle\rangle \hookrightarrow_t \langle q, \langle\gamma, \boldsymbol{D'}\rangle\rangle$). The cases **dig**, $x \leftarrow I$, and **check**(φ) are trivial and the case **pop** is shown in the same argument of the proof of the case **time**.

Proof of Case push We consider the following **push** transition:

$$\begin{aligned} & \langle q\ , \boldsymbol{W}\langle \gamma_1, \boldsymbol{D}_1\rangle\rangle \\ \xrightarrow{\mathsf{push}(\gamma_2)} & \langle q', \boldsymbol{W}\langle \gamma_1, \boldsymbol{D}_1[\dot{\mathtt{C}} := 0]\rangle\langle \gamma_2, \boldsymbol{D}_2\rangle\rangle. \end{aligned}$$

Our assumption is rewritten as follows:

$$\begin{gathered} \langle q', \boldsymbol{w}\langle \gamma_1, \Lambda_1'\rangle\langle \gamma_2, \Lambda_2\rangle\rangle \\ \wr \\ \langle q', \boldsymbol{W}\langle \gamma_1, \boldsymbol{D}_1[\dot{\mathtt{C}} := 0]\rangle\langle \gamma_2, \boldsymbol{D}_2\rangle\rangle \end{gathered}$$

for some collapsed valuations Λ_1' and Λ_2.

We define $\Lambda_1 : (\dot{\mathcal{X}}_{\mathtt{C}} \cup \underset{\bullet}{\mathcal{X}}_{\mathtt{C}}) \to \mathbb{C}$ so that it satisfies both the following conditions:

1. $\Lambda_1(x) = \Lambda_1'(x)$ for all $x \in \dot{\mathcal{X}} \cup \underset{\bullet}{\mathcal{X}}_{\mathtt{C}}$.
2. $\Lambda_1(\dot{\mathtt{C}}) = r$ where $r \in \mathbb{R}_{\geq 0}$ is a non-negative real number that satisfies $\Lambda_1 \models \boldsymbol{D}_1$.

We show the following:

(A) $\boldsymbol{w}\langle \gamma_1, \Lambda_1\rangle \models \boldsymbol{W}\langle \gamma_1, \boldsymbol{D}_1\rangle$.

(B) $\mathsf{WF}(\boldsymbol{w}\langle \gamma_1, \Lambda_1\rangle)$.

(C) $\langle q, \boldsymbol{w}\langle \gamma_1, \Lambda_1\rangle\rangle \xrightarrow{\mathsf{push}(\gamma_2)} \langle q', \boldsymbol{w}\langle \gamma_1, \Lambda_1'\rangle\langle \gamma_2, \Lambda_2\rangle\rangle$.

(A) is immediate from the definition of Λ_1. (B) is shown by the assumption $\mathsf{WF}(\boldsymbol{w}\langle \gamma_1, \Lambda_1'\rangle\langle \gamma_2, \Lambda_2\rangle)$ and the fact that $\Lambda_1 \upharpoonright \underset{\bullet}{\mathcal{X}}_{\mathtt{C}} = \Lambda_1' \upharpoonright \underset{\bullet}{\mathcal{X}}_{\mathtt{C}}$.

For (C), we show the following:

- $\Lambda_1'(\dot{x}) = \Lambda_2(\underset{\bullet}{x})$ for any $x \in \mathcal{X}_{\mathtt{C}}$.

We split this into the two parts: showing $\lfloor \Lambda_1'(\dot{x}) \rfloor = \lfloor \Lambda_2(\underset{\bullet}{x}) \rfloor$ and $\mathit{frac}(\Lambda_1'(\dot{x})) = \mathit{frac}(\Lambda_2(\underset{\bullet}{x}))$.

The former $\lfloor \Lambda_1'(\dot{x}) \rfloor = \lfloor \Lambda_2(\underset{\bullet}{x}) \rfloor$ comes from $\Lambda_1' \models \boldsymbol{D}_1[\dot{\mathtt{C}} := 0]$, $\Lambda_2 \models \boldsymbol{D}_2$, and $\boldsymbol{D}_1[\dot{\mathtt{C}} := 0] \mathbin{/\!/} \boldsymbol{D}_2$.

The latter $\mathit{frac}(\Lambda_1'(\dot{x})) = \mathit{frac}(\Lambda_2(\underset{\bullet}{x}))$ is somewhat involved and shown by the following steps:

1. $\Lambda_1' \precapprox \Lambda_2$ holds from $\mathsf{WF}(\boldsymbol{w}\langle \gamma_1, \Lambda_1'\rangle\langle \gamma_2, \Lambda_2\rangle)$.
2. $\mathit{frac}(\Lambda_1'(\dot{\mathtt{C}})) = 0.0$ holds from $\Lambda_1' \models \boldsymbol{D}_1[\dot{\mathtt{C}} := 0]$.
3. $\mathit{frac}(\Lambda_2(\underset{\bullet}{\mathtt{C}})) = 0.0$ holds from $\boldsymbol{D}_1[\dot{\mathtt{C}} := 0] \mathbin{/\!/} \boldsymbol{D}_2$ and $\Lambda_2 \models \boldsymbol{D}_2$.

Finally, combining $\Lambda_1' \precapprox \Lambda_2$ and $\mathit{frac}(\Lambda_1'(\dot{\mathtt{C}})) = \mathit{frac}(\Lambda_2'(\underset{\bullet}{\mathtt{C}})) = 0.0$, we have $\mathit{frac}(\Lambda_1'(\dot{x})) = \mathit{frac}(\Lambda_2(\underset{\bullet}{x}))$ for all clock $x \in \underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}}$. □

We remark that the above proof needs reference clocks; indeed, without reference clocks, the case **push** fails. Let us consider the following diagram:

$$\begin{array}{ccc} & & \dfrac{\{\dot{x} \mapsto 0;\ \underset{\bullet}{x} \mapsto 0.5\}}{\{\underset{\bullet}{x} \mapsto 0;\ \dot{x} \mapsto 0.4\}} \\ & & \top\!\!\top \\ \{(\underset{\bullet}{x}, 0)\}_0 \{(\dot{x}, 0)\} & \xrightarrow[\textsc{Digi}]{\mathsf{push}} & \dfrac{\{(\dot{x}, 0)\}_0 \{(\underset{\bullet}{x}, 0)\}}{\{(\underset{\bullet}{x}, 0)\}_0 \{(\dot{x}, 0)\}} \end{array}$$

Although the above diagram satisfies all the conditions of Lemma 5.10, the following transition is not allowed on the collapsed semantics without reference clocks:

$$\{\underset{\bullet}{x} \mapsto 0;\ \dot{x} \mapsto 0.4\} \ \not\xrightarrow[\textsc{Coll}]{\mathsf{push}}\ \dfrac{\{\dot{x} \mapsto 0;\ \underset{\bullet}{x} \mapsto 0.5\}}{\{\underset{\bullet}{x} \mapsto 0;\ \dot{x} \mapsto 0.4\}}$$

Compared to this, the following stack is not a well-formed stack and we do not need to consider it:

$$\dfrac{\{\ldots;\ \underset{\bullet}{\mathtt{C}} \mapsto 0.0;\ \underset{\bullet}{x} \mapsto 0.5\}}{\{\ldots;\ \dot{\mathtt{C}} \mapsto 0.0;\ \dot{x} \mapsto 0.4\}}$$

Proof of Case time Before giving the proof for the case **time**, we state a technical lemma that is key to the case **time**.

Lemma 5.11 (Key Lemma for Backward Simulation)**.** Let λ_1, λ_2, and λ_3 be collapsed valuations on a finite clock set $\mathcal{X}$. Also, let $\boldsymbol{d}_1$, $\boldsymbol{d}_2$, and $\boldsymbol{d}_3$ be digital valuations on the set $\mathcal{X}$.

$$\text{If } \begin{array}{ccccc} \lambda_1 & & \lambda_2 & & \lambda_3 \\ \rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} \\ \boldsymbol{d}_1 & \vdash^* & \boldsymbol{d}_2 & \vdash^* & \boldsymbol{d}_3 \end{array} \wedge \left\{ \begin{array}{c} \lambda_1 \precsim \lambda_3, \\ \lambda_2 \precsim \lambda_3, \\ \exists x. x \in_0 \boldsymbol{d}_1 \end{array} \right\},$$
$$\text{then } \lambda_1 \precsim \lambda_2.$$

For $\boldsymbol{d} = d_0 d_1 \dots d_n$, we write $x \in_0 \boldsymbol{d}$ if $x \in d_0$, i.e., a clock x belongs to d_0 of $\boldsymbol{d}$.

We put the proof of this lemma on Section 5.9 for the readability and continue to give a proof for the case **time**. First, we consider the following case:

$$\langle q, \langle \gamma, \boldsymbol{D} \rangle \rangle \to \langle q, \langle \gamma, \boldsymbol{D'} \rangle \rangle$$

where $\boldsymbol{D} \vdash^* \boldsymbol{D'}$. This case is immediate from Proposition 5.12.

Next, we consider the following general case:

$$\langle q, \boldsymbol{W} \langle \gamma_1, \boldsymbol{D}_1 \rangle \langle \gamma_2, \boldsymbol{D}_2 \rangle \rangle \to \langle q, \boldsymbol{W} \langle \gamma_1, \boldsymbol{D}_1 \rangle \langle \gamma_2, \boldsymbol{D}'_2 \rangle \rangle$$

where $\boldsymbol{D}_2 \vdash^* \boldsymbol{D}'_2$. The assumption can be rewritten as follows for some valuations:

$$\boldsymbol{w}\, \Lambda_1\, \Lambda'_2 \models \boldsymbol{W} \boldsymbol{D}_1 \boldsymbol{D}'_2.$$

Since $\boldsymbol{D}_2 \vdash^* \boldsymbol{D}'_2$, by Proposition 5.12, there is a collapsed valuation Λ_2 such that

$$\begin{array}{ccc} \Lambda_2 & \precsim & \Lambda'_2 \\ \rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} \\ \boldsymbol{D}_2 & \vdash^* & \boldsymbol{D}'_2. \end{array}$$

For this, $\boldsymbol{w}\, \Lambda_1 \Lambda_2 \models \boldsymbol{W} \boldsymbol{D}_1 \boldsymbol{D}_2$ is clear.

Next, we show that $\boldsymbol{w}\, \Lambda_1\, \Lambda_2$ is a well-formed stack. From the well-formedness $\mathsf{WF}(\boldsymbol{w} \Lambda_1 \Lambda'_2)$, it is clear that $\Lambda_1(\dot{\mathtt{C}}) = 0.0$ and $\mathsf{WF}(\boldsymbol{w} \Lambda_1)$. Hence, it suffices to show $\Lambda_1 \precapprox \Lambda_2$. To show this, we use the above technical lemma (Lemma 5.11).

We extract some valuations from Λ_1, Λ_2, and Λ'_2 as follows:

- $\dot{\lambda}_1 : \dot{\mathcal{X}}_{\mathtt{C}} \to \mathbb{C}$ is the restricted function of Λ_1 to $\dot{\mathcal{X}}_{\mathtt{C}}$. Furthermore, we unmark the clocks of $\dot{\lambda}_1$ and obtain the valuation $\lambda_1 : \mathcal{X}_{\mathtt{C}} \to \mathbb{C}$.
- $\underset{\cdot}{\lambda}_2 : \underset{\cdot}{\mathcal{X}}_{\mathtt{C}} \to \mathbb{C}$ is the restricted function of Λ_2 to $\underset{\cdot}{\mathcal{X}}_{\mathtt{C}}$. We also unmark the clocks of $\underset{\cdot}{\lambda}_2$ and obtain the valuation $\lambda_2 : \mathcal{X}_{\mathtt{C}} \to \mathbb{C}$. $\underset{\cdot}{\lambda}'_2 : \underset{\cdot}{\mathcal{X}}_{\mathtt{C}} \to \mathbb{C}$ and $\lambda'_2 : \mathcal{X}_{\mathtt{C}} \to \mathbb{C}$ are defined in the same way from Λ'_2.

We also extract some valuations from $\boldsymbol{D}_1$, $\boldsymbol{D}_2$, and $\boldsymbol{D}'_2$ as follows:

- $\boldsymbol{d}_1$ is defined by restricting $\boldsymbol{D}_1$ to $\dot{\mathcal{X}}_{\mathtt{C}}$ and unmarking the clocks.
- $\boldsymbol{d}_2$ is defined by restricting $\boldsymbol{D}_2$ to $\underset{\cdot}{\mathcal{X}}_{\mathtt{C}}$ and unmarking the clocks. $\boldsymbol{d}'_2$ is defined in the same manner from $\boldsymbol{D}'_2$.

We show $\lambda_1 \precsim \lambda_2$ since it immediately implies $\Lambda_1 \precapprox \Lambda_2$. The above definition leads to the following diagram:

$$\begin{array}{ccccc} \lambda_1 & & \lambda_2 & & \lambda'_2 \\ \rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} \\ \boldsymbol{d}_1 & \vdash^* & \boldsymbol{d}_2 & \vdash^* & \boldsymbol{d}'_2 \end{array}$$

where $\lambda_1 \models \boldsymbol{d}_1$, $\lambda_2 \models \boldsymbol{d}_2$, and $\lambda'_2 \models \boldsymbol{d}'_2$ are derived from $\Lambda_1 \models \boldsymbol{D}_1$, $\Lambda_2 \models \boldsymbol{D}_2$, and $\Lambda'_2 \models \boldsymbol{D}'_2$, respectively. The relation $\Lambda_1 \precapprox \Lambda'_2$ implies $\lambda_1 \precsim \lambda_2$ and $\Lambda_2 \precsim \Lambda'_2$ implies $\lambda_2 \precsim \lambda'_2$. The well-formedness $\mathsf{WF}(\boldsymbol{W} \boldsymbol{D}_1 \boldsymbol{D}_2)$ implies $\dot{\mathtt{C}} \in_0 \boldsymbol{D}_1$ and $\mathtt{C} \in_0 \boldsymbol{d}_1$.

Now we can use Lemma 5.11 and obtain $\lambda_1 \precsim \lambda_2$ and it implies $\Lambda_1 \precapprox \Lambda_2$. □

5.6.6 Decidability of Reachability Problem

Combining the forward simulation between the standard and collapsed semantics (Lemma 5.8) and the forward/backward simulation between the collapsed and digitized semantics (Lemma 5.9 and 5.10), we can show our main theorem, Theorem 5.4. We introduce a notation for the stack correspondence between the standard and digitized semantics $w \models \boldsymbol{W}$ to formalize our main theorem.

Definition 5.29. Let w and $\boldsymbol{W}$ be stacks of the standard and digitized semantics, respectively. If there is a stack $\boldsymbol{w}$ of the collapsed semantics such that $w \models \boldsymbol{w}$ and $\boldsymbol{w} \models \boldsymbol{W}$, we write $w \models \boldsymbol{W}$. ■

Theorem 5.4. Let $\langle q, w\rangle$ be a configuration of the standard semantics. The following are equivalent:

(1) $\langle q_{\text{init}}, \langle \bot, \mathbf{0}_{\mathcal{X}}\rangle\rangle \xrightarrow[\textsc{Stnd}]{}^{*} \langle q, w\rangle$.

(2) $\langle q_{\text{init}}, \langle \bot, \boldsymbol{D}_0\rangle\rangle \xrightarrow[\textsc{Digi}]{}^{*} \langle q, \boldsymbol{W}\rangle$ for some stack $\boldsymbol{W}$ such that $\mathsf{WF}(\boldsymbol{W})$ and $w \models \boldsymbol{W}$ where

$$\boldsymbol{D}_0 = \{(\underset{\bullet}{x}_1, 0), (\dot{x}_1, 0), \ldots, (\underset{\bullet}{x}_n, 0), (\dot{x}_n, 0), (\underset{\bullet}{\mathtt{C}}, 0), (\dot{\mathtt{C}}, 0)\}_0.$$

Proof. The direction (1) ⇒ (2) is shown by using Lemma 5.8 and 5.9 sequentially.

We consider the other direction (2) ⇒ (1). From the assumption $w \models \boldsymbol{W}$, there is a stack $\boldsymbol{w}$ of the collapsed semantics such that $w \models \boldsymbol{w} \models \boldsymbol{W}$. The backward simulation lemma (Lemma 5.10) implies the following:

$$\begin{aligned}&\exists(\lambda_0 : \text{collapsed valuation}).\\ &\lambda_0 \models \boldsymbol{D}_0 \wedge \langle q_{\text{init}}, \langle \bot, \lambda_0\rangle\rangle \xrightarrow[\textsc{Coll}]{}^{*} \langle q, \boldsymbol{w}\rangle.\end{aligned}$$

Since $\mathbf{0}_{\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}}}$ is the only collapsed valuation that satisfies $\mathbf{0}_{\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}}} \models \boldsymbol{D}_0$, we also have the following:

$$\langle q_{\text{init}}, \langle \bot, \mathbf{0}_{\underset{\bullet}{\mathcal{X}}_{\mathtt{C}} \cup \dot{\mathcal{X}}_{\mathtt{C}}}\rangle\rangle \xrightarrow[\textsc{Coll}]{}^{*} \langle q, \boldsymbol{w}\rangle.$$

The forward simulation of the collapsed semantics by the standard semantics (Lemma 5.8) implies the following transition:

$$\langle q_{\text{init}}, \langle \bot, \mathbf{0}_{\mathcal{X}}\rangle\rangle \xrightarrow[\textsc{Stnd}]{}^{*} \langle q, w\rangle.$$

□

The above theorem allows to reduce the configuration reachability problem of the standard semantics to that of the digitized semantics. For a given configuration $\langle q, w\rangle$, there are finitely many well-formed stack $\boldsymbol{W}$ such that $w \models \boldsymbol{W}$ and we can enumerate all such stacks $\boldsymbol{W}$. This computability and the decidability of the configuration reachability problem of pushdown systems imply the decidability of the configuration reachability problem of SRTA.

Corollary 5.2. The configuration reachability problem of SRTA is decidable.

We also consider the location reachability problem and emptiness problem of SRTA:

- The location reachability problem decides, for a given location q, whether or not $\langle q_{\text{init}}, \langle \bot, \mathbf{0}\rangle\rangle \xrightarrow[\textsc{Stnd}]{}^{*} \langle q, {}^{\exists}w\rangle$ for some stack w.
- The emptiness problem decides, for a given SRTA A, whether or not $L(A) = \emptyset$.

These two decision problems are EXPTime-complete.

Corollary 5.3. The location reachability problem and emptiness problem of SRTA are EXPTIME-complete.

Proof. It is clear that the location reachability problem is polynomial-timed reducible to the emptiness problem and vice versa. Therefore, it suffices to show that the location reachability problem is in EXPTIME and the emptiness problem is EXPTIME-hard.

$\boxed{\text{The location reachability problem is in EXPTIME.}}$

Let $A = (Q, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta)$ be an SRTA and $\langle q_{\text{init}}, \langle \bot, \mathbf{0} \rangle \rangle \xrightarrow[\text{STND}]{}^{*} \langle q, {}^{\exists}w \rangle$ be an instance of the location reachability problem. On the basis of A, we build another SRTA $B = (Q \cup \{q'\}, q_{\text{init}}, F, \Sigma, \Gamma, \mathcal{X}, \Delta')$ where q' is a fresh location and

$$\Delta' \triangleq \Delta \cup \left\{ q \xrightarrow[\epsilon]{\textsf{nop}} q' : q \in Q, \gamma \in \Gamma \right\} \cup \left\{ q' \xrightarrow[\epsilon]{\textsf{pop}(\gamma)} q' : \gamma \in \Gamma \right\}.$$

(**nop** is a tautology like $\mathit{frac}(x) = \mathit{frac}(x)$ for some clock $x \in \mathcal{X}$)

From the construction of the SRTA B, the following holds:

$$\begin{aligned} & A : \langle q_{\text{init}}, \langle \bot, \mathbf{0} \rangle \rangle \xrightarrow[\text{STND}]{}^{*} \langle q, {}^{\exists}w \rangle \\ \iff & B : \langle q_{\text{init}}, \langle \bot, \mathbf{0} \rangle \rangle \xrightarrow[\text{STND}]{}^{*} \langle q, {}^{\exists}w \rangle \xrightarrow[\text{STND}]{}^{*} \langle q', \langle \bot, {}^{\exists}\nu \rangle \rangle \\ \iff & B : \langle q_{\text{init}}, \langle \bot, \boldsymbol{D}_0 \rangle \rangle \xrightarrow[\text{DIGI}]{}^{*} \langle q', \langle \bot, {}^{\exists}\boldsymbol{D} \rangle \rangle. \end{aligned}$$

Let $P = (Q \cup \{q'\}, \Gamma \times \mathbb{D}, \Delta'_{\text{DIGI}})$ be the PDS that corresponds to the digitized semantics of the SRTA B. We build another PDS $C = (Q \cup \{q', q''\}, \Gamma \times \mathbb{D}, \Delta'')$ where q'' is a fresh location and

$$\Delta'' \triangleq \Delta'_{\text{DIGI}} \cup \left\{ \langle q', \langle \bot, \boldsymbol{D} \rangle \rangle \hookrightarrow \langle q'', \epsilon \rangle : \boldsymbol{D} \in \mathbb{D} \right\}.$$

From the construction of the PDS C, the following holds:

$$B : \langle q_{\text{init}}, \langle \bot, \boldsymbol{D}_0 \rangle \rangle \xrightarrow[\text{DIGI}]{}^{*} \langle q', \langle \bot, {}^{\exists}\boldsymbol{D} \rangle \rangle \iff C : \langle q_{\text{init}}, \langle \bot, \boldsymbol{D}_0 \rangle \rangle \Rightarrow^{*} \langle q'', \epsilon \rangle.$$

Therefore, it suffices to solve the configuration reachability problem for the PDS C. Since the configuration reachability problem of pushdown system is in polynomial-time [BEM97, FWW97] and the size of the PDS C is exponential with respect to the size of the SRTA A, we can solve the location reachability problem of SRTA in exponential-time with respect to the size of an input SRTA.

$\boxed{\text{The emptiness problem is EXPTIME-hard.}}$

This is immediately shown by the following two results:

- The emptiness problem of PTA is EXPTIME-hard (Corollary 3.2 and [AAS12a]).
- For a given PTA A, we can build an SRTA B such that $L(A) = L(B)$ in polynomial-time with respect to the size of A (Theorem 5.1).

□

5.7 Regions vs Digital Valuations

We compare the two kinds of regions:

- the conventional region given by Alur and Dill in [AD94], which does not keep the fractional parts of the clocks that exceed a given upper bound M.

- the region given by Abdulla et al. in [AAS12a], which keeps the fractional parts of clocks even if their values are beyond M. To the authors' best knowledge, such a non-standard region first appeared at the work of Ouaknine and Worrell [OW04] to show the decidability of the language inclusion problem on one-clock timed automata.

The difference between keeping and not keeping the fractional parts of the clocks that exceed M is significant: on the region of Alur and Dill, the key lemma for the backward simulation of DIGI by COLL (Lemma 5.11) does not hold.

For the ease of comparison, we consider a slightly modified version of the definition given by Alur and Dill. The region of Alur and Dill can be understood through introducing the following collapsed domain:

$$\mathbb{C}' \triangleq \big([0..(\mathsf{M}-1)] \times [0,1)\big) \cup \{\infty\}\,.$$

For this collapsed domain, the new collapsing function $\mathcal{C}' : \mathbb{R}_{\geq 0} \to \mathbb{C}'$ is defined as follows:

$$\mathcal{C}'(r) = \begin{cases} (\lfloor r \rfloor, \mathit{frac}(r)) & \text{if } r < \mathsf{M} \\ \infty & \text{if } r \geq \mathsf{M} \end{cases}$$

We forget the fractional parts of the clocks that are equal to or greater than M.

Instead of giving the precise definition of the region for $\mathbb{C}'$, let us consider the following example under $\mathsf{M} = 1$:

1. The region $R = (\{\}_0 \{(x,0),(y,0)\}, \emptyset)$ means that the values of the two clocks x and y are the same. For example, the collapsed valuation $\rho = \{x \mapsto 0.4;\ y \mapsto 0.4\}$ realizes this region ($\rho \models R$).
2. The successor region R' of the region R ($R \vdash R'$) is $R' = (\{\}_0, \{x,y\})$ and R' means that the values of the two clocks x and y exceed the upper bound $\mathsf{M} = 1$. On the region R', there is no information that the fractional parts of two clocks are the same. The collapsed valuation $\rho' = \{x \mapsto \infty;\ y \mapsto \infty\}$ of the collapsed domain $\mathbb{C}'$ realizes this region ($\rho' \models R'$).

5.7.1 Key Lemma fails on Region of Alur and Dill

As we have seen in the previous section, the following property (Lemma 5.11) is key to establishing the backward simulation of DIGI by COLL:

$$\begin{array}{ccccc} \lambda_1 & & \lambda_2 & & \lambda_3 \\ \top\!\!\top & & \top\!\!\top & & \top\!\!\top \\ \boldsymbol{d}_1 & \vdash^* & \boldsymbol{d}_2 & \vdash^* & \boldsymbol{d}_3 \end{array} \wedge \left\{ \begin{array}{c} \lambda_1 \preccurlyeq \lambda_3, \\ \lambda_2 \preccurlyeq \lambda_3, \\ \exists x. x \in_0 \boldsymbol{d}_1 \end{array} \right\} \Longrightarrow \lambda_1 \preccurlyeq \lambda_2.$$

This property does not hold on the region of Alur and Dill. Let us consider the following collapsed valuations and regions on $\mathbb{C}'$ under $\mathsf{M} = 1$:

$$(1): \begin{array}{c} \rho_1 = \{x \mapsto 0.0;\ y \mapsto 0.5\} \\ \top\!\!\top \\ R_1 = \big(\{(x,0)\}_0 \{(y,0)\}, \emptyset\big), \end{array}$$

$$(2): \begin{array}{c} \rho_2 = \{x \mapsto 0.5;\ y \mapsto 0.9\} \\ \top\!\!\top \\ R_2 = \big(\{(x,0)\}_0 \{(y,0)\}, \emptyset\big), \end{array}$$

$$(3): \begin{array}{c} \rho_3 = \{x \mapsto \infty;\ y \mapsto \infty\} \\ \top\!\!\top \\ R_3 = \big(\{\}_0, \{x,y\}\big). \end{array}$$

Although these valuations and regions satisfy the assumption of the above property, $\rho_1 \preccurlyeq \rho_2$ does not hold. Since we forget the fractional parts of the clocks that exceed an upper-bound constant, the assumptions $\rho_1 \preccurlyeq \rho_3$ and $\rho_2 \preccurlyeq \rho_3$ do not help to relate the fractional parts of ρ_1 and ρ_2.

On the other hand, on the collapsed domain $\mathbb{C} = \Big(\big([0..(\mathsf{M}-1)] \cup \{\infty\}\big) \times [0,1)\Big)$ considered by Abdulla et al, there is no collapsed valuation λ_3 that satisfies both $\{x \mapsto 0.0;\ y \mapsto 0.5\} \preccurlyeq \lambda_3$ and $\{x \mapsto 0.5;\ y \mapsto 0.9\} \preccurlyeq \lambda_3$. This enables us to avoid the above problem and establish the key lemma for the backward simulation.

5.8 Related Work

Trivedi and Wojtczak introduced recursive timed automata (RTA) [TW10] and Benerecetti, Minopoli, and Peron introduced timed recursive state machines [BMP10, BP16] independently. The two models are essentially equivalent. Compared to SRTA, timed transitions of RTA increase only the top frame of a stack as follows:

$$\langle q, \langle \gamma_1, \nu_1 \rangle \dots \langle \gamma_n, \nu_n \rangle \rangle \overset{\delta}{\rightsquigarrow} \langle q, \langle \gamma_1, \nu_1 \rangle \dots \langle \gamma_n, \nu_n + \delta \rangle \rangle$$

where $\delta \in \mathbb{R}_{\geq 0}$. The difference of timed transitions between SRTA and RTA is crucial because RTA can simulate two-counter machines by using the timed transitions effectively and thus the reachability problem of RTA is undecidable [TW10, BMP10].

Li, Cai, Ogawa, and Yuen introduced nested timed automata (NeTA) in [LCOY13]. A configuration of NeTA is a stack of timed automata that are synchronously evolved by timed transitions. NeTA can be seen as SRTA without fractional constraints and value passing mechanisms between frames like the rule **dig**. Due to the absence of value passing mechanisms, we cannot keep values of clocks in the top frame when removing the top frame; indeed, the following transition that mixes the top and next to the top frames is not allowed:

$$\langle \gamma_1, \{x \mapsto 1.0; y \mapsto 2.0\} \rangle \langle \gamma_2, \{x \mapsto 3.5; y \mapsto 4.5\} \rangle \xrightarrow{\textsf{pop}(\gamma_2, \{x\})} \langle \gamma_1, \{x \mapsto 1.0;\ y \mapsto 4.5\} \rangle.$$

NeTA only removes the top frame as follows:

$$\langle \gamma_1, \{x \mapsto 1.0; y \mapsto 2.0\} \rangle \langle \gamma_2, \{x \mapsto 3.5; y \mapsto 4.5\} \rangle \xrightarrow{\textsf{pop}'(\gamma_2)} \langle \gamma_1, \{x \mapsto 1.0;\ y \mapsto 2.0\} \rangle.$$

Krishna, Manasa, and Trivedi introduced the subset of RTA called $\mathrm{RTA}_{\mathrm{RN}}$ (RTA only with pass-by-reference and non-passing) in [KMT15] and showed that the location reachability problem of $\mathrm{RTA}_{\mathrm{RN}}$ is decidable. $\mathrm{RTA}_{\mathrm{RN}}$ can be seen as SRTA without fractional constraints. Although their proof closely followed that of Abdulla et al. [AAS12a] and depended on the details of the construction of Abdulla et al., we adapt the construction of Abdulla et al. to simplify our proof.

5.9 Proof of Lemma 5.11

We show the following technical lemma:

$$\begin{array}{ccccc} \lambda_1 & & \lambda_2 & & \lambda_3 \\ \Vdash & & \Vdash & & \Vdash \\ \boldsymbol{d}_1 & \vdash^* & \boldsymbol{d}_2 & \vdash^* & \boldsymbol{d}_3 \end{array} \wedge \left\{ \begin{array}{c} \lambda_1 \preccurlyeq \lambda_3, \\ \lambda_2 \preccurlyeq \lambda_3, \\ \exists x. x \in_0 \boldsymbol{d}_1 \end{array} \right\} \implies \lambda_1 \preccurlyeq \lambda_2.$$

Before giving a proof, let us see that we cannot drop any condition.

If we drop the condition $\lambda_1 \preccurlyeq \lambda_3$ ...? Let us consider the following diagram:

$$\begin{array}{ccccc}
\left\{\begin{matrix} x \mapsto 0.0 \\ y \mapsto 0.5 \end{matrix}\right\} & & \left\{\begin{matrix} x \mapsto 0.2 \\ y \mapsto 0.3 \end{matrix}\right\} & & \left\{\begin{matrix} x \mapsto 0.4 \\ y \mapsto 0.5 \end{matrix}\right\} \\
\rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} \\
\{(x,0)\}_0 \{(y,0)\} & \vdash^* & \boldsymbol{d}_2 & \vdash^* & \boldsymbol{d}_3
\end{array}$$

where $\boldsymbol{d}_2 = \boldsymbol{d}_3 = \{\}_0 \{(x,0)\} \{(y,0)\}$. Although this case satisfies the assumptions except $\lambda_1 \preccurlyeq \lambda_3$, $\lambda_1 \preccurlyeq \lambda_2$ does not hold. For the same reason, we cannot remove the condition $\lambda_2 \preccurlyeq \lambda_3$.

If we drop the condition $\exists x. x \in_0 \boldsymbol{d}_1$ ...? Let us consider the following diagram:

$$\begin{array}{ccccc}
\{x \mapsto 0.5\} & & \{x \mapsto 0.3\} & & \{x \mapsto 0.5\} \\
\rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} \\
\{\}_0 \{(x,0)\} & \vdash^* & \{\}_0 \{(x,0)\} & \vdash^* & \{\}_0 \{(x,0)\}
\end{array}$$

Although this case satisfies the assumptions except $\exists x. x \in_0 \boldsymbol{d}_1$, $\lambda_1 \preccurlyeq \lambda_2$ does not hold. Therefore, we cannot drop the condition $\exists x. x \in_0 \boldsymbol{d}_1$.

Proof of Lemma 5.11 We prepare several notations to prove this lemma. Let λ and λ' be collapsed valuations on a finite clock set $\mathcal{X}$.

$$\begin{array}{lcl}
\mathit{frac}(\lambda)(x) & \triangleq & \mathit{frac}(\lambda(x)), \\
\lfloor \lambda \rfloor (x) & \triangleq & \lfloor \lambda(x) \rfloor, \\
\lambda \doteqdot \lambda' & \iff & \exists \delta \in \mathbb{R}_{\geq 0}.\ \mathit{frac}(\lambda + \delta) = \mathit{frac}(\lambda').
\end{array}$$

It is clear that the relation $\doteqdot$ is an equivalence relation.

Proposition 5.14. Let λ and λ' be collapsed valuations on a finite clock set $\mathcal{X}$.
If $\lambda \doteqdot \lambda'$ and $\exists x \in \mathcal{X}. \mathit{frac}(\lambda)(x) = \mathit{frac}(\lambda')(x)$, then $\mathit{frac}(\lambda) = \mathit{frac}(\lambda')$.

We show that related valuations λ and λ' with $\lambda \doteqdot \lambda'$ imply $\lambda = \lambda'$ under some constraints.

Proposition 5.15.

$$\begin{array}{ccc}
\lambda & \doteqdot & \lambda' \\
\rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} \\
\boldsymbol{d} & = & \boldsymbol{d}
\end{array} \wedge (\exists x. x \in_0 \boldsymbol{d}) \implies \lambda = \lambda'.$$

Proof. It is clear that $\lfloor \lambda \rfloor = \lfloor \lambda' \rfloor$ from $\lambda \models \boldsymbol{d}$ and $\lambda' \models \boldsymbol{d}$. Since there exists a clock x such that $x \in_0 \boldsymbol{d}$, we have $\mathit{frac}(\lambda(x)) = \mathit{frac}(\lambda'(x)) = 0.0$ from the assumption $\lambda \doteqdot \lambda'$. Proposition 5.14 implies $\mathit{frac}(\lambda) = \mathit{frac}(\lambda')$. This and $\lfloor \lambda \rfloor = \lfloor \lambda' \rfloor$ imply $\lambda = \lambda'$. □

Furthermore, since $\lambda \preccurlyeq \lambda'$ implies $\lambda \doteqdot \lambda'$, we have the following property.

Proposition 5.16.

$$\begin{array}{ccc}
\lambda & \preccurlyeq & \lambda' \\
\rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} \\
\boldsymbol{d} & = & \boldsymbol{d}
\end{array} \wedge\ \exists x. x \in_0 \boldsymbol{d} \implies \lambda = \lambda'.$$

Although $\lambda \doteqdot \lambda'$ does not imply $\lambda \preccurlyeq \lambda'$ in general, if there are digital valuations $\boldsymbol{d}$ and $\boldsymbol{d'}$ such that $\lambda \models \boldsymbol{d}$, $\lambda' \models \boldsymbol{d'}$, and $\boldsymbol{d} \vdash \boldsymbol{d'}$, then $\lambda \doteqdot \lambda'$ implies $\lambda \preccurlyeq \lambda'$.

Proposition 5.17.

$$\begin{array}{ccc}
\lambda & \doteqdot & \lambda' \\
\rotatebox[origin=c]{-90}{$\models$} & & \rotatebox[origin=c]{-90}{$\models$} \\
\boldsymbol{d} & \vdash & \boldsymbol{d'}
\end{array} \implies \lambda \preccurlyeq \lambda'.$$

Proof. We proceed by case analysis on $\boldsymbol{d} \vdash \boldsymbol{d'}$.

Case $\emptyset\, d_1 \cdots d_{n-1}\, d_n \vdash d'_n\, d_1 \cdots d_{n-1}$**:** Note there exists a clock x that belongs to d'_n and thus $x \in_0 \boldsymbol{d'}$. By the forward simulation of digital valuations by collapsed valuations (Proposition 5.11), we can find a collapsed valuation λ'' that satisfies the following diagram:

$$\begin{array}{ccc} \lambda & \preccurlyeq & \lambda'' \\ \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} \\ \boldsymbol{d} & \vdash & \boldsymbol{d'} \end{array}$$

Since $\lambda \preccurlyeq \lambda''$ implies $\lambda \div \lambda''$, we have $\lambda' \div \lambda''$. Using Proposition 5.15 along with $\lambda' \models \boldsymbol{d'}$, $\lambda'' \models \boldsymbol{d'}$, and $x \in_0 \boldsymbol{d'}$, we have $\lambda' = \lambda''$ and thus $\lambda \preccurlyeq \lambda'$.

Case $d_0\, d_1 \cdots d_n \vdash \emptyset\, d'_0\, d_1 \cdots d_n$**:** We can use the same argument as above. There exists a clock x that belongs to d_0 and thus $x \in_0 \boldsymbol{d}$. By the backward simulation of digital valuations by collapsed valuations (Proposition 5.12), we can find a collapsed valuation λ'' that satisfies the following diagram:

$$\begin{array}{ccc} \lambda'' & \preccurlyeq & \lambda' \\ \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} \\ \boldsymbol{d} & \vdash & \boldsymbol{d'} \end{array}$$

Since $\lambda'' \preccurlyeq \lambda'$ implies $\lambda' \div \lambda''$, we have $\lambda \div \lambda''$. Using Proposition 5.15 along with $\lambda \models \boldsymbol{d}$, $\lambda'' \models \boldsymbol{d}$, and $x \in_0 \boldsymbol{d}$, we have $\lambda = \lambda''$ and thus $\lambda \preccurlyeq \lambda'$.

□

We can easily generalize the above proposition to the following one.

Proposition 5.18.

$$\begin{array}{ccc} \lambda & \div & \lambda' \\ \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} \\ \boldsymbol{d} & \vdash^{+} & \boldsymbol{d'} \end{array} \implies \lambda \preccurlyeq \lambda'.$$

Finally, Proposition 5.16 and 5.18 imply our key technical lemma.

Lemma 5.11.

$$\begin{array}{ccccc} \lambda_1 & & \lambda_2 & & \lambda_3 \\ \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} \\ \boldsymbol{d}_1 & \vdash^{*} & \boldsymbol{d}_2 & \vdash^{*} & \boldsymbol{d}_3 \end{array} \wedge \left\{ \begin{array}{c} \lambda_1 \preccurlyeq \lambda_3, \\ \lambda_2 \preccurlyeq \lambda_3, \\ \exists x. x \in_0 \boldsymbol{d}_1 \end{array} \right\} \implies \lambda_1 \preccurlyeq \lambda_2.$$

Proof. From the assumption $\lambda_1 \preccurlyeq \lambda_3$ and $\lambda_2 \preccurlyeq \lambda_3$, $\lambda_1 \div \lambda_2 \div \lambda_3$ holds.

First, we consider the case $\boldsymbol{d}_1 = \boldsymbol{d}_2$. Since $\lambda_1 \div \lambda_2$, this case is immediate from Proposition 5.16.

Next, we consider the other case $\boldsymbol{d}_1 \neq \boldsymbol{d}_2$ (i.e., $\boldsymbol{d}_1 \vdash^{+} \boldsymbol{d}_2$). This case is immediate from Proposition 5.18. □

Chapter 6

Conclusion

Summary

In the present thesis, we have introduced the two classes of pushdown extensions of timed automata—*timed pushdown automata with multiple local clocks (MTPDA)* and *synchronized recursive timed automata (SRTA)*—by extending existing models of computation. We have not only introduced new classes but also shown new results on existing models, timed automata and pushdown timed automata:

1. On timed automata, we have shown that the configuration reachability problem, $\langle q_{\text{init}}, \mathbf{0}\rangle \Rightarrow_{?}^{*} \langle q, \nu\rangle$, is PSPACE-complete. Our decidability proof is based on the backward simulation between collapsed timed automata and digital automata. The decidability of the more general problem called the binary configuration reachability problem had been shown [CJ99, Dim02, Dan03, QSW17]. The binary configuration reachability problem decides, for a timed automaton and given two configurations $\boldsymbol{c}_{\text{start}}$ and $\boldsymbol{c}_{\text{goal}}$, whether or not there is a computation from $\boldsymbol{c}_{\text{start}} \Rightarrow_{?}^{*} \boldsymbol{c}_{\text{goal}}$. The known result about the complexity is that the binary configuration reachability problem is EXPTIME-hard. Our proof differs from the existing proofs and we show that the configuration reachability problem is a tractable case of the binary configuration reachability problem and PSPACE-complete.

2. On pushdown timed automata (PTA) of Bouajjani et al. [BER94], we have shown that the location reachability problem is EXPTIME-complete. The EXPTIME-hardness of the reachability problem of PTA refines the known complexity result that the reachability problem of dense-timed pushdown automata (DTPDA) of Abdulla et al. is EXPTIME-complete [AAS12a]; indeed, DTPDA are obtained by extending PTA.

For our two new models of computation, we have presented the following results:

1. The language classes of MTPDA and timed pushdown automata are the same even though MTPDA can be obtained by augmenting timed pushdown automata with multiple local clocks, the reset operation for local clocks $\mathsf{reset}(z)$, and diagonal constraints between a local clock and global clock $z - x \in_{?} I$.

2. The language classes of SRTA and timed pushdown automata are different due to the presence fractional constraints of SRTA.

3. The location and configuration reachability problems of SRTA are decidable and in EXPTIME-complete. Since the class of SRTA subsumes the class of MTPDA, this result means that the language class and the complexity class of the location reachability problem are the same between MTPDA and pushdown timed automata.

We have clarified that considering digital valuation is key to showing the decidability of the reachability problems of SRTA; indeed, the classical regions which are used to show the decidability of the location reachability problems of timed automata are insufficient to establish the important technical lemma (Lemma 5.11 of Chapter 5). We also have presented a simple and modular decidability proof of the reachability problems of SRTA. The most important technique is using a backward simulation rather than a forward simulation. This enables us to decompose the original decidability proof of Abdulla et al. based on an intricate hybrid simulation, forward-backward simulation, into a sequence of simple simulations along with a few intermediate semantics.

Further Directions

SRTA can be seen as MTPDA augmented with the following features (Theorem 5.3 of Chapter 5):

- fractional constraints;
- the ability to copy the value of a local clock z to a global clock x, $x \leftarrow z$, and the value of a global clock to a local clock, $z \leftarrow x$;
- bounded update of clocks: $x \leftarrow I$ where $I = (a : b)$ or $I = [a : b]$.

To show that SRTA are more expressive than pushdown timed automata, we use only fractional constraints to recognize the timed language L_{SRTA} of Chapter 5. Although that language cannot be recognized by any pushdown timed automaton, it can be accepted by 1-MTPDA with fractional constraints. Therefore, the following questions are natural:

- Is the class of MTPDA with the ability to copy a local clock to a global clock and vice versa more expressive than the class of MTPDA?
- Is the class of MTPDA with bounded update of local clocks more expressive than the class of MTPDA? Similarly, is the class of dense-timed pushdown automata, which does not allow diagonal constraints, with (unbounded) update of local clocks more expressive than the class of MTPDA?

On those classes, we can easily confirm that the monotonicity of a stack that is a key property to the untiming theorem of MTPDA does not hold. For the latter question, it seems difficult to apply the technique that removes updates from a given updatable timed automata while preserving its language. At the present moment, we have no clues to determine the expressiveness power of such new extensions; therefore, to solve the above question, we need to develop new techniques. We believe such techniques will further deepen and widen the theory of pushdown extensions of timed automata.

Bibliography

[AAS12a] Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman. Dense-timed pushdown automata. In *LICS '12: Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science*, pages 35–44. IEEE Computer Society, 2012.

[AAS12b] Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman. The minimal cost reachability problem in priced timed pushdown systems. In *LATA '12: Proceedings of the 6th International Conference on Language and Automata Theory and Applications*, volume 7183 of *Lecture Notes in Computer Science*, pages 58–69. Springer, 2012.

[AAS14a] Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman. Computing optimal reachability costs in priced dense-timed pushdown automata. In *LATA '14: Proceedings of the 8th International Conference on Language and Automata Theory and Applications*, volume 8370 of *Lecture Notes in Computer Science*, pages 62–75. Springer, 2014.

[AAS14b] Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman. Zenoness for timed pushdown automata. In *INFINITY '13: Proceedings of the 15th International Workshop on Verification of Infinite-State Systems*, volume 140 of *EPTCS*, pages 35–47, 2014.

[ACD93] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[AD90] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In *ICALP '90: Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[Aho68] Alfred V. Aho. Indexed grammars—an extension of context-free grammars. *Journal of the ACM*, 15(4):647–671, 1968.

[Aho69] Alfred V. Aho. Nested stack automata. *Journal of the ACM*, 16(3):383–406, 1969.

[AJKO97] Rajeev Alur, Lalita Jategaonkar Jagadeesan, Joseph J. Kott, and James Von Olnhausen. Model-checking of real-time systems: A Telecommunications Application (experience report). In *Pulling Together, Proceedings of the 19th International Conference on Software Engineering*, pages 514–524. ACM, 1997.

[BDFP00a] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Are timed automata updatable? In *CAV '00: Proceedings of the 12th International Conference on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 464–479. Springer, 2000.

[BDFP00b] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Expressiveness of updatable timed automata. In *MFCS '00: Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science*, volume 1893 of *Lecture Notes in Computer Science*, pages 232–242. Springer, 2000.

[BDFP04] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Updatable timed automata. *Theoretical Computer Science*, 321(2-3):291–345, 2004.

[BEM97] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR '97: Proceedings of the 8th International Conference on Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 1997.

[BER94] Ahmed Bouajjani, Rachid Echahed, and Riadh Robbana. On the automatic verification of systems with continuous variables and unbounded discrete data structures. In *Hybrid Systems II: Proceedings of the Third International Workshop on Hybrid Systems*, volume 999 of *Lecture Notes in Computer Science*, pages 64–85. Springer, 1994.

[BGK$^+$02] Johan Bengtsson, W. O. David Griffioen, Kåre J. Kristoffersen, Kim Guldstrand Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Automated verification of an audio-control protocol using UPPAAL. *The Journal of Logic and Algebraic Programming*, 52-53:163–181, 2002.

[BHR09] Patricia Bouyer, Serge Haddad, and Pierre-Alain Reynier. Undecidability results for timed automata with silent transitions. *Fundamenta Informaticae*, 92(1-2):1–25, 2009.

[BLR05] Patricia Bouyer, François Laroussinie, and Pierre-Alain Reynier. Diagonal constraints in timed automata: Forward analysis of timed systems. In *FORMATS '05: Proceedings of the Third International Conference on Formal Modeling and Analysis of Timed Systems*, volume 3829 of *Lecture Notes in Computer Science*, pages 112–126. Springer, 2005.

[BMP10] Massimo Benerecetti, Stefano Minopoli, and Adriano Peron. Analysis of timed recursive state machines. In *TIME '10: Proceedings of the 17th International Symposium on Temporal Representation and Reasoning*, pages 61–68. IEEE Computer Society, 2010.

[BP16] Massimo Benerecetti and Adriano Peron. Timed recursive state machines: Expressiveness and complexity. *Theoretical Computer Science*, 625:85–124, 2016.

[BPDG98] Béatrice Bérard, Antoine Petit, Volker Diekert, and Paul Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2-3):145–182, 1998.

[Cho59] Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137–167, 1959.

[CJ99] Hubert Comon and Yan Jurski. Timed automata and the theory of real numbers. In *CONCUR '99: Proceedings of the 10th International Conference on Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 242–257. Springer, 1999.

[CL15a] Lorenzo Clemente and Slawomir Lasota. Timed pushdown automata revisited. In *LICS' 15: Proceedings of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 738–749. IEEE Computer Society, 2015.

[CL15b] Lorenzo Clemente and Slawomir Lasota. Timed pushdown automata revisited. arXiv:abs/1503.02422 (`http://arxiv.org/abs/1503.02422`), 2015.

[Dam82] Werner Damm. The IO- and OI-hierarchies. *Theoretical Computer Science*, 20(2):95–207, 1982.

[Dan03] Zhe Dang. Pushdown timed automata: a binary reachability characterization and safety verification. *Theoretical Computer Science*, 302(1-3):93–121, 2003.

[Den16] Tobias Denkinger. An automata characterisation for multiple context-free languages. In *DLT '16: Proceedings of the 20th International Conference on Developments in Language Theory*, volume 9840 of *Lecture Notes in Computer Science*, pages 138–150. Springer, 2016.

[Dim02] Catalin Dima. Computing reachability relations in timed automata. In *LICS '02: Proceedings of the 17th IEEE Symposium on Logic in Computer Science*, page 177. IEEE Computer Society, 2002.

[DKN04] Conrado Daws, Marta Z. Kwiatkowska, and Gethin Norman. Automatic verification of the IEEE 1394 root contention protocol with KRONOS and PRISM. *International Journal on Software Tools for Technology Transfer*, 5(2-3):221–236, 2004.

[Eng91] Joost Engelfriet. Iterated stack automata and complexity classes. *Information and Computation*, 95(1):21–75, 1991.

[Eve63] R. James Evey. Application of pushdown-store machines. In *AFIPS '63: Proceedings of the 1963 Fall Joint Computer Conference*, pages 215–227. ACM, 1963.

[FJ15] John Fearnley and Marcin Jurdziński. Reachability in two-clock timed automata is PSPACE-complete. *Information and Computation*, 243:26–36, 2015.

[FWW97] Alain Finkel, Bernard Willems, and Pierre Wolper. A direct symbolic approach to model checking pushdown systems. *ENTCS*, 9:27–37, 1997.

[HKW95] Thomas A. Henzinger, Peter W. Kopke, and Howard Wong-Toi. The expressive power of clocks. In *ICALP '95: Proceedings of the 22nd International Colloquium on Automata, Languages and Programming*, volume 944 of *Lecture Notes in Computer Science*, pages 417–428. Springer, 1995.

[HLMS12] Alexander Heußner, Jérôme Leroux, Anca Muscholl, and Grégoire Sutre. Reachability analysis of communicating pushdown systems. *Logical Methods in Computer Science*, 8(3), 2012.

[HMOS08] Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. In *LICS '08: Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science*, pages 452–461. IEEE Computer Society, 2008.

[HMOS17] Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. *ACM Transactions on Computational Logic*, 18(3):25:1–25:42, 2017.

[HO08] Matthew Hague and C.-H. Luke Ong. Symbolic backwards-reachability analysis for higher-order pushdown systems. *Logical Methods in Computer Science*, 4(4), 2008.

[HU79] John Hopcroft and Jeffrey Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[JLT75] Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163, 1975.

[KMT15] Shankara Narayanan Krishna, Lakshmi Manasa, and Ashutosh Trivedi. What's decidable about recursive hybrid automata? In *HSCC '15: Proceedings of the 18th International Conference on Hybrid Systems*, pages 31–40. ACM, 2015.

[Koz77] Dexter Kozen. Lower bounds for natural proof systems. In *FoCS '77: Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 254–266. IEEE Computer Society, 1977.

[LCOY13] Guoqiang Li, Xiaojuan Cai, Mizuhito Ogawa, and Shoji Yuen. Nested timed automata. In *FORMATS '13: Proceedings of the 11th International Conference on Formal Modeling and Analysis of Timed Systems*, volume 8053 of *Lecture Notes in Computer Science*, pages 168–182. Springer, 2013.

[LMP07] Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. A robust class of context-sensitive languages. In *LICS '07: Proceedings of the 22nd IEEE Symposium on Logic in Computer Science*, pages 161–170. IEEE Computer Society, 2007.

[LMS04] François Laroussinie, Nicolas Markey, and Philippe Schnoebelen. Model checking timed automata with one or two clocks. In *CONCUR '04: Proceedings of the 15th International Conference on Concurrency Theory*, volume 3170 of *Lecture Notes in Computer Science*, pages 387–401. Springer, 2004.

[LST15] Jérôme Leroux, Grégoire Sutre, and Patrick Totzke. On the coverability problem for pushdown vector addition systems in one dimension. In *ICALP '15: Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming*, volume 9135 of *Lecture Notes in Computer Science*, pages 324–336. Springer, 2015.

[LV95] Nancy A. Lynch and Frits W. Vaandrager. Forward and backward simulations: I: untimed systems. *Information and Computation*, 121(2):214–233, 1995.

[LV96] Nancy A. Lynch and Frits W. Vaandrager. Forward and backward simulations, II: timing-based systems. *Information and Computation*, 128(1):1–25, 1996.

[Mas74] A. N. Maslov. The hierarchy of indexed languages of an arbitrary level. *Soviet Math. Dokl.*, 15:1170–1174, 1974.

[Mas76] A. N. Maslov. Multilevel stack automata. *Problems of Information Transmission*, 12:38–43, 1976.

[Min61] Marvin L. Minsky. Recursive unsolvability of post's problem of "tag" and other topics in theory of turing machines. *Annals of Mathematics*, 74(3):437–455, 1961.

[Min67] Marvin L. Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc., 1967.

[MP11] P. Madhusudan and Gennaro Parlato. The tree width of auxiliary storage. In *POPL '11: Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 283–294. ACM, 2011.

[NP05] V. Krishna Nandivada and Jens Palsberg. Timing analysis of TCP servers for surviving denial-of-service attacks. In *RTAS '05: Proceedings of the 11th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 541–549. IEEE Computer Society, 2005.

[OW04] Joël Ouaknine and James Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *LICS '04: Proceedings of the 19th IEEE Symposium on Logic in Computer Science*, pages 54–63. IEEE Computer Society, 2004.

[PS12] Paritosh K. Pandya and P. Vijay Suman. An introduction to timed automata. In *Modern Applications of Automata Theory*, pages 111–146. World Scientific, 2012.

[QSW17] Karin Quaas, Mahsa Shirmohammadi, and James Worrell. Revisiting reachability in timed automata. In *LICS '17: Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 1–12. IEEE Computer Society, 2017.

[Sch63] M.P. Schützenberger. On context-free languages and push-down automata. *Information and Control*, 6(3):246–264, 1963.

[SMFK91] Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229, 1991.

[TW10] Ashutosh Trivedi and Dominik Wojtczak. Recursive timed automata. In *ATVA '10: Proceedings of the 8th International Symposium on Automated Technology for Verification and Analysis*, volume 6252 of *Lecture Notes in Computer Science*, pages 306–324. Springer, 2010.

[Uez18] Yuya Uezato. Dense-timed pushdown automata with multiple-local clocks (in Japanese, 複数の局所クロックを持つ時間プッシュダウン・オートマトン). *IPSJ Transactions on Programming*, To appear in 2018.

[UM15] Yuya Uezato and Yasuhiko Minamide. Synchronized recursive timed automata. In *LPAR-20: Proceedings of the 20th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 9450 of *Lecture Notes in Computer Science*, pages 249–265. Springer, 2015.

[UM18] Yuya Uezato and Yasuhiko Minamide. Configuration reachability analysis of synchronized recursive timed automata. *Computer Software*, To appear in 2018.

[Vij87] K. Vijayashanker. *A Study of Tree Adjoining Grammars.* PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 1987.