# Computing Optimal Reachability Costs in Priced Dense-Timed Pushdown Automata

Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman

Dept. of Information Technology, Uppsala University, Sweden
{parosh.abdulla, mohamed_faouzi.atig, jari.stenman}@it.uu.se

**Abstract.** We study priced dense-timed pushdown automata that are a generalization of the classic model of pushdown automata, in the sense that they operate on real-valued clocks, and that the stack symbols have real-valued ages. Furthermore, the model allows a cost function that assigns transition costs to transitions and storage costs to stack symbols. We show that the optimal cost, i.e., the infimum of the costs of the set of runs reaching a given control state, is computable.

## 1 Introduction

Pushdown automata are a widely used model both in language theory and program verification. Recently, several models have been introduced that extend pushdown automata with clocks and real-time constraints [10, 12, 1]. In the mean time, several works have extended the model of timed automata [5] with *prices* (*weights*) (e.g., [6, 7, 9]). Weighted timed automata are used in the modeling of embedded systems, where the behavior of the system is usually constrained by the availability of different types of resources.

In this paper, we consider *Priced Dense-Timed Pushdown Automata* (PTPA) that subsume all the above models. PTPA are a generalization of classic pushdown automata with real-valued clocks, timed constraints, and prices for computations. More precisely, a PTPA contains a finite set of global clocks, and each symbol in the stack is equipped with a real number indicating its age. The global clocks admit the same kind of operations as in timed automata, and timed transitions increase the clock values and the ages of stack symbols at the same rate. Pop operations may only be performed if the age of the topmost stack symbol is within a given time interval. Furthermore, the model is priced in the sense that there is a cost function that assigns transition costs to transitions and storage costs to stack symbols.

We study the problem of computing the optimal cost to reach a given control state. In general, a cost-optimal computation may not exist (e.g., even in priced timed automata it can happen that there is no computation of cost 0, but there exist computations of cost $\leq \epsilon$ for every $\epsilon > 0$). However, we show that the infimum of the costs is computable. To do this, we perform a sequence of reductions that ultimately translates the problem to the problem of control state reachability for plain (unpriced and untimed) pushdown automata. The latter problem is known to be decidable [8].

*Related Work.* Priced extensions of timed models have been studied in the literature. The paper [4] studies a priced dense-timed extension of Petri nets, where the optimal cost is computed for satisfying coverability objectives (reaching an upward closed set of markings). Proofs for solving the coverability problem in Petri nets are in general quite different from those for the solving control state reachability problem in pushdown systems. This is already the case for the unpriced untimed case, where the former relies on Karp-Miller constructions [11] or backward reachability analysis [3], while the latter uses finite automata constructions [8]. This difference is also reflected in the priced timed case. In particular, [4] (using backward reachability analysis) reduces optimal cost computation to the reachability problem for a more powerful model than plain Petri nets, namely that of Petri nets with one inhibitor arc. In our case, we reduce the problem to the plain pushdown model.

Several timed extensions of pushdown automata have been considered [12, 10, 1]. Since our model extends these, some of the techniques need to be reused. However, priced timed models are nontrivial extensions of (unpriced) timed models. Here, in a similar manner to priced extensions of timed Petri nets [4] and timed automata [9], we need to reason about special forms of computations, and a nontrivial modification of the (region-based) symbolic encoding is also necessary to represent the infinite state space.

In [2] we study priced *discrete-timed* pushdown automata. In the discrete-time case, time is interpreted as being incremented in discrete steps and thus the clock values and ages of stack symbols are in a countable domain. The method of [2] cannot be extended to the dense time case. It is well-known that, in timed models, using discrete domains represents a substantial simplification compared to using dense time. In particular, the absence of fractional parts in clock values and stack symbol ages leads to a much simpler symbolic representation of the stack. The model of priced discrete-timed pushdown automata is generalized in [13], where the authors consider pushdown systems that can modify the whole stack using transducers.

## 2 Preliminaries

We use $\mathbb{R}^{\geq 0}$ to denote the non-negative reals. For $r \in \mathbb{R}^{\geq 0}$, where $r = n + r'$ for $n \in \mathbb{N}$, $r' \in \mathbb{R}^{\geq 0}$ and $r' < 1$, we let $\lfloor r \rfloor = n$ denote the *integral part* and $\mathit{fract}\,(r) = r'$ denote the *fractional part* of $r$. Given a set $A$, we use $2^A$ for the powerset of $A$. For sets $A$ and $B$, $f : A \to B$ denotes a (possibly partial) function from $A$ to $B$. If $f$ is undefined at $a$, we write $f(a) = \bot$. We use $\mathit{dom}\,(f)$ and $\mathit{range}\,(f)$ to denote the domain and range of $f$. Given a function $f$ and a set $A$, we use $f(A)$ for the image $\{f(x) \,|\, x \in A\}$ of $A$ under $f$. The image $\mathit{img}(f)$ of $f$ is then defined as $f(\mathit{dom}\,(f))$. We write $f[a \leftarrow b]$ to denote the function $f'$ such that $f'(a) = b$ and $f'(x) = f(x)$ for $x \neq a$. Given a function $f : A \times B \to C$, we sometimes write $f(a)(b) = c$ to make explicit that $f$ might be applied partially, i.e. to get a function $f(a) : B \to C$. The set of *intervals* of the form $[a : b]$, $(a : b]$, $[a : b)$,$(a : b)$,$[a : \infty)$ or $(a : \infty)$, where $a, b \in \mathbb{N}$, is denoted by $\mathcal{I}$. Given a set $A$, we use $\inf(A)$, $\max(A)$ and $\min(A)$ to denote the infimum, the maximum and the minimum of $A$, respectively.

Let $A$ be an alphabet. We denote by $A^*$, (resp. $A^+$) the set of all *words* (resp. non-empty words) over $A$. The empty word is denoted by $\epsilon$. For a word $w$, $|w|$ denotes the length of $w$ (we have $|\epsilon| = 0$). For words $w_1, w_2$, we use $w_1 \cdot w_2$ for the concatenation of $w_1$ and $w_2$. We extend $\cdot$ to sets $W_1, W_2$ of words by defining $W_1 \cdot W_2 = \{w_1 \cdot w_2 \mid w_1 \in W_1, w_2 \in W_2\}$. For a word $w = a_0 \dots a_n$, and $i \in \{0, \dots, n\}$, we let $w[i]$ denote $a_i$. Given a word $w = \langle x_0, y_0\rangle \dots \langle x_n, y_n\rangle \in (X \times Y)^*$, we define the *first projection* $\mathit{proj}_1(t) = x_0 \dots x_n$ and the *second projection* $\mathit{proj}_2(t) = y_0 \dots y_n$. We define a binary *shuffle operation* $\otimes$ inductively: For $w \in (2^A)^*$, define $w \otimes \epsilon = \epsilon \otimes w = w$. For sets $r_1, r_2 \in 2^A$ and words $w_1, w_2 \in (2^A)^*$, define $(r_1 \cdot w_1) \otimes (r_2 \cdot w_2) = (r_1 \cdot (w_1 \otimes (r_2 \cdot w_2))) \cup (r_2 \cdot ((r_1 \cdot w_1) \otimes w_2))) \cup ((r_1 \cup r_2) \cdot (w_1 \otimes w_2))$.

## 3 Priced Timed Pushdown Automata

In this section, we introduce Ptpa and define cost-optimal reachability.

*Model.* Formally, a Ptpa is a tuple $\mathcal{T} = \langle Q, q_{init}, \Gamma, X, \Delta, \mathit{Cost}\rangle$, where $Q$ is a finite set of states, $q_{init} \in Q$ is the initial state, $\Gamma$ is a finite stack alphabet, $X$ is a finite set of clocks, and $\mathit{Cost} : (\Gamma \cup \Delta) \to \mathbb{N}$ is a function assigning transition costs to transition rules and storage costs to stack symbols. The set $\Delta$ consists of a finite number of transition rules of the form $\langle q, op, q'\rangle$, where $q, q' \in Q$ and $op$ is either (i) *nop*, an operation that does not modify the clocks or the stack, (ii) $\mathit{push}(a)$, where $a \in \Gamma$, which pushes $a$ onto the stack with initial age 0, (iii) $\mathit{pop}(a, I)$, where $a \in \Gamma$ and $I \in \mathcal{I}$, which pops $a$ of the stack if its age is in $I$, (iv) $\mathit{test}(x, I)$, where $x \in X$ and $I \in \mathcal{I}$, which is only enabled if $x \in I$, or (v) $\mathit{reset}(x)$, where $x \in X$, which sets the value of the clock $x$ to 0.

*Semantics.* A *clock valuation* is a function $\mathtt{X} : X \to \mathbb{R}^{\geq 0}$ which assigns a concrete value to each clock. A *stack content* is a word $w \in (\Gamma \times \mathbb{R}^{\geq 0})^*$, i.e. a sequence of stack symbols and their corresponding ages. A *configuration* is a tuple $\langle q, \mathtt{X}, w\rangle$, where $q \in Q$ is a state, $\mathtt{X}$ is a clock valuation, and $w$ is a stack content. For a configuration $\gamma = \langle q, \mathtt{X}, w\rangle$, define the functions $\mathtt{State}(\gamma) = q$, $\mathtt{ClockVal}(\gamma) = \mathtt{X}$, and $\mathtt{Stack}(\gamma) = w$. For any transition rule $t = \langle q, op, q'\rangle \in \Delta$, define $\mathtt{Op}(t) = op$. Given a Ptpa $\mathcal{T}$, we use $\mathit{Conf}(\mathcal{T})$ to denote the set of all configurations of $\mathcal{T}$. Let $\mathtt{X}_{init}$ be the clock valuation such that $\mathtt{X}_{init}(x) = 0$ for each $x \in X$. The *initial configuration* $\gamma_{init}$ is the configuration $\langle q_{init}, \mathtt{X}_{init}, \epsilon\rangle$. The operational semantics of a Ptpa $\mathcal{T}$ are defined by a transition relation over the set of configurations $\mathit{Conf}(\mathcal{T})$. It consists of two types of transitions; *timed* transitions, which simulate time passing, and *discrete* transitions, which are applications of the transition rules in $\Delta$.

*Timed Transitions.* Fix some $r \in \mathbb{R}^{\geq 0}$. Given a clock valuation $\mathtt{X}$, let $\mathtt{X}^{+r}$ be the function defined by $\mathtt{X}^{+r}(x) = \mathtt{X}(x) + r$ for all $x \in X$. For any stack content $w = \langle a_0, v_0\rangle \cdots \langle a_n, v_n\rangle$, let $w^{+r}$ be the stack content $\langle a_0, v_0 + r\rangle \cdots \langle a_n, v_n + r\rangle$.

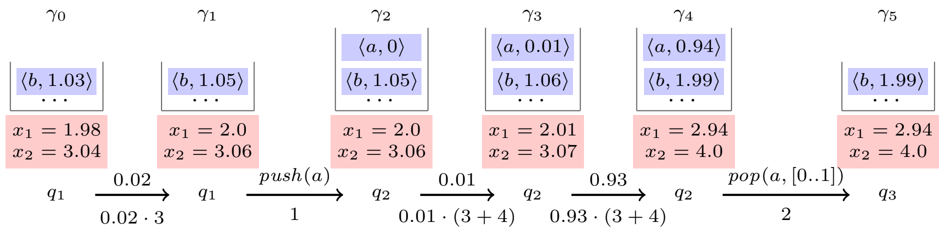

**Fig. 1.** A fragment of a computation

Given configurations $\gamma = \langle q, \mathtt{X}, w\rangle$ and $\gamma' = \langle q', \mathtt{X}', w'\rangle$, we have $\gamma \xrightarrow{r} \gamma'$ if $q' = q$, $\mathtt{X}' = \mathtt{X}^{+r}$ and $w' = w^{+r}$. This means that a PTPA may perform a timed transition, whereby it advances all clocks and ages of stack symbols by some non-negative real number.

*Discrete Transitions.* Let $t = \langle q, op, q'\rangle \in \Delta$ be a transition rule. For configurations $\gamma = \langle q, \mathtt{X}, w\rangle$ and $\gamma' = \langle q', \mathtt{X}', w'\rangle$, we have $\gamma \xrightarrow{t} \gamma'$ if either (i) $op = nop$, $w' = w$, and $\mathtt{X}' = \mathtt{X}$, (ii) $op = push(a)$, $w' = w \cdot \langle a, 0\rangle$, and $\mathtt{X}' = \mathtt{X}$, (iii) $op = pop(a, I)$, $w = w' \cdot \langle a, v\rangle$ for some $v \in I$, and $\mathtt{X}' = \mathtt{X}$, (iv) $op = test(x, I)$, $w' = w$, $\mathtt{X}' = \mathtt{X}$, and $\mathtt{X}(x) \in I$, or (v) $op = reset(x)$, $w' = w$, and $\mathtt{X}' = \mathtt{X}[x \leftarrow 0]$.

A *computation* $\pi$ to a configuration $\gamma$ of a PTPA $\mathcal{T}$ is a finite sequence of the form $\gamma_0 \xrightarrow{t_1} \gamma_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \gamma_n$, where $\gamma_0 = \gamma_{init}$, $\gamma_n = \gamma$, and for all $1 \le i \le n$, $\gamma_i \in Conf(\mathcal{T})$, and either $t_i \in \Delta$ or $t_i \in \mathbb{R}^{\ge 0}$. We define $Comp(\mathcal{T}, q)$ to be the set of computations to a configuration $\gamma$ such that $\mathtt{State}\,(\gamma) = q$. If $\pi$ is a computation to $\gamma$, and $\mathtt{State}\,(\gamma) = q$, we say that $\pi$ is a computation to $q$.

*Cost of Computation.* We will now extend the cost function *Cost*, which we originally defined on stack symbols and transition rules, to transitions and computations. The cost of a discrete transition is given by the cost of the corresponding transition rule, and the cost of a timed transition is the total cost of the stack scaled by the length of the timed transition. Fix a discrete or timed transition $\gamma \xrightarrow{t} \gamma'$, and let $\gamma = \langle q, \mathtt{X}, \langle a_0, v_0\rangle \cdots \langle a_n, v_n\rangle\rangle$. Formally, $Cost(\gamma \xrightarrow{t} \gamma') = Cost(t)$ if $\gamma \xrightarrow{t} \gamma'$ is discrete, and $Cost(\gamma \xrightarrow{t} \gamma') = t \cdot \sum_{i=0}^{n} Cost(a_i)$ if $\gamma \xrightarrow{t} \gamma'$ is timed. The cost of a computation $\pi = \gamma_0 \xrightarrow{t_1} \gamma_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \gamma_n$ is defined as the sum of the costs of its transitions, i.e. $Cost(\pi) = \sum_{i=1}^{n} Cost(\gamma_{i-1} \xrightarrow{t_i} \gamma_i)$. Fig. 1 shows a fragment of a computation of a PTPA where the set of clocks is $\{x_1, x_2\}$ and the stack alphabet is $\{a, b, c\}$ (the stack not shown is filled with $c$). We assume that the storage costs of $a$, $b$ and $c$ are 4, 3 and 0, respectively, that the transition cost of $\langle q_1, push(a), q_2\rangle$ is 1, and that the transition cost of $\langle q_2, pop(a, [0:1]), q_3\rangle$ is 2. The accumulated cost for the example is 9.64.

*Cost-Optimality.* In this paper, we address the problem of computing the optimal cost required to reach a certain state. The optimal cost is defined as the *infimum* of the set of reachability costs. Formally, we define the optimal cost $Cost_{opt}(\mathcal{T}, q)$ as $Cost_{opt}(\mathcal{T}, q) = \inf\{Cost(\pi) \mid \pi \in Comp(\mathcal{T}, q)\}$ if

$\mathit{Comp}(\mathcal{T}, q) \neq \emptyset$ and $\mathit{Cost}_{opt}(\mathcal{T}, q) = \infty$ otherwise. The *cost-optimal reachability problem* is then, given a PTPA $\mathcal{T}$ and a state $q$, to compute the optimal cost $\mathit{Cost}_{opt}(\mathcal{T}, q)$.

**Theorem 1.** *Let $\mathcal{T}$ be* PTPA *and $q$ be a state. Then $\mathit{Cost}_{opt}(\mathcal{T}, q)$ is computable.*

The rest of the paper is devoted to the proof of the above theorem. In Sec. 4, we show that it is sufficient to consider computations in a certain form in order to compute the optimal cost. In Sec. 5, we introduce our symbolic automata (PPA) to which we reduce the cost-optimal reachability problem for PTPA. In Sec. 6, we formally define the region encoding and some related operations. Finally, we construct a PPA that simulates all the computations (in a certain form) of PTPA in Sec. 7.

## 4 Forms of Computations

*Detailed Computations.* In order to solve the cost-optimal reachability problem, we will only consider computations that are of a certain form, called *detailed.* This yields a closer correspondence between computations of PTPA and computations in the untimed automaton, defined in section 7. Consider a timed transition $\langle q, \mathtt{X}, \langle a_0, v_0\rangle \cdots \langle a_n, v_n\rangle\rangle \xrightarrow{r} \langle q', \mathtt{X}', \langle a_0, v_0'\rangle \cdots \langle a_n, v_n'\rangle\rangle$. Let $V = \mathit{fract}\,(\mathtt{X}(X)) \cup \mathit{fract}\,(\{v_0, \ldots, v_n\})$ and let $m = \min(V)$ and $d = \max(V)$. In other words, $m$ is the minimal fractional part and $d$ is the maximal fractional part of any value. We can classify the timed transition into two different types:

- *Type* 1: A transition which is taken when no value has fractional part 0, and which may make the values with the largest fractional parts reach the next integer. This is the case when $r > 0$, $m > 0$ and $r \leq 1 - d$.
- *Type* 2: A transition which is taken when some values have fractional parts 0, which makes the fractional parts of those value positive, and which does not change the integral part of any value. This is the case when $r > 0$, $m = 0$ and $r < 1 - d$.

A detailed computation is a computation where all timed transitions are of either type 1 or 2. We use $\mathit{Comp}^{\mathsf{d}}(\mathcal{T}, q)$ to denote all the detailed computations in $\mathit{Comp}(\mathcal{T}, q)$. Since the cost function is linear, considering only detailed computations is not a restriction. Formally, we have:

**Lemma 2.** $\forall.\, \pi \in \mathit{Comp}(\mathcal{T}, q)$*,* $\exists.\, \pi' \in \mathit{Comp}^{\mathsf{d}}(\mathcal{T}, q)$ *s.t.* $\mathit{Cost}(\pi') = \mathit{Cost}(\pi)$*.*

*Computations in $\delta$-form.* A computation is in *$\delta$-form* if the values of all clocks and symbol ages along the computation are strictly within $\delta$ of an integer. Formally, given some $\delta : 0 < \delta < \frac{1}{10}$, we say that a configuration $\langle q, \mathtt{X}, \langle a_0, v_0\rangle \cdots \langle a_m, v_m\rangle\rangle$ is in $\delta$-form if (i) for all $x \in X$, $\mathit{fract}\,(\mathtt{X}(x)) < \delta$ or $\mathit{fract}\,(\mathtt{X}(x)) > 1 - \delta$, and (ii) for all $i : 1 \leq i \leq m$, $\mathit{fract}\,(v_i) < \delta$ or $\mathit{fract}\,(v_i) > 1 - \delta$. A computation $\pi = \gamma_0 \xrightarrow{t_1} \gamma_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \gamma_n$ is in $\delta$-form if for each $i : 1 \leq i \leq n$, the configuration $\gamma_i$ is in $\delta$-form. Note that we need an upper bound (e.g. $\frac{1}{10}$) on $\delta$ to ensure that short and long timed transitions are not mixed. We use $\mathit{Comp}^{\mathsf{d}}_{\delta}(\mathcal{T}, q)$ to denote all the computations in $\mathit{Comp}^{\mathsf{d}}(\mathcal{T}, q)$ in $\delta$-form.

We use established linear programming techniques, first used for priced timed automata [9] and later for priced timed Petri nets [4], to show that the feasible delays for a set of computations with fixed "structure", i.e. discrete transitions, form a polyhedron with integral vertices. Since the cost-function is linear, its extreme values in the vertices of the polyhedron, which means that the optimal cost is a natural number.

**Lemma 3.** *For any* Ptpa $\mathcal{T}$ *and state* $q$ *of* $\mathcal{T}$, *we have* $\mathit{Cost}_{opt}(\mathcal{T}, q) \in \mathbb{N}$.

Due to strict inequalities in the structure of the computation, the exact delays represented by the vertex which represents the optimal cost might not be feasible, but by choosing the delays arbitrarily close to the vertex, we can get arbitrarily close to the optimal cost.

**Lemma 4.** *For every* $\pi \in \mathit{Comp}^{\mathsf{d}}(\mathcal{T}, q)$ *and* $\delta : 0 < \delta < \frac{1}{10}$, *there is* $\pi' \in \mathit{Comp}^{\mathsf{d}}_{\delta}(\mathcal{T}, q)$ *such that* $\mathit{Cost}(\pi') \leq \mathit{Cost}(\pi)$.

From Lemma 4 it follows that in order to find the optimal cost, we only need to consider computations in $\delta$-form for arbitrarily small $\delta$, i.e.

**Lemma 5.** $\forall \delta : 0 < \delta < \frac{1}{10}, \mathit{Cost}_{opt}(\mathcal{T}, q) = \inf\{\mathit{Cost}(\pi) \mid \pi \in \mathit{Comp}^{\mathsf{d}}_{\delta}(\mathcal{T}, q)\}$.

The fact that a computation is detailed and in $\delta$-form implies that the length of any timed transition is either in $(0 : \delta)$ (a *short* timed transition) or in $(1-\delta : 1)$ (a *long* timed transition). For example, if $\delta = \frac{1}{11}$, the timed transition between $\gamma_3$ and $\gamma_4$ in Fig. 1 is long, while the other two are short. We use this to construct a Ppa (the *symbolic* automaton), which simulates Ptpa computations that are detailed and in $\delta$-form for arbitrarily small $\delta$. Since $\delta$ is arbitrarily small, the model simulates long timed transitions with length arbitrarily close to 1, and short timed transition with length arbitrarily close to 0. Therefore, in the discrete model, we pay the full stack cost for long timed transitions and nothing for short timed transitions. The cost for simulating the computation in Fig. 1 would be 10, i.e. slightly higher than the real cost of 9.64. However, the difference between the real and symbolic computations can be made arbitrarily small by decreasing $\delta$. We build on techniques from [1] to handle the simulation of the timed part, and extend the concepts with the necessary information to handle costs.

## 5 Priced Pushdown Automata

A *priced pushdown automaton* (Ppa) $\mathcal{P}$ is a tuple $\langle Q, q_{init}, \Gamma, \Delta_1, \Delta_2, \mathit{Cost}\rangle$, where $Q$ is a finite set of states, $q_{init} \in Q$ is an initial state, $\Gamma$ is a finite stack alphabet, $\Delta_1$ and $\Delta_2$ are finite sets of transition rules, and $\mathit{Cost} : (\Delta_1 \cup \Gamma) \to \mathbb{N}$ is a cost function assigning costs to both transition rules in $\Delta_1$ and stack symbols in $\Gamma$. Intuitively, we separate the transition rules into two different sets. When we perform transitions in $\Delta_1$, we pay the transition cost, and when we perform

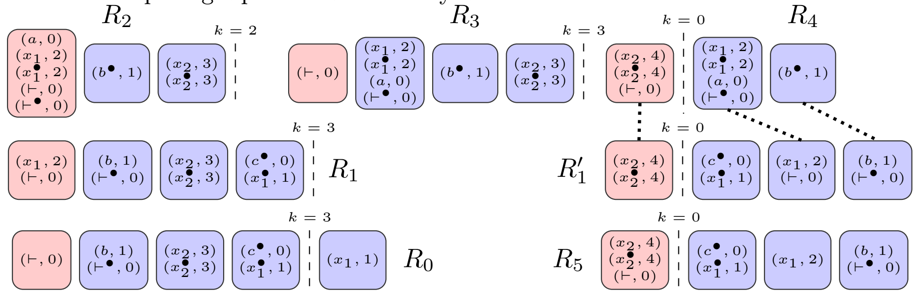

**Fig. 2.** Example regions used in the simulation of the example in Fig. 1

transitions in $\Delta_2$, we pay for each stack symbol in the stack. The set $\Delta_1$ contains transition rules of the form $\langle q, op, q' \rangle$, where *op* is one of (i) *nop*, an operation that does not modify the stack, (ii) *push*$(a)$, which pushes $a$ on top of the stack, or (iii) *pop*$(a)$ which pops $a$ off the stack. The set $\Delta_2$ contains transition rules of the form $\langle q, sc, q' \rangle$, where $q, q' \in Q$ ($sc$ stands for "stack cost").

*Semantics.* A configuration $\beta$ of $\mathcal{P}$ is a tuple $\langle q, w \rangle$, where $q \in Q$ is a state and $w$ is a word over $\Gamma$. The initial configuration $\beta_{init}$ is the configuration $\langle q_{init}, \epsilon \rangle$. Let $Conf(\mathcal{P})$ denote the set of configurations of $\mathcal{P}$.

For two configurations $\beta = \langle q, w \rangle$ and $\beta' = \langle q', w' \rangle$, we have $\gamma \xrightarrow{t} \gamma'$ if either (i) $w' = w$ and $t = \langle q, nop, q' \rangle \in \Delta_1$, (ii) $w' = w \cdot a$ and $t = \langle q, push(a), q' \rangle \in \Delta_1$ for some $a \in \Gamma$, (iii) $w = w' \cdot a$ and $t = \langle q, pop(a), q' \rangle \in \Delta_1$ for some $a \in \Gamma$, or (iv) $w' = w$ and $t = \langle q, sc, q' \rangle \in \Delta_2$. We define the functions $\texttt{State}\,(\beta) = q$ and and $\texttt{Stack}\,(\beta) = w$. A computation $\pi$ to a configuration $\beta$ of $\mathcal{P}$ is a finite sequence $\beta_0 \xrightarrow{t_1} \beta_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \beta_n$ where $\beta_0 = \beta_{init}$, $\beta_n = \beta$, and for all $1 \leq i \leq n$, $\beta_i \in Conf(\mathcal{P})$ and $t_i \in \Delta_1 \cup \Delta_2$. We define $Comp(\mathcal{P}, q)$ to be the set of computations to a configuration $\beta$ such that $\texttt{State}\,(\beta) = q$. In this case, we also say that $\pi$ is a computation to $q$.

*Cost of Computations.* For a transition $\beta \xrightarrow{t} \beta'$, where $\beta = \langle q, a_0 \cdots a_m \rangle$ and $\beta' = \langle q', w \rangle$, we define its cost as $Cost(\beta \xrightarrow{t} \beta') = Cost(t)$ if $t \in \Delta_1$ and $Cost(\beta \xrightarrow{t} \beta') = \sum_{i=0}^{m} Cost(a_i)$ if $t \in \Delta_2$. The cost of a computation $\pi = \beta_0 \xrightarrow{t_1} \beta_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \beta_n$ is then defined as $Cost(\pi) = \sum_{i=1}^{n} Cost(\beta_{i-1} \xrightarrow{t_i} \beta_i)$. We define the *cost-optimal reachability problem* for PPA in the same manner as for PTPA. Lemma 6 follows by a reduction to the same problem for *priced discrete-timed pushdown automata* [2]:

**Lemma 6.** *Let $\mathcal{P}$ be a* PPA *and $q$ be a state. Then $Cost_{opt}(\mathcal{P}, q)$ is computable.*

## 6 Regions

For timed automata, the proof of reachability is achieved using the classical region encoding [5], which is a finite-state abstraction of an uncountable state space. In [1], we showed that it is possible to extend this encoding to timed pushdown automata. More precisely, we show that given a timed pushdown automaton, it is possible to construct an untimed pushdown automaton that simulates it w.r.t. reachability properties. This works by storing regions in the stack of the untimed automaton. The main difficulty in the construction is that the automaton can only access the top element of the stack. However, we might need to remember relationships between elements that lie arbitrarily far apart in the stack. We showed that it is possible to extend the regions in a finite way to capture all such dependencies.

Let $\mathcal{T} = \langle Q, q_{init}, \Gamma, X, \Delta, \mathit{Cost}\rangle$ be a PTPA. We define the set $Y = X \cup \Gamma \cup \{\vdash\}$ of *plain items* and a corresponding set $Y^\bullet = X^\bullet \cup \Gamma^\bullet \cup \{\vdash^\bullet\}$ of *shadow items*. We then define the set of *items* $Z = Y \cup Y^\bullet$. Intuitively, a shadow item in a region on the stack records the value of its plain counterpart in the region below. Let $c_{max}$ be the largest constant in the definition of $\mathcal{T}$ and $\mathit{Max} = \{0, 1, \ldots, c_{max}, \infty\}$. Here, $\infty$ is a symbolic value representing anything larger than $c_{max}$. A *region* $R$ is a tuple $\langle w, k\rangle$, consisting of a word $w = r_0 r_1 \ldots r_n \in (2^{Z \times \mathit{Max}})^+$ and a *boundary position* $k : 0 \leq k \leq n$, such that $w$ satisfies the following conditions:

- $\sum_{i=0}^{n} |r_i \cap (\Gamma \times \mathit{Max})| = 1$ and $\sum_{i=0}^{n} |r_i \cap (\Gamma^\bullet \times \mathit{Max})| = 1$. There is exactly one occurrence of a stack symbol and one occurrence of a shadow stack symbol.
- $\sum_{i=0}^{n} |r_i \cap (\{\vdash\} \times \mathit{Max})| = 1$ and $\sum_{i=0}^{n} |r_i \cap (\{\vdash^\bullet\} \times \mathit{Max})| = 1$. There is exactly one occurrence of $\vdash$ and one occurrence of $\vdash^\bullet$.
- For all clocks $x \in X$, $\sum_{i=0}^{n} |r_i \cap (\{x\} \times \mathit{Max})| = 1$ and $\sum_{i=0}^{n} |r_i \cap (\{x^\bullet\} \times \mathit{Max})| = 1$. Each plain clock symbol and shadow clock symbol occurs exactly once.
- $r_i \neq \emptyset$ for all $1 \leq i \leq n$. Only the first set may be empty.

The purpose of the boundary position is to separate the items with low fractional parts from those with high fractional parts. For $z \in Z$, if $\langle z, m\rangle \in r_i$ for some (unique) $m \in \mathit{Max}$ and $i \in \{0, \ldots, n\}$, then let $\mathit{Val}(R)(z) = m$ and $\mathit{Index}(R)(z) = i$. Otherwise, define $\mathit{Val}(R)(z) = \perp$ and $\mathit{Index}(R)(z) = \perp$ (this may only be the case for stack symbols). We define $R^\top = \{z \in Z \mid \mathit{Index}(R)(z) \neq \perp\}$. Note that the set of regions, w.r.t. fixed $X$, $\Gamma$ and $c_{max}$, is finite. As an example, the region $R_0$ in Fig. 2 is the topmost region in the symbolic stack representing the configuration $\gamma_0$ in Fig. 1.

Next, we define a number of operations on regions that we need for the construction of the symbolic automaton.

*Satisfiability.* Given an item $z \in Z$, an interval $I \in \mathcal{I}$, and a region $R$ such that $z \in R^\top$, we write $R \models (z \in I)$ if one of the following conditions holds:

- $\mathit{Index}(R)(z) = 0$, $\mathit{Val}(R)(z) \neq \infty$ and $\mathit{Val}(R)(z) \in I$. If the fractional part of $z$ is 0, we test if the value of $z$ is in $I$.
- $\mathit{Index}(R)(z) > 0$, $\mathit{Val}(R)(z) \neq \infty$ and $(\mathit{Val}(R)(z) + v) \in I$ for all $v \in \mathbb{R}^{\geq 0}$ such that $0 < v < 1$. If the fractional part of $z$ is greater than 0, we test if its integral part increased by any real number $v : 0 < v < 1$ is in the interval.
- $\mathit{Val}(R)(z) = \infty$, and $I$ is of the form $(m : \infty)$ or of the form $[m : \infty)$. If the integral part of $z$ is $\infty$, then the interval cannot have an upper bound.

*Adding and Removing Items.* For a region $R = \langle r_0 \dots r_n, k\rangle$, an item $z \in Z$, and $m \in \mathit{Max}$, we define $R \oplus \langle z, m\rangle$ to be the set of regions $R'$ which satisfy one of the following conditions:

- $R' = \langle r_0 \dots r_{i-1}(r_i \cup \{\langle z, m\rangle\})r_{i+1} \dots r_n, k\rangle$, where $0 \le i \le n$. The item is added into an existing set, in which case $k$ is unchanged.
- $R' = \langle r_0 \dots r_i\{\langle z, m\rangle\}r_{i+1} \dots r_n, k+1\rangle$, where $1 \le i \le k$. The item is added as a new singleton set to the left of $k$, in which case $k$ is increased by 1.
- $R' = \langle k, r_0 \dots r_i\{\langle z, m\rangle\}r_{i+1} \dots r_n, k\rangle$, where $k \le i \le n$. The item is added as a new set to the right of $k$, which is left unchanged.

We define $R \ominus z$ to be the region $R' = \langle r_0 \cdots r_{j-1}(r_j \setminus (\{z\} \times \mathit{Max}))r_{j+1} \dots r_n, k'\rangle$, where $j$ is the unique value s.t. $r_j \cap (\{z\} \times \mathit{Max}) \neq \emptyset$ and where $k' = k-1$ if $0 < j \le k$ and $|r_j| = 1$, and $k' = k$ otherwise. In other words, we delete $z$ from a set, if it exists, and update $k$ accordingly. We extend the definition of $\ominus$ to sets of items by letting $R \ominus \{z_1 \dots z_n\} = (\cdots((R \ominus z_1) \ominus z_2) \cdots) \ominus z_n$.

*Resetting.* For a region $R = \langle r_0 \dots r_n, k\rangle$ and an item $z \in Z$, we define $R[z \leftarrow 0]$ to be the unique region $\langle (r'_0 \cup \{\langle z, 0\rangle\})r'_1 \cdots r'_n, k\rangle$, where $r'_0 r'_1 \cdots r'_n = R \ominus z$. We delete $z$ from the region and reintroduce it with value 0. This operation is used when we simulate the resetting of clocks.

*Pushing New Regions.* This operation is used to simulate the pushing of new regions. It takes a region $R$ and a stack symbol $a \in \Gamma$, and creates a new region $R'$ in which the shadow items record the values of the plain items in $R$ and where the value of $a$ is 0. We define $\mathit{New}(R, a)$ to be the region $R'$ such that there are $R_1, R_2, R_3$ satisfying the following conditions (for example, in Fig. 2, $R_2 = \mathit{New}(R_1, a)$):

- $R_1 = \langle r_0 \cdots r_{n_1}, k\rangle = R \ominus (R^\top \cap Y^\bullet)$. Delete all shadow items from $R$.
- $R_2 = \left\langle r'_0 \dots r'_{n_1}, k\right\rangle$, where $r'_i = r_i \cup \{\langle y^\bullet, m\rangle \mid \langle y, m\rangle \in r_i\}$ for $0 \le i \le n_1$. Add fresh shadow items with the same values and the same indices as their plain counterparts.
- $R_3 = \left\langle r''_0 \dots r''_{n_2}, k\right\rangle = R_2 \ominus (R^\top \cap \Gamma)$. Delete the previous stack symbol.
- $R' = \left\langle (r''_0 \cup \{\langle a, 0\rangle\})r''_1 \dots r''_{n_2}, k\right\rangle$. Introduce $a$ with value 0.

*Passage of Time.* Next, we describe operations that simulate the passage of time. Given a pair $\langle z, m\rangle \in Z \times \mathit{Max}$, define $\langle z, m\rangle^\oplus = \langle z, m'\rangle$, where $m' = m+1$ if $m < c_{max}$, and $m' = \infty$ otherwise. For a set $r \in 2^{Z \times \mathit{Max}}$, define $r^\oplus = \{\langle z, m\rangle^\oplus \mid \langle z, m\rangle \in r\}$. In other words, we increase the integral part of each item by 1, up to $c_{max}$. For a region $R = \langle r_0 \dots r_n, k\rangle$, define $R^\oplus = \langle w', k'\rangle$ in the following way:

- If $r_0 \neq \emptyset$, then $w' = \emptyset r_0 \dots r_n$, and $k' = k+1$. A small amount of time passes, which results in the first region being "pushed" out.
- If $r_0 = \emptyset$ and $k < n$, then $w' = {r_n}^\oplus r_1 \dots r_{n-1}$ and $k' = k$. The items in the last region reach their next integral value but the small fractional parts remain small.
- If $r_0 = \emptyset$ and $k = n$, then $w' = w$ and $k' = 0$. All small fractional parts become large, but no integral part changes.

We denote by $R^{\oplus\oplus}$ the set $\{R, R^{\oplus}, (R^{\oplus})^{\oplus}, \ldots\}$. Note that this set is finite. We define $R^{\oplus}_{\vdash}$ to be the region $R'$ such that there are $R_1$ and $R_2$ satisfying the following conditions:

– $R_1 = R^{\oplus} \ominus \vdash$. We remove the item $\vdash$.
– $R_2 \in R_1 \oplus (\vdash, 0)$ and $R_2 \models (\vdash \in [0..0])$. We reintroduce $\vdash$ by placing it in the leftmost set. Note that $R_2$ is unique.

The operation $R^{\oplus}_{\vdash}$ simulates passage of time while maintaining the value and index of $\vdash$. We define $R^{\oplus\oplus}_{\vdash}$ similarly to $R^{\oplus\oplus}$. In Fig. 2, we have that $R_3 \in {R_2}^{\oplus\oplus}$.

*Product.* We now define the product operator $\odot$ that merges the information in two regions. This operation is used when simulating *pop* transitions. For regions $R_1 = \langle w_1, k_1 \rangle$ and $R_2 = \langle w_2, k_2 \rangle$, we write $R_1 \preceq R_2$ if there are $i_0 < i_1 < \cdots < i_\ell \leq |w_1|$ and $j_0 < j_1 < \cdots < j_\ell \leq |w_2|$ such that: (1) $i_0 = j_0 = 0$, (2) $\{i_1, \ldots, i_\ell\} \subseteq \mathit{Index}\,(R_1)\left(R_1^{\top} \cap Y\right) \subseteq \{i_0, i_1, \ldots, i_\ell\}$, (3) $\{j_1, \ldots, j_\ell\} \subseteq \mathit{Index}\,(R_2)\left({R_2}^{\top} \cap Y^{\bullet}\right) \subseteq \{j_0, j_1, \ldots, j_\ell\}$, (4) $({R_2}^{\top} \cap Y^{\bullet}) = \{y^{\bullet} \,|\, y \in (R_1^{\top} \cap Y)\}$, and (5) for every $h : 0 \leq h \leq \ell$ and $y \in (R_1^{\top} \cap Y)$, we have: (*i*) $\mathit{Index}\,(R_1)\,(y) = i_h$ iff $\mathit{Index}\,(R_2)\,(y^{\bullet}) = j_h$, and (*ii*) $j_h \leq k_2$ iff $i_h \leq k_1$. In this case we say that $R_1$ *supports* $R_2$. Intuitively, this means that the shadow items in $R_2$ match their plain counterparts in $R_1$ and that the information in the two regions can be merged. In Fig. 2, we have that $R_1' \preceq R_4$. The matching is illustrated by dotted lines.

Assume that $R_1 \preceq R_2$, $w_1 = r_0 \ldots r_n$ and $w_2 = r_0' \ldots r_{n'}'$. Let $v_h = r_{i_h+1} \cdots r_{i_{h+1}-1}$, $v_h' = r_{i_h+1}' \cdots r_{i_{h+1}-1}'$ for all $h : 0 \leq h < \ell$, $v_\ell = r_{i_\ell+1} \cdots r_n$, and $v_\ell' = r_{i_\ell+1}' \cdots r_{n'}'$. Note that the sequences of indices $i_0, \ldots i_\ell$ and $j_0, \ldots, j_\ell$ are unique. We define $p_h = r_{i_h} \cap (Y^{\bullet} \cup \Gamma)$ and $p_h' = r_{j_h}' \cap (X \cup \{\vdash\})$ for all $h : 0 \leq h \leq \ell$. We define $q_0 = p_0 \cup p_0'$ and, for $1 \leq h \leq \ell$, define $q_h = p_h \cup p_h'$ if $p_h \cup p_h' \neq \emptyset$ and $q_h = \epsilon$ otherwise. Then, $w \in w_1 \odot w_2$ if $w = q_0 \cdot w_0' \cdot q_1 \cdots q_\ell \cdot w_\ell'$ and $w_h' \in (v_h \otimes v_h')$ for $h : 0 \leq h \leq \ell$. Intuitively, we take the clocks from $w_2$, and the shadow items and stack symbol from $w_1$. For regions $R_1 = \langle w_1, k_1 \rangle$ and $R_2 = \langle w_2, k_2 \rangle$, we have $R = \langle w, k \rangle \in R_1 \odot R_2$ if $w \in w_1 \odot w_2$ and for all $z \in Z$, the following conditions hold (in Fig. 2 we have $R_5 \in R_1' \odot R_4$):

– $\mathit{Index}\,(R)\,(z) \leq k$ iff $\mathit{Index}\,(R_1)\,(z) \leq k_1$, for $z \in \Gamma^{\bullet} \cup X^{\bullet} \cup \Gamma$. The boundary should be preserved for the shadow items and the stack symbol from $R_1$.
– $\mathit{Index}\,(R)\,(z) \leq k$ iff $\mathit{Index}\,(R_2)\,(z) \leq k_2$, for $z \in X$. The boundary should be preserved for the clocks taken from $R_2$.

## 7 Simulation

We will describe how, given a PTPA $\mathcal{T} = (Q, q_{init}, \Gamma, X, \Delta, \mathit{Cost})$, one can construct a priced pushdown automaton PPA $\mathcal{P} = (Q^{\mathcal{P}}, q^{\mathcal{P}}_{init}, \Gamma^{\mathcal{P}}, \Delta^{\mathcal{P}}_1, \Delta^{\mathcal{P}}_2, \mathit{Cost}^{\mathcal{P}})$ that simulates detailed computations in $\delta$-form of $\mathcal{T}$ for arbitrarily small $\delta$.

The states of $\mathcal{P}$ contain both the states of $\mathcal{T}$, which are called *genuine*, and a set of *temporary* states which are used as intermediate states to simulate the transitions of $\mathcal{T}$. More precisely, we use a set $Q_{\mathtt{tmp}}$ of temporary states s.t.

$Q_{\texttt{tmp}} \cap Q = \emptyset$. We write $\texttt{tmp}(\dots)$, $\texttt{tmp}_1(\dots)$, $\texttt{tmp}_2(\dots)$ and $\texttt{tmp}_3(\dots)$ to denote unique elements of $Q_{\texttt{tmp}}$, where the arguments are used to uniquely identify elements of $Q_{\texttt{tmp}}$. We also assume that $q^{\mathcal{P}}_{init} \notin Q$ and $q^{\mathcal{P}}_{init} \notin Q_{\texttt{tmp}}$. Then, $Q^{\mathcal{P}}$ is defined as $Q \cup Q_{\texttt{tmp}} \cup \{q^{\mathcal{P}}_{init}\}$.

The stack alphabet $\Gamma^{\mathcal{P}}$ of $\mathcal{P}$ is the set of regions over the stack alphabet $\Gamma \cup \{\bot\}$, the set of clocks $X$ and the maximal constant $c_{max}$, where $\bot \notin \Gamma$ is a special symbol that represents the bottom of the stack. Note that $\Gamma^{\mathcal{P}}$ is finite. We define $Cost^{\mathcal{P}}(\bot) = 0$. The cost of a region $R$ is defined as $Cost^{\mathcal{P}}(R) = Cost(a)$, where $a$ is the unique stack symbol s.t. $R^{\top} \cap \Gamma = \{a\}$.

Let $w_{init} = \{\langle z, 0\rangle \mid z \in X \cup X^{\bullet} \cup \{\vdash, \vdash^{\bullet}\}\} \cup \{\langle \bot, 0\rangle, \langle \bot^{\bullet}, 0\rangle\}$. The symbolic automaton starts in its initial configuration $\langle q^{\mathcal{P}}_{init}, \epsilon\rangle$ and pushes the initial region $R_{init} = \langle w_{init}, 0\rangle$ on the stack while moving to the initial state of $\mathcal{T}$, i.e. $\Delta^{\mathcal{P}}_1$ contains the rule $\langle q^{\mathcal{P}}_{init}, push(R_{init}), q_{init}\rangle$. We define

$$Cost^{\mathcal{P}}(\langle q^{\mathcal{P}}_{init}, push(R_{init}), q_{init}\rangle) = 0.$$

Then, $\mathcal{P}$ starts the simulation of $\mathcal{T}$. The transitions of $\mathcal{T}$ are simulated in the following way (as an example, Fig. 2 shows the regions involved in the simulation of Fig. 1):

– *Nop.* For every $t = \langle q_1, nop, q_2\rangle \in \Delta$, the set $\Delta^{\mathcal{P}}_1$ contains $t = \langle q_1, nop, q_2\rangle$. Define $Cost^{\mathcal{P}}(t) = Cost(t)$.

– *Push.* We need two temporary states to simulate this transition. First, we move to a temporary state and pop the topmost region in order to remember its content. Then, we push back that region, moving to the second temporary state. Finally, we push a new topmost region that we construct using the remembered values. For every $t = \langle q_1, push(a), q_2\rangle \in \Delta$, and every region $R$, the set $\Delta^{\mathcal{P}}_1$ contains $t_1 = \langle q_1, pop(R), \texttt{tmp}_1(t, R)\rangle$, $t_2 = \langle \texttt{tmp}_1(t, R), push(R), \texttt{tmp}_2(t, R)\rangle$, and $t_3 = \langle \texttt{tmp}_2(t, R), push(New(R, a)), q_2\rangle$. We define $Cost^{\mathcal{P}}(t_1) = 0$, $Cost^{\mathcal{P}}(t_2) = 0$ and $Cost^{\mathcal{P}}(t_3) = Cost(t)$.

– *Pop.* To simulate pop transitions, we use two temporary states. We first pop the topmost region. The new topmost region then needs to be updated to reflect the changes that occurred while it was inaccessible. To do this, we rotate it until it matches the popped region. Then, we merge the information in both regions. In this way, the information about changes "ripples" down the stack as elements are popped. The effect of both these steps is captured by popping the new topmost region and pushing a region which is a product of the two popped regions. Formally, for every $t = \langle q_1, pop(a \in I), q_2\rangle \in \Delta$, and all regions $R_1, R_2$ such that $R_2 \models a \in I$, the set $\Delta^{\mathcal{P}}_1$ contains $t_1 = \langle q_1, pop(R_2), \texttt{tmp}_1(t, R_2)\rangle$ and $t_2 = \langle \texttt{tmp}_1(t, R_2), pop(R_1), \texttt{tmp}_2(t, R_2, R_1)\rangle$. Additionally, $\Delta^{\mathcal{P}}_1$ contains $t_{R_3} = \langle \texttt{tmp}_2(t, R_2, R_1), push(R_3), q_2\rangle$ for all $R_3 \in \{R'_1 \odot R_2 \mid R'_1 \in {R_1}^{\oplus\oplus}, R'_1 \preceq R_2\}$. We define their costs as $Cost^{\mathcal{P}}(t_1) = 0$, $Cost^{\mathcal{P}}(t_2) = 0$ and $Cost^{\mathcal{P}}(t_{R_3}) = Cost(t)$ for all the above $R_3$. In Fig. 2, $R_5$ is the result of popping $R_4$ when $R_1$ is the second topmost region.

– *Test.* We simulate a test transition with two transitions. First, we pop the region if it satisfies the condition, while moving to a temporary state. Next, we push the same region and move to the new genuine state. For every

$t = \langle q_1, \mathit{test}(x \in I), q_2\rangle \in \Delta$, and every region $R$ such that $R \models x \in I$, the set $\Delta_1$ contains $t_1 = \langle q_1, \mathit{pop}(R), \mathtt{tmp}(t,R)\rangle$ and $t_2 = \langle \mathtt{tmp}(t,R), \mathit{push}(R), q_2\rangle$. Define $\mathit{Cost}^{\mathcal{P}}(t_1) = 0$ and $\mathit{Cost}^{\mathcal{P}}(t_2) = \mathit{Cost}(t)$.

– *Reset.* We simulate resetting $x$ by popping the topmost region and pushing back a region which is identical, except that $x$ is 0. For every $t = \langle q_1, \mathit{reset}(x), q_2\rangle \in \Delta$, and every region $R$, the set of $\Delta_1^{\mathcal{P}}$ contains $t_1 = \langle q_1, \mathit{pop}(R), \mathtt{tmp}(t,R)\rangle$ and $t_2 = \langle \mathtt{tmp}(t,R), \mathit{push}(R[x \leftarrow 0]), q_2\rangle$.

$$\text{Define } \mathit{Cost}^{\mathcal{P}}(t_1) = 0 \text{ and } \mathit{Cost}^{\mathcal{P}}(t_2) = \mathit{Cost}(t).$$

– *Timed Transitions.* To simulate timed transitions, we pop the topmost region, rotate it, and push it back. Let $R = \langle r_0 \dots r_n, k\rangle$ be a region.
  - If $r_0 \neq \emptyset$ (resp. $r_0 = \emptyset$ and $k < n$), then we simulate a short timed transition which makes the fractional part of all value positive (resp. the highest fractional part 0). In this case, $\Delta_1^{\mathcal{P}}$ contains
$$t_1 = \langle q, \mathit{pop}(R), \mathtt{tmp}_1(\mathit{time}, q, R)\rangle \text{ and}$$
$$t_2 = \left\langle \mathtt{tmp}_1(\mathit{time}, q, R), \mathit{push}(R_{\vdash}^{\oplus}), q\right\rangle .$$
We define $\mathit{Cost}^{\mathcal{P}}(t_1) = \mathit{Cost}^{\mathcal{P}}(t_2) = 0$.
  - If $r_0 = \emptyset$ and $k = n$, then we simulate a long timed transition. In this case, $\Delta_1^{\mathcal{P}}$ contains
$$t_1 = \langle q, \mathit{pop}(R), \mathtt{tmp}_1(\mathit{time}, q, R)\rangle \, \mathit{and}$$
$$t_2 = \left\langle \mathtt{tmp}_1(\mathit{time}, q, R), \mathit{push}(R_{\vdash}^{\oplus}), \mathtt{tmp}_2(\mathit{time}, q, R)\right\rangle .$$
Define $\mathit{Cost}^{\mathcal{P}}(t_1) = \mathit{Cost}^{\mathcal{P}}(t_2) = 0$. Since the transition is long, we must pay the stack cost. Therefore, $\Delta_2^{\mathcal{P}}$ contains $\langle \mathtt{tmp}_2(\mathit{time}, q, R), \mathit{sc}, q\rangle$.

*Correctness.* Consider a detailed computation $\pi_{\mathcal{T}}$ in $\delta$-form of length $n$ in $\mathcal{T}$ and its simulation $\pi_{\mathcal{P}}$ in $\mathcal{P}$. We will give a bound on the difference in cost between each step in $\pi_{\mathcal{T}}$ compared to its corresponding steps in $\pi_{\mathcal{P}}$. The costs of the discrete steps are identical. Now we consider timed transitions. If the timed transition is short, its cost is bounded by $\delta$ multiplied by the stack cost at that step, while the cost of the corresponding steps in $\pi_{\mathcal{P}}$ is equal to 0. On the other hand, if the timed transition is long, the cost is bounded by $(1 - \delta)$ multiplied by the stack cost, while the cost of the corresponding steps is exactly equal to the stack cost. Consequently, the difference is always bounded by $\delta$ multiplied by the stack cost. Since the stack cost is bounded by the length of the stack multiplied by the cost of the most expensive symbol, and since the length of the stack is bounded by $n$, the difference in price at each step is bounded by $\delta \cdot |\pi_{\mathcal{T}}| \cdot \max\{\mathit{Cost}(a) \mid a \in \Gamma\}$. This implies that the total difference in cost between $\pi_{\mathcal{T}}$ and $\pi_{\mathcal{P}}$ is bounded by $\delta \cdot |\pi_{\mathcal{T}}|^2 \cdot \max\{\mathit{Cost}(a) \mid a \in \Gamma\}$. This gives the following lemma:

**Lemma 7.** *$\forall \delta : 0 < \delta < \frac{1}{10}$ and $\forall \pi_{\mathcal{T}}.\, \pi_{\mathcal{T}} \in \mathit{Comp}_{\delta}^{\mathsf{d}}(\mathcal{T}, q)$, $\exists \pi_{\mathcal{P}}.\, \pi_{\mathcal{P}} \in \mathit{Comp}(\mathcal{P}, q)$ s.t. $|\mathit{Cost}^{\mathcal{P}}(\pi_{\mathcal{P}}) - \mathit{Cost}(\pi_{\mathcal{T}})| \leq \delta \cdot |\pi_{\mathcal{T}}|^2 \cdot \max\{\mathit{Cost}(a) \mid a \in \Gamma\}$.*

The other direction can be explained in a similar manner:

**Lemma 8.** $\forall \pi_{\mathcal{P}}.\, \pi_{\mathcal{P}} \in Comp(\mathcal{P}, q)$*, and* $\forall \delta : 0 < \delta < \frac{1}{10}$*,* $\exists \pi_{\mathcal{T}}.\, \pi_{\mathcal{T}} \in Comp^{\mathsf{d}}_{\delta}(\mathcal{T}, q)$ *s.t.* $|Cost^{\mathcal{P}}(\pi_{\mathcal{P}}) - Cost(\pi_{\mathcal{T}})| \leq \delta \cdot |\pi_{\mathcal{T}}|^2 \cdot \max\{Cost(a) \mid a \in \Gamma\}$*.*

We now combine Lemma 2, Lemma 4, and Lemmas 8 and 7 to prove the following:

**Theorem 9.** $Cost_{opt}(\mathcal{T}, q) = Cost_{opt}(\mathcal{P}, q)$ *for any state* $q \in Q$*.*

Thus, the cost-optimal reachability problem for PTPA reduces to the same problem for PPA, which is computable by Lemma 6. This concludes the proof of Theorem 1.

## 8 Conclusion

We have studied the cost-optimal reachability problem for priced dense-timed pushdown automata, and shown that this problem can be reduced to the same problem for priced (untimed) pushdown automata, which in turn can be reduced to the reachability problem for ordinary pushdown automata. This yields an algorithm for computing the optimal reachability cost for PTPA. To simplify the exposition, we assumed that push (resp. reset) operations result in the stack symbol having age (resp. affected clock having value) 0. The model still strictly subsumes that of [1] (we can encode the intervals in the stack symbols).

A simple generalization is to make the stack symbol storage cost dependent on the current control-state. Our construction can be trivially extended to handle this case. (We need to consider a similar extension for the priced pushdown automata.)

A challenging problem which we are currently considering is to extend our results to the case of negative costs.

## References

1. Abdulla, P.A., Atig, M.F., Stenman, J.: Dense-timed pushdown automata. In: LICS (2012)
2. Abdulla, P.A., Atig, M.F., Stenman, J.: The minimal cost reachability problem in priced timed pushdown systems. In: LATA. pp. 58–69. Springer (2012)
3. Abdulla, P.A., Čerāns, K., Jonsson, B., Tsay, Y.K.: General decidability theorems for infinite-state systems. In: LICS. pp. 313–321 (1996)
4. Abdulla, P.A., Mayr, R.: Computing optimal coverability costs in priced timed Petri nets. In: LICS (2011)
5. Alur, R., Dill, D.L.: A theory of timed automata. TCS 126(2), 183–235 (1994)
6. Alur, R., La Torre, S., Pappas, G.J.: Optimal paths in weighted timed automata. TCS 318(3), 297–322 (2004)
7. Behrmann, G., Fehnker, A., Hune, T., Larsen, K.G., Pettersson, P., Romijn, J., Vaandrager, F.W.: Minimum-cost reachability for priced timed automata. In: HSCC. pp. 147–161. LNCS 2034, Springer (2001)
8. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Application to model-checking. In: CONCUR (1997)

9. Bouyer, P., Brihaye, T., Bruyere, V., Raskin, J.F.: On the optimal reachability problem of weighted timed automata. FMSD 31(2), 135–175 (2007)
10. Dang, Z.: Pushdown timed automata: a binary reachability characterization and safety verification. TCS 302(1-3), 93–121 (2003)
11. Karp, R.M., Miller, R.E.: Parallel program schemata. JCSS (1969)
12. Trivedi, A., Wojtczak, D.: Recursive timed automata. In: ATVA. pp. 306–324 (2010)
13. Uezato., Y., Minamide, Y.: Pushdown systems with stack manipulation. In: ATVA (2013)