Minimum-Cost Reachability for Priced Timed Automata*

Gerd Behrmann[1], Ansgar Fehnker[3], Thomas Hune[2], Kim Larsen[1]
Paul Pettersson[4], Judi Romijn[3], and Frits Vaandrager[3]

[1] Basic Research in Computer Science, Aalborg University,
E-mail: {behrmann,kgl}@cs.auc.dk.
[2] Basic Research in Computer Science, Aarhus University,
E-mail: baris@brics.dk.
[3] Computing Science Institute, University of Nijmegen,
E-mail: {ansgar,judi,fvaan}@cs.kun.nl.
[4] Department of Computer Systems, Information Technology,
Uppsala University, E-mail: paupet@docs.uu.se.

Abstract. This paper introduces the model of *linearly priced timed automata* as an extension of timed automata with prices on both transitions and locations. For this model we consider the minimum-cost reachability problem: i.e. given a linearly priced timed automaton and a target state, determine the minimum cost of executions from the initial state to the target state. This problem generalizes the minimum-time reachability problem for ordinary timed automata to more general optimization problems like static scheduling problems. We prove decidability of this problem by offering an algorithmic solution, which is based on a combination of branch-and-bound techniques and a new notion of priced regions. The latter allows symbolic representation and manipulation of reachable states together with the cost of reaching them.
Keywords: Timed Automata, Verification, Data Structures, Algorithms, Optimization.

1 Introduction

Recently, real-time verification tools such as UPPAAL [13], KRONOS [6] and HYTECH [10], have been applied to synthesize feasible solutions to static job-shop scheduling problems [8, 12, 17]. The basic common idea of these works is to reformulate the static scheduling problem as a reachability problem that can be solved by the verification tools. In this approach, the timed automata [3] based modeling languages of the verification tools serve as the basic input language in which the scheduling problem is described. These modeling languages have proved particularly well-suited in this respect, as they allow for easy and flexible modeling of systems, consisting of several parallel components that interact in a time-critical manner and constrain each other's behavior in a multitude of ways.

* This work is partially supported by the European Community Esprit-LTR Project 26270 VHS (Verification of Hybrid Systems).

In this paper we introduce the model of *linearly priced timed automata* and offer an algorithmic solution to the problem of determining the minimum cost of reaching a designated set of target states. This result generalizes previous results on computation of minimum-time reachability and accumulated delays in timed automata, and should be viewed as laying a theoretical foundation for algorithmic treatments of more general optimization problems as encountered in static scheduling problems.

As an example consider the very simple static scheduling problem represented by the timed automaton in Fig. 1 from [16], which contains 5 'tasks' $\{A, B, C, D, E\}$. All tasks are to be performed precisely once, except task C, which should be performed *at least* once. The order of the tasks is given by the timed automaton, e.g. task B must not commence before task A has finished. In addition, the timed automaton specifies three timing requirements to be satisfied: the delay between the start of the first execution of task C and the start of the execution of E should be at least 3 time units; the delay between the start of the last execution of C and the start of D should be no more than 1 time units; and, the delay between the start of B and the start of D should be at least 2 time units, each of these requirements are represented by a clock in the model. Using a standard timed model checker we are able to verify that location E of

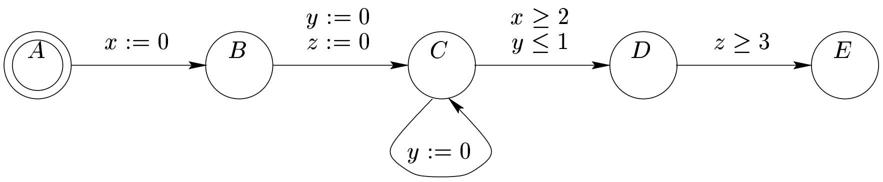

Fig. 1. Timed automata model of scheduling example.

the timed automaton is reachable. This can be demonstrated by a trace leading to the location[1]:

$$(A, 0, 0, 0) \xrightarrow{\tau} \xrightarrow{\epsilon(1)} (B, 1, 1, 1) \xrightarrow{\tau} \xrightarrow{\epsilon(1)} (C, 2, 1, 1) \xrightarrow{\tau} \xrightarrow{\epsilon(2)} (D, 4, 3, 3) \xrightarrow{\tau} (E, 4, 3, 3) \quad (1)$$

The above trace may be viewed as a feasible solution to the original static scheduling problem. However, given an optimization problem, one is often not satisfied with an arbitrary feasible solution but insist on solutions which are *optimal* in some sense. When modeling a problem like this one using timed automata an obvious notion of optimality is that of minimum accumulated time. For the timed automaton of Fig. 1 the trace of (1) has an accumulated time-duration of 4. This, however, is not optimal as witnessed by the following alternative trace,

[1] Here a quadruple (X, v_x, v_y, v_z) denotes the state of the automaton in which the control location is X and where v_x, v_y and v_z give the values of the three clocks x, y and z. The transitions labelled τ are actual transitions in the model, and the transitions labelled $\epsilon(d)$ represents a delay of d time units.

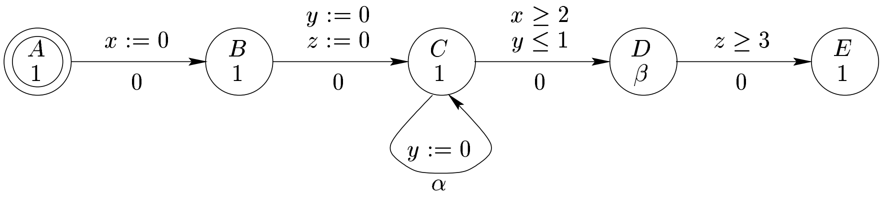

Fig. 2. A linearly priced timed automaton.

which by exploiting the looping transition on C reaches E within a total of 3 time-units[2]:

$$(A,0,0,0) \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\epsilon(2)} (C,2,2,2) \xrightarrow{\tau} (C,2,0,2) \xrightarrow{\tau}\xrightarrow{\epsilon(1)} (D,3,1,3) \xrightarrow{\tau} (E,3,1,3) \quad (2)$$

In [4] algorithmic solutions to the minimum-time reachability problem and the more general problem of controller synthesis has been given using a backward fix-point computation. In [16] an alternative solution based on forward reachability analysis is given, and in [5] an algorithmic solution is offered, which applies branch-and-bound techniques to prune parts of the symbolic state-space which are guaranteed not to contain optimal solutions. In particular, by introducing an additional clock for accumulating time-elapses, the minimum-time reachability problem may be dealt with using the existing efficient data structures (e.g. DBMs [7], CDDs [14] and DDDs [15]) already used in the real-time verification tools UPPAAL and KRONOS for reachability. The results of the present paper also extends the work in [2] which provides an algorithm for computing the accumulated delay in a timed automata.

In this paper, we provide the basis for dealing with more general optimization problems. In particular, we introduce the model of *linearly priced timed automata*, as an extension of timed automata with *prices* on both transitions and locations: the price of a transition gives the cost for taking it and the price on a location specifies the cost *per time-unit* for staying in that location. This model can capture not only the passage of time, but also the way that e.g. tasks with different prices for use per time unit, contributes to the total cost.

Figure 2 gives a linearly priced extension of the timed automaton from Fig. 1. Here, the price of location D is set to β and the price on all other locations is set to 1 (thus simply accumulating time). The price of the looping transition on C is set to α, whereas all other transitions are free of cost (price 0). Now for $(\alpha, \beta) = (1,3)$ the costs of the traces (1) and (2) are 8 and 6, respectively (thus it is cheaper to actually exploit the looping transition). For $(\alpha, \beta) = (2,2)$ the costs of the two traces are both 6, thus in this case it is immaterial whether the looping transition is taken or not. In fact, the optimal cost of reaching E will in general be the minimum of $2 + 2 * \beta$ and $3 + \alpha$, and the optimal trace will include the looping transition on C depending on the particular values of α and β.

[2] In fact, 3 is the minimum time for reaching E.

In this paper we deal with the problem of determining the minimum cost of reaching a given location for linearly priced timed automata. In particular, we offer an algorithmic solution to this problem [3] In contrast to minimum-time reachability for timed automata, the minimum-cost reachability problem for linearly priced timed automata requires the development of new data structures for symbolic representation and the manipulation of reachable *sets* of states *together with* the cost of reaching them. In this paper we put forward one such data structure, namely a priced extension of the fundamental notion of *clock regions* for timed automata [3].

The remainder of the paper is structured as follows: Section 2 formally introduces the model of linearly priced timed automata together with its semantics. Section 3 develops the notion of priced clock regions, together with a number of useful operations on these. The priced clock regions are then used in Section 4 to give a symbolic semantics capturing (sufficiently) precisely the cost of executions and used as a basis for an algorithm solution to the minimum-cost problem. Finally, in Section 5 we give some concluding remarks. In Appendix A and B, some figures and an example are given, and in Appendix C the reader may find the proofs not included in the main paper.

2 Linearly Priced Timed Automata

In this section, we introduce the model of linearly priced timed automata, which is an extension of timed automata [3] with prices on both locations and transitions. Dually, linearly priced timed automata may be seen as a special type of linear hybrid automata [9] or multirectangular automata [9] in which the accumulation of prices (i.e. the cost) is represented by a single continuous variable. However, in contrast to known undecidability results for these classes, minimum-cost reachability is computable for linearly priced timed automata[4].

Let C be a finite set of clocks. Then $\mathcal{B}(C)$ is the set of formulas obtained as conjunctions of atomic constraints of the form $x \bowtie n$ where $x \in C$, n is natural number, and $\bowtie \in \{<, \leq, =, \geq, >\}$. Elements of $\mathcal{B}(C)$ are called *clock constraints* over C. Note that for each timed automaton that has constraints of the form $x - y \bowtie c$, there exists a strongly bisimilar timed automaton with only constraints of the form $x \bowtie c$. Therefore, the results in this paper are applicable to automata having constraints of the type $x - y \bowtie c$ as well.

Definition 1 (Linearly Priced Timed Automaton). *A Linearly Priced Timed Automaton (LPTA) over clocks C and actions Act is a tuple (L, l_0, E, I, P) where L is a finite set of locations, l_0 is the initial location, $E \subseteq L \times \mathcal{B}(C) \times Act \times \mathcal{P}(C) \times L$ is the set of edges, $I : L \to \mathcal{B}(C)$ assigns invariants to locations, and $P : (L \cup E) \to \mathbb{N}$ assigns prices to both locations and edges. In the case of $(l, g, a, r, l') \in E$, we write $l \xrightarrow{g,a,r} l'$.*

[3] Thus settling an open problem given [4].

[4] An intuitive explanation for this is that the additional (cost) variable does not influence the behavior of the automata.

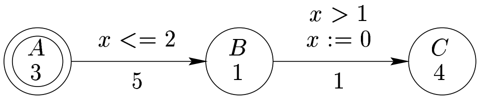

Fig. 3. An example LPTA

Formally, clock values are represented as functions called *clock assignments* from C to the non-negative reals $\mathbb{R}_{\geq 0}$. We denote by $\mathbb{R}^C$ the set of clock assignments for C ranged over by u, u' etc. We define the operation $u' = [r \mapsto 0]u$ to be the assignment such that $u'(x) = 0$ if $x \in r$ and $u(x)$ otherwise, and the operation $u' = u + d$ to be the assignment such that $u'(x) = u(x) + d$. Also, a clock valuation u satisfies a clock constraint g, $u \in g$, if $u(x) \bowtie n$ for any atomic constraint $x \bowtie n$ in g.

The semantics of a LPTA A is defined as a transition system with the state space $L \times \mathbb{R}^C$, with initial state (l_0, u_0) (where u_0 assigns zero to all clocks in C), and with the following transition relation:

- $(l,u) \xrightarrow{\epsilon(d),p} (l, u+d)$ if $u + d \in I(l)$, and $p = P(l) * d$.
- $(l,u) \xrightarrow{a,p} (l', u')$ if there exists g, r such that $l \xrightarrow{g,a,r} l'$, $u \in g$, $u' = [r \mapsto 0]u$, $u' \in I(l')$ and $p = P((l, g, a, r, l'))$.

Note that the transitions are decorated with two labels: a delay-quantity or an action, together with the cost of the particular transition. For determining the cost, the price of a location gives the cost rate of staying in that location (per time unit), and the price of a transition gives the cost of taking that transition. In the remainder, states and executions of the transition system for LPTA A will be referred to as states and executions of A.

Definition 2 (Cost). *Let* $\alpha = (l_0, u_0) \xrightarrow{a_1,p_1} (l_1, u_1) \dots \xrightarrow{a_n,p_n} (l_n, u_n)$ *be a finite execution of LPTA A. The* cost *of α,* $\mathsf{cost}(\alpha)$*, is the sum* $\Sigma_{i \in \{1,\dots,n\}} p_i$.

For a given state (l,u), the minimal cost of reaching (l,u), $\mathsf{mincost}((l,u))$*, is the infimum of the costs of finite executions ending in (l,u). Similarly, the minimal cost of reaching a location l,* $\mathsf{mincost}(l)$*, is the infimum of the costs of finite executions ending in a state of the form (l,u).*

Example 1. Consider the LPTA of Fig. 3. The LPTA has a single clock x, and the locations and transitions are decorated with prices. A sample execution of this LPTA is for instance:

$$(A, 0) \xrightarrow{\epsilon(1.5),4.5} (A, 1.5) \xrightarrow{\tau,5} (B, 1.5) \xrightarrow{\tau,1} (C, 1.5)$$

The cost of this execution is 10.5. In fact, there are executions with cost arbitrarily close to the value 7, obtainable by avoiding delay in location A, and delaying just long enough in location B. Due to the infimum definition of $\mathsf{mincost}$, it follows that $\mathsf{mincost}(c) = 7$. However, note that because of the strict comparison in the guard of the second transition, no execution actually achieves this cost. □

3 Priced Clock Regions

For ordinary timed automata, the key to decidability results has been the valuable notion of *region* [3]. In particular, regions provide a finite partitioning of the uncountable set of clock valuations, which is also stable with respect to the various operations needed for exploration of the behavior of timed automata (in particular the operations of delay and reset).

In the setting of linearly priced timed automata, we put forward a new extended notion of *priced region*. Besides providing a finite partitioning of the set of clock-valuations (as in the case of ordinary regions), priced regions also associate costs to each individual clock-valuation within the region. However, as we shall see in the following, priced regions may be presented and manipulated in a symbolic manner and are thus suitable as an algorithmic basis.

Definition 3 (Priced Regions). *Given set S, let $Seq(S)$ be the set of finite sequences of elements of S. A* priced clock region *over a finite set of clocks C*

$$R = (h, [r_0, \dots, r_k], [c_0, \dots, c_l])$$

is an element of $(C \to \mathbb{N}) \times Seq(2^C) \times Seq(\mathbb{N})$, with $k = l$, $C = \cup_{i \in \{0,\dots,k\}} r_i$, $r_i \cap r_j = \emptyset$ when $i \neq j$, and $i > 0$ implies that $r_i \neq \emptyset$.

Given a clock valuation $u \in \mathbb{R}^C$, and region $R = (h, [r_0, \dots, r_k], [c_0, \dots, c_k])$, $u \in R$ iff

1. *h and u agree on the integer part of each clock in C,*
2. *$x \in r_0$ iff $\mathsf{frac}(u(x)) = 0$,*
3. *$x, y \in r_i \Rightarrow \mathsf{frac}(u(x)) = \mathsf{frac}(u(y))$, and*
4. *$x \in r_i$, $y \in r_j$ and $i < j \Rightarrow \mathsf{frac}(u(x)) < \mathsf{frac}(u(y))$.*

For a priced region $R = (h, [r_0, \dots, r_k], [c_0, \dots, c_k])$ the first two components of the triple constitute an ordinary (unpriced) region $\hat{R} = (h, [r_0, \dots, r_k])$. The naturals $c_0, \dots, c_k$ are the costs, which are associated with the vertices of the closure of the (unpriced) region, as follows. We start in the left-most lower vertex of exterior of the region and associate cost c_0 with it, then move one time unit in the direction of set r_k to the next vertex of the exterior, and associate cost c_1 with that vertex, then move one unit in the direction of r_{k-1}, etc. In this way, the costs $c_0, \dots, c_k$, span a linear cost plane on the k-dimensional unpriced region.

The closure of the unpriced region R is the convex hull of the vertices. Each clock valuation $u \in R$ is a (unique) convex combination[5] of the vertices. Therefore the cost of u can be defined as the same convex combination of the cost in the vertices. This gives the following definition:

Definition 4 (Cost inside Regions). *Given priced region $R = (h, [r_0, \dots, r_k], [c_0, \dots, c_k])$ and clock valuation $u \in R$, the* cost *of u in R is defined as:*

$$\mathsf{cost}(u, R) = c_0 + \sum_{i=0}^{k-1} \mathsf{frac}(u(x_{k-i})) * (c_{i+1} - c_i)$$

[5] A linear expression $\sum a_i v_i$ where $\sum a_i = 1$, and $a_i \geq 0$.

where x_j is some clock in r_j. The minimal cost associated with R is $\mathsf{mincost}(R) = \mathsf{min}(\{c_0, \ldots, c_k\})$.

In Fig. 6 of Appendix A an example of a typical three-dimensional priced region is given.

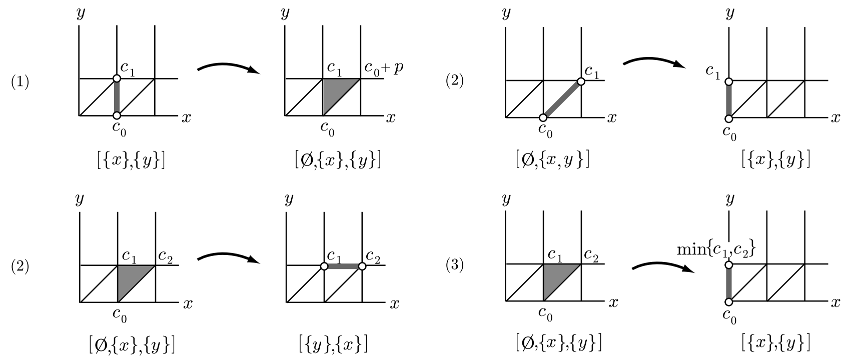

Fig. 4. Delay and reset operations for two-dimensional priced regions.

In the symbolic state-space, constructed with the priced regions, the costs will be computed such that for each concrete state in a symbolic state, the cost associated with it is the minimal cost for reaching that state by the symbolic path that was followed. In this way, we always have the minimal cost of all concrete paths represented by that symbolic path, but there may be more symbolic paths leading to a symbolic state in which the costs are different. Note that the cost of a clock valuation in the region is computed by adding fractions of costs for equivalence sets of clocks, rather than for each clock.

To prepare for the symbolic semantics, we define in the following a number of operations on priced regions. These operations are also the one used in the algorithm for finding the optimal cost of reaching a location.

The delay operation computes the time successor, which works exactly as in the classical (unpriced) regions. The changing dimensions of the regions cause the addition or deletion of vertex and thus of the associated cost.

The price argument will be instantiated to the price of the location in which time is passing; this is needed only when a vertex is added. The operation is illustrated in Fig. 4 (the operations on the left hand side, the numbers in brackets refer to the cases in the definition).

Definition 5 (Delay). *Given a priced region* $R = (h, [r_0, \ldots, r_k], [c_0, \ldots, c_k])$ *and a price* p*, the function* delay *is defined as follows:*

1. *If* r_0 *is not empty, then*

$$\mathsf{delay}(R, p) = (h, [\emptyset, r_0, \ldots, r_k], [c_0, \ldots, c_k, c_0 + p])$$

2. *If r_0 is empty, then*

$$\mathsf{delay}(R, p) = (h', [r_k, r_1, \dots, r_{k-1}], [c_1, \dots, c_k])$$
$$\text{where } h' = h \text{ incremented for all clocks in } r_k$$

When resetting a clock, a priced region may lose a dimension. If so, the two costs, associated with the vertices that are collapsed, are compared and the minimum is taken for the new vertex. The operation is illustrated in Fig. 4 (on the right hand side, where the numbers in brackets refer to the cases in the definition).

Definition 6 (Reset). *Given a priced region $R = (h, [r_0, \dots, r_k], [c_0, \dots, c_k])$ and a clock $x \in r_i$, the function* reset *is defined as follows:*

1. *If $i = 0$ then $\mathsf{reset}(x, R) = (h', [r_0, \dots, r_k], [c_0, \dots, c_k])$, where $h' = h$ with x set to zero*
2. *If $i > 0$ and $r_i \neq \{x\}$, then*

$$\mathsf{reset}(x, R) = (h', [r_0 \cup \{x\}, \dots, r_i \setminus \{x\}, \dots, r_k], [c_0, \dots, c_k])$$
$$\text{where } h' = h \text{ with } x \text{ set to zero}$$

3. *If $i > 0$ and $r_i = \{x\}$, then*

$$\begin{aligned}\mathsf{reset}(x, R) = (h', &[r_0 \cup \{x\}, \dots, r_{i-1}, r_{i+1}, \dots, r_k],\\ &[c_0, \dots, c_{k-i-1}, c', c_{k-i+2}, \dots, c_k])\\ \text{where } c' &= \min(c_{k-i}, c_{k-i+1})\\ h' &= h \text{ with } x \text{ set to zero}\end{aligned}$$

The reset operation on a set of clocks: $\mathsf{reset}(C \cup \{x\}, R) = \mathsf{reset}(C, \mathsf{reset}(x, R))$, *and* $\mathsf{reset}(\emptyset, R) = R$.

The price argument in the increment operation will be instantiated to the price of the particular transition taken; all costs are updated accordingly.

Definition 7 (Increment). *Given a priced region $R = (h, [r_0, \dots, r_k], [c_0, \dots, c_k])$ and a price p, the increment of R with respect to p is the priced region $R \oplus p = (h, [r_0, \dots, r_k], [c'_0, \dots, c'_k])$ where $c'_i = c_i + p$.*

If in region R, no clock has fractional part 0, then time may pass in R, that is, each clock valuation in R has a time successor and predecessor in R. When changing location with R, we must choose for each clock valuation u in R between delaying in the previous location until u is reached, followed by the change of location, or changing location immediately and delaying to u in the new location. This depends on the price of either location. For this the following operation self is useful.

Definition 8 (Self). *Given a priced region $R = (h, [r_0, \dots, r_k], [c_0, \dots, c_k])$ and a price p, the function* self *is defined as follows:*

1. *If r_0 is not empty, then* $\mathsf{self}(R, p) = R$.
2. *If r_0 is empty, then*

$$\mathsf{self}(R, p) = (h, [r_0, \ldots, r_k], [c_0, \ldots, c_{k-1}, c'])$$
$$\textit{where } c' = \min(c_k, c_0 + p)$$

Definition 9 (Comparison). *Two priced regions may be compared only if their unpriced versions are equal:* $(h, [r_0, \ldots, r_k], [c_0, \ldots, c_k]) \leq (h', [r'_0, \ldots, r'_{k'}], [c'_0, \ldots, c'_{k'}])$ *iff* $h = h'$, $k = k'$, *and for* $0 \leq i \leq k$: $r_i = r'_i$ *and* $c_i \leq c'_i$.

4 Symbolic Semantics and Algorithm

In this section, we provide a symbolic semantics for linearly priced timed automata based on the notion of priced regions and the associated operations presented in the previous section. As a main result we shown that the cost of an execution of the underlying automaton is captures sufficiently accruate. Finally, we present an algorithm based on priced regions. Proofs of the lemmas and theorems presented in this sectoin can be found in Appendix C.

Definition 10 (Symbolic Semantics). *The symbolic semantics of a LPTA A is defined as a transition system with the state space* $L \times ((C \to \mathbb{N}) \times \mathit{Seq}(2^C) \times \mathit{Seq}(\mathbb{N}))$, *with initial state* $(l_0, (h_0, [C], [0]))$ *(where h_0 assigns zero to the integer part of all clocks in C), and with the following transition relation:*

- $(l, R) \to (l, \mathsf{delay}(R, P(l)))$ *if* $\mathsf{delay}(R, P(l)) \in I(l)$.
- $(l, R) \to (l', R')$ *if there exists g, r such that* $l \xrightarrow{g,a,r} l'$, $R \in g$, $R' = \mathsf{reset}(R, r) \oplus P((l, g, a, r, l'))$ *and* $R' \in I(l')$.
- $(l, R) \to (l, \mathsf{self}(R, P(l)))$

In the remainder, states and executions of the symbolic transition system for LPTA A will be referred to as the symbolic states and executions of A.

Lemma 1. *Given LPTA A, for each execution α of A that ends in state (l, u), there is a symbolic execution β of A, that ends in symbolic state (l, R), such that $u \in R$, and* $\mathsf{cost}(u, R) \leq \mathsf{cost}(\alpha)$.

Lemma 2. *Whenever (l, R) is a reachable symbolic state and $u \in R$, then* $\mathsf{mincost}((l, u)) \leq \mathsf{cost}(u, R)$.

Combining the two lemmas we obtain as a main theorem that the symbolic semantics captures (sufficiently) accurate the cost of reaching states and locations:

Theorem 1. *Let l be a location of a LPTA A. Then*

$$\mathsf{mincost}(l) = \mathsf{min}(\{\, \mathsf{mincost}(R) \mid (l, R) \textit{ is reachable} \})$$

In Appendix B the region based symbolic state-space of the linearly priced timed automaton in Fig. 2 is shown.

The introduction of priced regions provides a first step towards an algorithmic solution for the minimum-cost reachability problem. However, in the present form both the integral part as well as the cost of vertices of priced regions may grow beyond any given bound during symbolic exploration. In the unpriced case, the growth of integral parts is often dealt with by suitable abstractions of (unpriced) regions, taking the maximal constant of the given timed automaton into account. Here we have chosen a very similar approach exploiting the fact, that any LPTA A may be transformed into an equivalent "*bounded*" LPTA $\tilde{A}$ in the sense that A and $\tilde{A}$ reaches the same locations with the exact same cost.

Theorem 2. *Let $A = (L, l_0, E, I, P)$ be a LPTA with maximal constant* max. *Then there exists a* bounded time equivalent *of A, $\tilde{A} = (L, l_0, E', I', P')$, satisfying the following:*

1. *Whenever (l, u) is reachable in $\tilde{A}$, then for all $x \in C$, $u(x) \leq \max + 2$.*
2. *For any location $l \in L$, l is reachable with cost c in A if and only if l is reachable with cost c in $\tilde{A}$*

Now, we suggest in Fig. 5 a branch-and-bound algorithm for determining the minimum-cost of reaching a given target location l_g from the initial state of a LPTA. All found states are stored in the two data structures PASSED and WAITING, divided into explored and unexplored states, respectively. The global variable COST stores the lowest cost for reaching the target location found so far. In each iteration, a state is taken from WAITING. If it matches the target location l_g and has a lower cost than the previously lowest cost COST, then COST is updated. Then, only if the state has not been previously explored with a lower cost do we add it to PASSED and add the successors to WAITING. This bounding of the search in line 8 of Fig. 5 may be optimized even further by adding the constraint $\mathsf{mincost}(R) < \text{COST}$; i.e. we only need to continue exploration if the minimum cost of the current region is below the optimal cost computed so far. Due to Theorem 1, the algorithm of Fig. 5 does indeed yield the correct minimum-cost value.

```
COST := ∞, PASSED := ∅, WAITING := {(l_0, R_0)}
while WAITING ≠ ∅ do
    select (l, R) from WAITING
    if l = l_g and mincost(R) < COST then
        COST := mincost(R)
    if for all (l, R') in PASSED: R' ⋠ R then
        add (l, R) to PASSED
        for all (l', R') such that (l, R) → (l', R'): add (l', R') to WAITING
return COST
```

Fig. 5. Branch-and-bound state-space exploration algorithm.

Theorem 3. *When the algorithm in Fig. 5 terminates, the value of* COST *equals* $\mathsf{mincost}(l_g)$.

For bounded LPTA, application of Higman's Lemma [11] ensures termination. In short, Higman's Lemma says that under certain conditions the embedding order on strings is a well quasi-order.

Theorem 4. *The algorithm in Fig. 5 terminates for any bounded LPTA.*

Finally, combining Theorem 3 and 4, it follows, due to Theorem 2, that the minimum-cost reachability problem is decidable.

Theorem 5. *The minimum-cost problem for LPTA is decidable.*

5 Conclusion

In this paper, we have successfully extended the work on regions and their operations to a setting of timed automata with linear prices on both transitions and locations. We have given the principle basis of a branch-and-bound algorithm for the minimum-cost reachability problem, which is based on an accurate symbolic semantics of timed automata with linear prices, and thus showing the minimum-cost reachability problem to be decidable.

The algorithm is guaranteed to be rather inefficient and highly sensitive to the size of constants used in the guards of the automata — a characteristic inherited from the time regions used in the basic data-structure of the algorithm. An obvious continuation of this work is therefore to investigate if other more (in practice) efficient data structures can be found. Possible candidates include data structures used in reachability algorithms of timed automata, such as DBMs, extended with costs on the vertices of the represented zones (i.e. convex sets of clock assignments). In contrast to the priced extension of regions, operations on such a notion of priced zones[6] can not be obtained as direct extensions of the corresponding operations on zones with suitable manipulation of cost of vertices.

The need for infimum in the definition of minimum cost executions arises from linearly priced timed automata with strict bounds in the guards, such as the one shown in Fig. 3 and discussed in Example 1. Due to the use of infimum, a linearly priced timed automaton is not always able to realize an execution with the exact minimum cost of the automata, but will be able to realize one with a cost (infinitely) close to the minimum value. If all guards include only non-strict bounds, the minimum cost trace can always be realized by the automaton. This fact can be shown by defining the minimum-cost problem for executions covered by a given symbolic trace as a linear programming problem.

In this paper we have presented an algorithm for computing minimum-costs for reachability of linearly priced timed automata, where prices are given as constants (natural numbers). However, a slight modification of our algorithm provides an extension to a parameterized setting, in which (some) prices may be

[6] In particular, the reset-operation.

parameters. In this setting, costs within priced regions will be finite collections, C, of linear expressions over the given parameters rather than simple natural numbers. Also, we are currently working on extending the algorithmic solution offered here to synthesis of minimum-cost controllers in the sense of [4]. In this extension, a priced region will be given by a conventional unpriced region together with a min-max expression over cost vectors for the vertices of the region. In both the parametric and the controller synthesis case, it follows from recent results in [1] (generalizing Higman's lemma) that the orderings on symbolic states are again well-quasi orderings, hence guaranteeing termination of our algorithms.

Acknowledgements

The authors would like to thank Lone Juul Hansen for her great, creative effort in making the figures of this paper. Also, the authors would like to thank Parosh Abdulla for sharing with us some of his expertise knowledge on the world beyond well-quasi orderings.

References

1. Parosh Aziz Abdulla and Aletta Nylén. Better is better than well: On efficient verification of infinite-state systems. In *Proc. of the 14*th *IEEE Symp. on Logic in Computer Science*. IEEE, 2000.
2. R. Alur, C. Courcoubetis, and T. A. Henzinger. Computing accumulated delays in real-time systems. In *Proc. of the 5th Int. Conf. on Computer Aided Verification*, number 697 in Lecture Notes in Computer Science, pages 181–193, 1993.
3. R. Alur and D. Dill. Automata for Modelling Real-Time Systems. *Theoretical Computer Science*, 126(2):183–236, April 1994.
4. E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *Hybrid Systems: Computation and Control*, number 1569 in Lecture Notes in Computer Science, pages 19–30. Springer–Verlag, March 1999.
5. Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Larsen, Paul Pettersson, and Judi Romijn. Efficient guiding towards cost-optimality in UPPAAL. Submitted.
6. Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A Model-Checking Tool for Real-Time Systems. In *Proc. of the 10th Int. Conf. on Computer Aided Verification*, number 1427 in Lecture Notes in Computer Science, pages 546–550. Springer–Verlag, 1998.
7. David Dill. Timing Assumptions and Verification of Finite-State Concurrent Systems. In J. Sifakis, editor, *Proc. of Automatic Verification Methods for Finite State Systems*, number 407 in Lecture Notes in Computer Science, pages 197–212. Springer–Verlag, 1989.
8. Ansgar Fehnker. Scheduling a steel plant with timed automata. In *Proceedings of the 6th International Conference on Real-Time Computing Systems and Applications (RTCSA99)*, pages 280–286. IEEE Computer Society, 1999.
9. T. A. Henzinger. The theory of hybrid automata. In *Proc. of 11th Annual Symp. on Logic in Computer Science (LICS 96)*, pages 278–292. IEEE Computer Society Press, 1996.

10. Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HyTech: A Model Checker for Hybird Systems. In Orna Grumberg, editor, *Proc. of the 9th Int. Conf. on Computer Aided Verification*, number 1254 in Lecture Notes in Computer Science, pages 460–463. Springer–Verlag, 1997.
11. G. Higman. Ordering by divisibility in abstract algebras. *Proc. of the London Math. Soc.*, 2:326–336, 1952.
12. Thomas Hune, Kim G. Larsen, and Paul Pettersson. Guided Synthesis of Control Programs Using Uppaal. In Ten H. Lai, editor, *Proc. of the IEEE ICDCS International Workshop on Distributed Systems Verification and Validation*, pages E15–E22. IEEE Computer Society Press, April 2000.
13. Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, October 1997.
14. Kim G. Larsen, Carsten Weise, Wang Yi, and Justin Pearson. Clock difference diagrams. *Nordic Journal of Computing*, 6(3):271–298, 1999.
15. J. Møller, J. Lichtenberg, H. R. Andersen, and H. Hulgaard. Difference decision diagrams. Technical Report IT-TR-1999-023, Department of Information Technology, Technical University of Denmark, Building 344, DK-2800 Lyngby, Denmark, February 1999.
16. Peter Niebert, Stavros Tripakis, and Sergio Yovine. Minimum-time reachability for timed automata. In *IEEE Mediteranean Control Conference*, 2000. Accepted for publication.
17. Peter Niebert and Sergio Yovine. Computing optimal operation schemes for multi batch operation of chemical plants. VHS deliverable, May 1999. Draft.

A Figure from Section 3

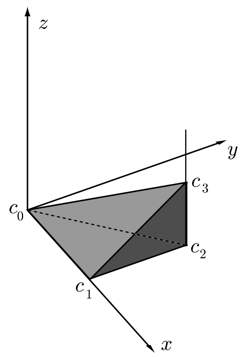

Fig. 6. A three-dimensional priced region.

B Example of Symbolic State-Space

In this appendix, we present part of the symbolic state space of the linearly priced timed automaton in Fig. 2 where the value of both α and β is two. Figures 7(i)-(viii) show some of the priced regions reachable in a symbolic representation of the states space. We only show the priced regions with integer value less than or equal to three.

Initially all three clocks have value zero and when delaying the clocks keep on all having the same value. Therefore the priced regions reachable from the initial state are the ones on the line from $(0, 0, 0)$ through $(3, 3, 3)$ shown in Fig. 7(i). The numbers on the line are the costs of the vertices of the priced regions represented by the line. Since the cost of staying in location A is one, the price of delay one time unit is one. Therefore the cost of reaching the point $(3, 3, 3)$ is three.

The priced regions presented in Fig. 7(ii) are the ones reachable after taking the transition to location B, resetting the x clock. Performing the reset does not change any of the costs, since the new priced regions are still one-dimensional and no vertices are collapsed. In Fig. 7(iii) the reachable priced regions are marked by a shaded area, including the lines inside the area and on the boundary. These priced regions are reachable from the priced regions in Fig. 7(ii).

Taking the transition from location B to location C causes clocks y and z to be reset. After resetting the priced regions in Fig. 7(iii), the priced regions in Fig. 7(iv) are reachable. Finding the cost of a state s after the reset (projection) is done by taking the minimum of the cost of the states projecting to s. When delaying from these priced regions, the priced regions in Fig. 7(v) are reached (again represented by the shaded area and the lines in and surrounding it).

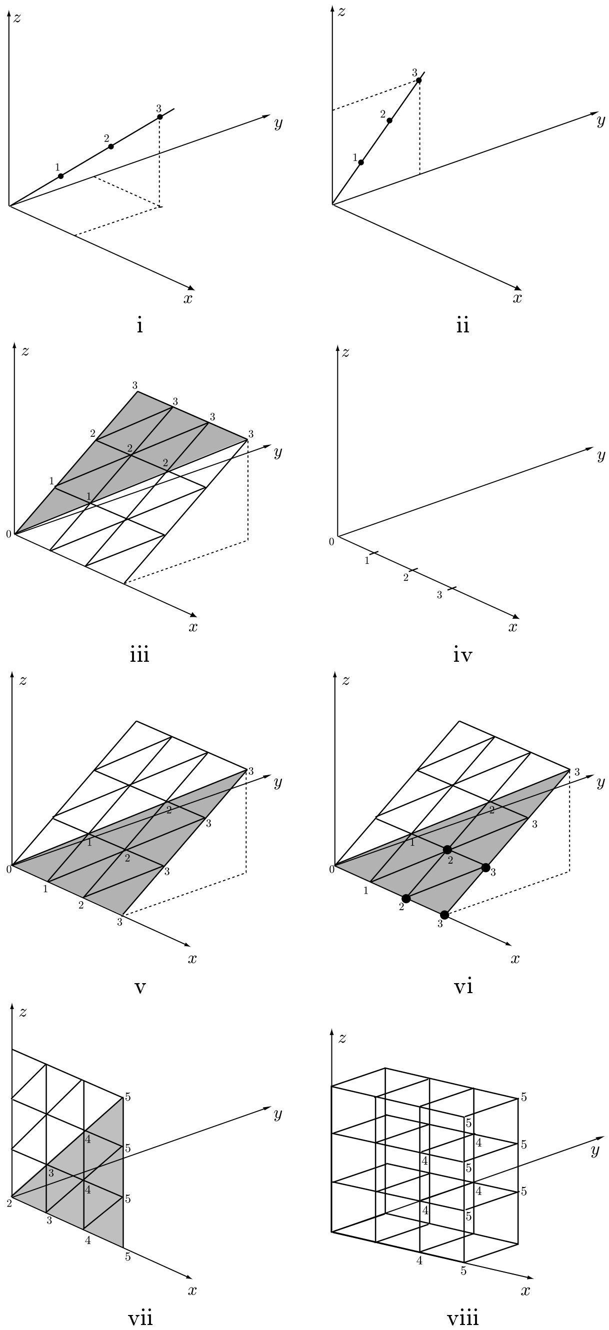

Fig. 7. Sets of reachable priced regions.

Now we are left with a choice; Either we can take the transition to location D, or take the loop transition back to location C. Taking the transition to location D is only possible if the guard $x \geq 2 \wedge y \leq 1$ is satisfied. Some of the vertices in Fig. 7(vi) are marked: only priced regions where all vertices are marked satisfy the guard. Before reaching location E from D with these priced regions, we must delay at least two time units to satisfy the guard $z \geq 3$ on the transition from location D to location E (this part of the symbolic state space is not shown in the figure). The minimum cost of reaching location E in this way is six.

The other possibility from location C is to take the loop transition which resets the y clock. After resetting y in the priced regions in Fig. 7(v), the priced regions in Fig. 7(vii) are reachable. From these priced regions we again can let time pass. However, a two dimensional picture of the three dimensional priced regions, reachable from the priced regions in Fig. 7(vii), is very hard to understand. Therefore, we have chosen to focus on the priced regions which satisfy the guard on the transition to location D. These priced regions are displayed by stating the cost of their vertices in Fig. 7(viii). The reachable priced regions satisfying the guard are the ones for which all vertices are marked with a cost in Fig. 7(viii). Three of the priced regions satisfying the guard on the transition from location C to location D, also satisfies the guard on the transition to location E. This is the two vertices where z has the value three and the line between these two points. The cost of reaching these points is five, so it is also possible to reach location E with this cost.

After taking the loop transition in location C once we also had the choice of taking it again. Doing this would yield the same priced regions as displayed in Fig. 7(vii) but now with two added to the cost. Therefore the new priced regions would be more costly than the priced regions already found and hence not explored by our algorithm.

C Proofs of Lemmas and Theorems

The following two propositions state a number of useful properties of the operations delay and self, which will be used throughout the proofs in this appendix.

Proposition 1 (Interaction Properties).

1. $\mathsf{self}(R,p) \leq R$,
2. $\mathsf{self}(\mathsf{self}(R,p),p) = \mathsf{self}(R,p)$,
3. $\mathsf{delay}(\mathsf{self}(R,p),p) \leq \mathsf{delay}(R,p)$,
4. $\mathsf{self}(\mathsf{delay}(R,p),p) = \mathsf{delay}(R,p)$,
5. $\mathsf{self}(R \oplus q,p) = \mathsf{self}(R,p) \oplus q$,
6. $\mathsf{delay}(R \oplus q,p) = \mathsf{delay}(R,p) \oplus q$,
7. *For* $g \in \mathcal{B}(C)$*, whenever* $R \in g$ *then* $\mathsf{self}(R,p) \in g$.

Proof. The proofs follow directly from the definitions of the operators and $\leq$. □

Stated in terms of the cost, $\mathsf{cost}(u,R)$, of an individual clock valuation, u, of a priced region, R, the symbolic operations behave as follows:

Proposition 2 (Cost Relations).

1. *Let* $R = (h, [r_0, \ldots, r_k], [c_0, \ldots, c_k])$. *If* $u \in R$ *and* $u + d \in R$ *then* $\mathsf{cost}(u + d, R) = \mathsf{cost}(u, R) + d * (c_k - c_0)$.
2. *If* $R = \mathsf{self}(R, p)$, $u \in R$ *and* $u + d \in \mathsf{delay}(R, p)$ *then* $\mathsf{cost}(u + d, \mathsf{delay}(R, p)) = \mathsf{cost}(u, R) + d * p$.
3. $\mathsf{cost}(u, \mathsf{reset}(x, R)) = \mathsf{inf}\{\, \mathsf{cost}(v, R) \mid [x \mapsto 0]v = u \,\}$.

Proof. The proofs follow directly from the definitions of the operators and cost. □

Lemma 1. *Given LPTA* A*, for each execution* α *of* A *that ends in state* (l, u)*, there is a symbolic execution* β *of* A*, that ends in symbolic state* (l, R)*, such that* $u \in R$*, and* $\mathsf{cost}(u, R) \leq \mathsf{cost}(\alpha)$.

Proof. For this proof we first observe that, given $g \in \mathcal{B}(C)$, if $u \in R$ and $u \in g$, then $R \in g$.

By induction on the length of α. Suppose α ends in state (l, u). The base step concerns α with length 0, consisting of only the initial state (l_0, u_0) where u_0 is the valuation assigning zero to all clocks. Clearly, $\mathsf{cost}(\alpha) = 0$. Since the initial state of the symbolic semantics is the state $(l_0, (h_0, [C], [0]))$, in which h_0 assigns zero to the integer part of all clocks, and the fractional part of all clocks is zero, we can take β to be $(l_0, (h_0, [C], [0]))$. Clearly, there is only one valuation $u \in (h_0, [C], [0])$, namely the valuation u that assigns zero to all clocks, which is exactly u_0, and by definition, $\mathsf{cost}(u_0, (h_0, [C], [0])) = 0$ and trivially $0 \leq 0$.

For the induction step, assume the following. We have an execution α' in the concrete semantics, ending in (l', u'), a corresponding execution β' in the symbolic semantics, ending in (l', R'), such that $u' \in R'$, and $\mathsf{cost}(u', R') \leq \mathsf{cost}(\alpha')$.

Suppose $(l', u') \xrightarrow{a,p} (l, u)$. Then there is a transition $l' \xrightarrow{a,g,r} l$ in the automaton A such that $u \in g$, $u = [r \mapsto 0]u'$, $u \in I(l)$ and $p = P((l', a, g, r, l))$. Now $u' \in g$ implies that $R' \in g$. Let $R = \mathsf{reset}(R', r) \oplus p$. It is easy to show that $u = [r \mapsto 0]u' \in R$ and as $u \in R$ we then have that $R \in I(l)$. So $(l', R') \to (l, R)$ and

$$\begin{aligned}
\mathsf{cost}(u, R) &= \mathsf{inf}\{\, \mathsf{cost}(v, R') \mid [r \mapsto 0]v = u \,\} \, + \, p \\
&\leq \mathsf{cost}(u', R') + p \\
&\leq^{\text{IH}} \mathsf{cost}(\alpha') + p \\
&= \mathsf{cost}(\alpha)
\end{aligned}$$

Suppose $(l', u') \xrightarrow{\epsilon(d), p * d} (l, u)$, where $p = P(l)$, i.e. $l = l'$ and $u' = u + d$. Now there exist sequences $R_o, R_1, \ldots, R_m$ and $d_1, \ldots, d_m$ of price regions and delays such that $d = d_1 + \cdots + d_m$, $R_0 = R'$ and for $i \in \{1, \ldots, m\}$, $R_i = \mathsf{delay}(R_{i-1}, p)$ with $u' + \sum_{k=1}^{i} d_k \in R_i$. Now let $R_0' = \mathsf{self}(R_0, p)$ and for $i \in \{1, \ldots, m\}$ let $R_i' = \mathsf{self}(R_{i-1}', p)$ (in fact, for $i \in \{1, \ldots, m\}$, $R_i' = \mathsf{self}(R_i', p)$ and $R_i' \leq R_i$). Clearly we have the following symbolic extension of β':

$$\beta' \to (l', R') \to (l', R_0') \to \cdots \to (l', R_m')$$

Now by Proposition 2.2 (the condition of Proposition 2.2 is satisfied for all $R'_i (i \geq 0)$ because of Proposition 1.4:

$$\begin{aligned}
\mathsf{cost}(u' + d, R'_m) &= \mathsf{cost}(u', R'_0) + d * p \\
&\leq \mathsf{cost}(u', R') + d * p \\
&\leq^{\text{IH}} \mathsf{cost}(\alpha') + d * p \\
&= \mathsf{cost}(\alpha)
\end{aligned}$$

□

Lemma 2. *Whenever (l, R) is a reachable symbolic state and $u \in R$, then* $\mathsf{mincost}((l, u)) \leq \mathsf{cost}(u, R)$.

Proof. The proof is by induction on the length of the symbolic trace β reaching (l, R). In the base case, the length of β is 0 and $(l, R) = (l_0, R_0)$, where R_0 is the initial price region $(h_0, [C], [0])$ in which h_0 associates 0 with all clocks. Clearly, there is only one valuation $u \in R_0$, namely the valuation which assigns 0 to all clocks. Obviously, $\mathsf{mincost}((l_0, u_0)) = 0 \leq \mathsf{cost}(u_0, R_0) = 0$.

For the induction step, assume that (l, R) is reached by a trace β with length greater than 0. In particular, let (l', R') be the immediate predecessor of (l, R) in β. Let $u \in R$. We consider three cases depending on the type of symbolic transition from (l', R') to (l, R).

Case 1: Suppose $(l', R') \to (l, R)$ is a symbolic delay transition. That is, $l = l'$, $R = \mathsf{delay}(R', p)$ with $p = P(l)$ and $R \in I(l)$. We consider two sub-cases depending on whether R' is self-delayable or not[7].
Assume that R' is not self-delayable, i.e. $R' = (h', [r'_0, \ldots, r'_k], [c'_0, \ldots, c'_k])$ with $r'_0 \neq \emptyset$. Then $R = (h', [\emptyset, r'_0, \ldots, r'_k], [c'_0, \ldots, c'_k, c'_0 + p])$. Let $x \in r'_0$ and let $u' = u - d$ where $d = \mathsf{frac}(u(x))$. Then $u' \in R'$ and $(l', u') \xrightarrow{\epsilon(d), q} (l, u)$ where $q = d * p$. Thus $\mathsf{mincost}((l, u)) \leq \mathsf{mincost}((l', u')) + d * p$. By induction hypothesis, $\mathsf{mincost}((l', u')) \leq \mathsf{cost}(u', R')$, and as $\mathsf{cost}(u, R) = \mathsf{cost}(u', R) + d * p$, we obtain, as desired, $\mathsf{mincost}((l, u)) \leq \mathsf{cost}(u, R)$.
Assume that R' is self-delayable. That is, $R' = (h', [r'_0, r'_1, \ldots, r'_k], [c'_0, \ldots, c'_k])$ with $r'_0 = \emptyset$ and $R = (h'', [r'_k, r'_1, \ldots, r'_{k-1}], [c'_1, \ldots, c'_k])$. Now, let $x \in r'_1$. Then for any $d < \mathsf{frac}(u(x))$ we let $u_d = u - d$. Clearly, $u_d \in R'$ and $(l, u_d) \xrightarrow{\epsilon(d), p*d} (l, u)$. Now,

$$\begin{aligned}
\mathsf{mincost}((l, u)) &\leq \mathsf{mincost}((l, u_d)) + p * d \\
&\leq^{\text{IH}} \mathsf{cost}(u_d, R') + p * d
\end{aligned}$$

Now $\mathsf{cost}(u, R) = \mathsf{cost}(u_d, R') + (c'_k - c'_0) * d$ so it is clear that $\mathsf{cost}(u_d, R') + k * d \to \mathsf{cost}(u, R)$ when $d \to 0$ for any k. In particular, $\mathsf{cost}(u_d, R') + p * d \to \mathsf{cost}(u, R)$ when $d \to 0$. Thus $\mathsf{mincost}((l, u)) \leq \mathsf{cost}(u, R)$ as desired.

[7] A priced region, $R = (h, [r_0, \ldots, r_k], [c_0, \ldots, c_k])$, is self-delayable whenever $r_0 = \emptyset$.

Case 2: Suppose $(l', R') \to (l, R)$ is a symbolic action transition. That is $R = \mathsf{reset}(R', r) \oplus p$ for some transition $l \xrightarrow{l,a,r} l'$ of the automaton with $R' \in g$ and $p = P(l)$. Now let $v \in R'$ such that $[r \mapsto 0]v = u$. Then clearly $(l, v) \xrightarrow{a,p} (l, u)$. Thus:

$$\begin{aligned} \mathsf{mincost}((l,u)) &\leq \inf\{\, \mathsf{mincost}((l,v)) \mid v \in R', [r \mapsto 0]v = u \,\} \\ &\leq^{\mathrm{IH}} \inf\{\, \mathsf{cost}(v, R') \mid [r \mapsto 0]v = u \,\} \\ &= \mathsf{cost}(u, R) \quad \text{by Proposition 2.3} \end{aligned}$$

Case 3: Suppose $(l', R') \to (l, R)$ is a symbolic self-delay transition. Thus, in particular $l = l'$. If $R = R'$ the lemma follows immediately by applying the induction hypothesis to (l', R'). Otherwise, R' is self-delayable and R' and R are identical except for the cost of the 'last' vertex; i.e. $R' = (h, [r_0, \ldots, r_k], [c_0, \ldots, c_{k-1}, c_k])$ and $R = (h, [r_0, \ldots, r_k], [c_0, \ldots, c_{k-1}, c_0 + p])$ with $r_0 = \emptyset$, $c_0 + p < c_k$ and $p = P(l)$. Now let $x \in r_1$. Then for any $d > u(x)$ we let $u_d = u - d$. Clearly, $u_d \in R$ (and $u_d \in R'$) and $(l, u_d) \xrightarrow{\epsilon(d), p*d} (l, u)$. Now:

$$\begin{aligned} \mathsf{mincost}((l,u)) &\leq \mathsf{mincost}((l,u_d)) + p * d \\ &\leq^{\mathrm{IH}} \mathsf{cost}(u_d, R') + p * d \end{aligned}$$

Now let $R'' = (h, [r_1, \ldots, r_k], [c_0, \ldots, c_{k-1}])$. Then $R = \mathsf{delay}(R'', p)$ and $R' = \mathsf{delay}(R'', c_k - c_0)$. Now $\mathsf{cost}(u_d, R') = \mathsf{cost}(u_{u(x)}, R'') + (c_k - c_0) * (d - u(x))$ which converges to $\mathsf{cost}(u_{u(x)}, R'')$ when $d \to u(x)$. Thus $\mathsf{cost}(u_d, R') + p * d \to \mathsf{cost}(u_{u(x)}, R'') + p * d = \mathsf{cost}(u, R)$ for $d \to u(x)$. Hence, as desired, $\mathsf{mincost}((l,u)) \leq \mathsf{cost}(u, R)$. □

Theorem 2. *Let $A = (L, l_0, E, I, P)$ be a LPTA where for each $x \in C$, $\mathsf{max}_A(x)$ is the maximal integer, used in any guard or invariant for x. Then there exists a* bounded time equivalent *of A, $\tilde{A} = (L, l_0, E', I', P')$, satisfying the following:*

1. *Whenever (l, u) is reachable in $\tilde{A}$, then for all $x \in C$, $u(x) \leq \mathsf{max}_A(x) + 2$.*
2. *For any location $l \in L$, l is reachable with cost c in A if and only if l is reachable with cost c in $\tilde{A}$*

Proof. We construct $\tilde{A} = (L, l_0, E \cup E', I', P')$, as follows. $E' = \{(l, x == \mathsf{max}_A(x){+}2, \tau, x := \mathsf{max}_A(x){+}1, l) | x \in C, l \in L\}$. For $l \in L$, $I'(l) = I(l) \bigwedge_{x \in C} x \leq \mathsf{max}_A(x) + 2$, $P'(l) = P(l)$. For $e \in (E \cup E')$, if $e \in E$ then $P'(e) = P(e)$ else $P'(e) = 0$.

By definition, $\tilde{A}$ satisfies the first requirement.

As to the second requirement. Let R be a relation between states from A and $\tilde{A}$ such that for $((l_1, u_1), (l_2, u_2)) \in R$ iff $l_2 = l_1$, and for each $x \in C$, if $u_1(x) \leq \mathsf{max}_A(x)$ then $u_2(x) = u_1(x)$, else $u_2(x) > \mathsf{max}_A(x)$. We show that for each state (l_1, u_1) of A which is reached with cost c, there is a state (l_2, u_2) of $\tilde{A}$, such that $((l_1, u_1), (l_2, u_2)) \in R$ and (l_2, u_2) is reached with cost c, and vice versa.

Let (l_1, u_1), (l_2, u_2) be states of A and $\tilde{A}$, respectively, We use induction on the length of some execution leading to (l_1, u_1) or (l_2, u_2).

For the base step, the length of such an execution is 0. Trivially, the cost of such an execution is 0, and if (l_1, u_1) and (l_2, u_2) are initial states, clearly $((l_1, u_1), (l_2, u_2)) \in R$.

For the transition steps, we first observe that for each clock $x \in C$, $u_1(x) \sim c$ iff $u_2(x) \sim c$ with $\sim \in \{<, \leq, >, \geq\}$ and $c \leq \mathsf{max}_A(x)$ $(*)$. Assume $((l_1, u_1), (l_2, u_2)) \in R$, and (l_1, u_1) and (l_2, u_2) can both be reached with cost c. We make the following case distinction.

Case 1: Suppose $(l_1, u_1) \xrightarrow{\epsilon(d),p}_A (l_1, u_1 + d)$. In order to let d time pass in (l_2, u_2), we have to alternatingly perform the added τ transition to reset those clocks that have reached the $\mathsf{max}_A(x) + 2$ bound as many times as needed, and then let a bit of the time pass. Let $d_1 \dots d_m$ be a sequence of delays, such that $d = d_1 + \dots + d_m$, and for $x \in C$ and $i \in \{1, \dots, m\}$, if $\mathsf{max}_A(x) + 2 - (u_1(x) + d_1 + \dots + d_{i-1}) \geq 0$ then $d_i \leq \mathsf{max}_A(x) + 2 - (u_1(x) + d_1 + \dots + d_{i-1})$, else $d_i \leq 1 - \mathsf{frac}(u_1(x))$. It is easy to see that for some u'_2, $(l_2, u_2)(\xrightarrow{\tau,0})^* \xrightarrow{\epsilon(d_1),p_1} \dots (\xrightarrow{\tau,0})^* \xrightarrow{\epsilon(d_m),p_m} (l_2, u'_2)$ where $p_i = d_i * P(l_2)$. The cost for reaching $(l_1, u_1 + d)$ is $c + d * P_A(l_1) = c + d * P_{\tilde{A}}(l_2) = c + (d_1 + \dots + d_m) * P_{\tilde{A}}(l_2)$, which is the cost for reaching (l_2, u'_2). Now, $((l_1, u_1 + d), (l_2, u'_2)) \in R$, because of the following. For each $x \in C$, If $u_1(x) > \mathsf{max}_A(x)$, then $u_2(x) > \mathsf{max}_A(x)$, and either x is not reset to $\mathsf{max}_A(x) + 1$ by any of the τ transitions, in which case still $u'_2(x) > \mathsf{max}_A(x)$, or x is reset by some of the τ transitions, and then $\mathsf{max}_A(x) + 1 \leq u'_2(x) \leq \mathsf{max}_A(x) + 2$, so $u'_2(x) > \mathsf{max}_A(x)$. If $u_1(x) \leq \mathsf{max}_A(x)$, then by $u_1(x) = u_2(x)$, $u_2(x) \leq \mathsf{max}_A(x)$. If $(u_1 + d)(x) \leq \mathsf{max}_A(x)$, then x is not touched by any of the τ transitions leading to (l_2, u'_2), hence $u'_2(x) = u_2(x) + d_1 + \dots + d_m = u_2(x) + d = (u_1 + d)(x)$. If $(u_1 + d)(x) > \mathsf{max}_A(x)$, then x may be reset by some of the τ transitions leading to (l_2, u'_2). If so, then $\mathsf{max}_A(x) + 1 \leq u'_2(x) \leq \mathsf{max}_A(x) + 2$, so $u'_2(x) > \mathsf{max}_A(x)$. If not, then $u'_2(x) = u_2(x) + d_1 + \dots + d_m = u_2(x) + d = (u_1 + d)(x) > \mathsf{max}_A(x)$.

Case 2: Suppose $(l_2, u_2) \xrightarrow{\epsilon(d),p}_A (l_2, u_2 + d)$. Then trivially $((l_1, u_1 + d), (l_2, u_2 + d)) \in R$. Now we show $(l_1, u_1) \xrightarrow{\epsilon(d),p}_A (l_1, u_1 + d)$. Since $(l_2, u_2 + d) \in I_{\tilde{A}}$, since $I_{\tilde{A}}$ implies I_A and since $((l_1, u_1 + d), (l_2, u_2 + d)) \in R$, from observation $(*)$ it follows that $(l_1, u_1 + d) \in I_A$. So $(l_1, u_1) \xrightarrow{\epsilon(d),p}_A (l_1, u_1 + d)$, and trivially, the cost of reaching $(l_2, u_2 + d)$ is $c + d * P_{\tilde{A}}(l_2) = c + d * P_A(l_1)$, which is the cost of reaching $(l_1, u_1 + d)$.

Case 3: Suppose $(l_1, u_1) \xrightarrow{a,p}_A (l'_1, u'_1)$. Let (l, g, a, r, l') be a corresponding edge. Then $p = P_A((l, g, a, r, l'))$. By definition, $(l, g, a, r, l') \in E_{\tilde{A}}$ and $P_{\tilde{A}}((l, g, a, r, l')) = P_A((l, g, a, r, l'))$. From observation $(*)$ it follows that $u_1 \in g$ implies $u_2 \in g$. It is easy to see that for $x \in r$, $u'_1(x) = 0 = u_2[r \mapsto 0](x)$, and for $x \notin r$, $u'_1(x) = u_1(x)$ and $u_2(x) = u_2[r \mapsto 0](x)$, so $((l'_1, u'_1), (l', u_2[r \mapsto 0])) \in R$. Combining this with observation $(*)$ it follows that $u_1[r \mapsto 0] \in I_A(l')$ implies $u_2[r \mapsto 0] \in I_{\tilde{A}}(l')$, hence $(l_2, u_2) \xrightarrow{a,p}_{\tilde{A}} (l', u_2[r \mapsto 0])$. Clearly, the cost of reaching (l_1, u'_1) is $c + d * P_{\tilde{A}}((l, g, a, r, l')) = c + d * P_A((l, g, a, r, l'))$, which is the cost of reaching $(l_2, u_2[r \mapsto 0])$.

Case 4: Suppose $(l_2, u_2) \xrightarrow{a,p}_{\tilde{A}} (l_2', u_2')$. Let (l, g, a, r, l') be a corresponding edge. If $(l, g, a, r, l') \in E_A$, then the argument goes exactly like in the previous case. If $(l, g, a, r, l') \notin E_A$, then $a = \tau$, $p = 0$, $l_2' = l' = l = l_2$, and $x \in r$ implies $u_2'(x) = \mathsf{max}_A(x) + 1$ and $u_2(x) = \mathsf{max}_A(x) + 2$. Since the cost of reaching (l_2', u_2') is $c + 0 = c$, it suffices to show $((l_1, u_1), (l_2, u_2')) \in R$. For $x \notin r$, this follows trivially. For $x \in r$, $u_2(x) = \mathsf{max}_A(x) + 2$, so $u_1(x) > \mathsf{max}_A(x)$ and by $u_2'(x) = \mathsf{max}_A(x) + 1$ we have $u_2'(x) > \mathsf{max}_A(x)$.

Theorem 3. *When the algorithm in Figure 5 terminates, the value of* COST *equals* $\mathsf{mincost}(l_g)$.

Proof (by contradiction). First, notice that if (l_1, R_1) can reach (l_2, R_2), then a state (l_1, R_1'), where $R_1' \leq R_1$, can reach a state (l_2, R_2'), such that $R_2' \leq R_2$. We prove that COST equals $\min\{\mathsf{mincost}(R) \mid (l_g, R) \text{ is reachable}\}$. Assume that this does not hold. Then there exists a reachable state (l_g, R) where $\mathsf{mincost}(R) <$ COST. Thus the algorithm must at some point have discarded a state (l', R') on the path to (l_g, R). This can only happen in line 8, but then there must exist a state $(l', R'') \in$ PASSED, where $R'' \leq R'$, encountered in a prior iteration of the loop. Then, there must be a state (l_g, R''') reachable from (l', R''), and COST $\leq \mathsf{mincost}(R''') \leq \mathsf{mincost}(R)$, contradicting the assumption. The theorem now follows from Theorem 1. □

Theorem 4. *The algorithm in Figure 5 terminates for any bounded LPTA.*

Proof. Even if A is bounded (and hence yields only finitely many unpriced regions), there are still infinitely many priced regions, due to the unboundedness of cost of vertices. However, application of Higman's lemma ensures that one cannot have an infinite sequence $\langle (c_1^i, \ldots, c_m^i) : 0 \leq i < \infty \rangle$ of cost-vectors (for any fixed length m) without $c_l^j \leq c_l^k$ for all $l = 1, \ldots, m$ for some $j < k$. Consequently, due to the finiteness of the sets of locations and unpriced regions, it follows that one cannot have an infinite sequence $\langle (l_i, R_i) : 0 \leq i < \infty \rangle$ of symbolic states without $l_j = l_k$ and $R_j \leq R_k$ for some $j < k$, thus ensuring termination of the algorithm. □