Undecidability of Cost-Bounded Reachability in Priced Probabilistic Timed Automata*

Jasper Berendsen[1], Taolue Chen[2], and David N. Jansen[1]

[1] Radboud University Nijmegen, Institute for Computing and Information Sciences, Nijmegen, The Netherlands
[2] CWI, Department of Software Engineering, Amsterdam, The Netherlands

Abstract. Priced Probabilistic Timed Automata (PPTA) extend timed automata with cost-rates in locations and discrete probabilistic branching. The model is a natural combination of Priced Timed Automata and Probabilistic Timed Automata. In this paper we focus on cost-bounded probabilistic reachability for PPTA, which determines if the maximal probability to reach a goal location within a given cost bound (and time bound) exceeds a threshold $p \in (0, 1]$. We prove undecidability of the problem for simple PPTA in 3 variants: with 3 clocks and stopwatch cost-rates or strictly positive cost-rates. Because we encode a 2-counter machine in a new way, we can also show undecidability for cost-rates in $\mathbb{Z}$ and only 2 clocks.

1 Introduction

Digital technology has been widely deployed in safety-critical situations and real-life environments, which leads to increased interest in computer systems that satisfy quantitative timing constraints. Timed automata [1] are a prominent and well-established formalism for modeling, analysis and verification of such *real-time* systems, which have received much attention both in terms of theoretical and practical developments.

In addition to computation time, systems also use other finite resources, e. g. energy, memory, or bandwidth. In many cases, some resources are scarce; the system should not use more resources than a certain budget. *Priced (or weighted) timed automata* [2,3] model resource use and resource constraints.

Traditional approaches to the formal description of real-time systems usually express the system model purely in terms of nondeterminism. However, many real-life systems, such as multimedia equipment, communication protocols and networks, exhibit random behavior. Thus we may ask for the likelihood that certain properties are satisfied. This suggests the study of *probabilistic* models. In this paper, we investigate *priced probabilistic timed automata* (PPTAs) [4], which are a probabilistic extension of priced timed automata. This model is an orthogonal extension of priced as well as probabilistic timed automata [5].

* Research supported by NWO/EW project 612.000.103 FRAAI, the Dutch Bsik project BRICKS. and the European Community's 7th Framework Programme No. 214755 (QUASIMODO).

J. Chen and S.B. Cooper (Eds.): TAMC 2009, LNCS 5532, pp. 128–137, 2009.

One of the most fundamental problems for timed automata and their variants is reachability. In the setting of PPTA, *cost-bounded probabilistic reachability* asks: "Is it possible to reach a goal state with probability $\geq p$ within a given cost (and time) bound?" This problem has been studied in [4], where the authors provided a *semi-algorithm:* If the answer is affirmative or the symbolic state space is finite, the algorithm terminates; however, the decidability of the problem remained open. In this paper, we show its *undecidability.* The proof reduces a 2-counter machine to a PPTA with three clocks; the 2-counter machine does not terminate iff some state in the PPTA is reachable with probability 1. Moreover, the PPTA can be restricted to: 1. either only cost-rates $\in \{0, 1\}$ or only cost-rates > 0, 2. no difference constraints nor strict constraints, and 3. no probabilistic resets. So, even when cost must increase with time passing, it may be necessary for the semi-algorithm of [4] to investigate infinitely many symbolic states. Undecidability also holds for PPTA with two clocks that allow cost-rates $\in \mathbb{Z}$.

Related Work. Although greatly inspired by [6], there are some thorough changes in our encoding of a 2-counter machine. First, we use a single clock to encode both counters, similar to [7]. We find our encoding simpler, since it uses the third clock only in one subautomaton. Second, [6] shows undecidability in the setting of the logic WCTL on priced timed automata, as well as in the setting of weighted timed games. In both settings the goal state is reached by simulating a terminating execution, or by doing a test after simulating an initial fragment of any execution. The 2-counter machine terminates iff the goal state cannot be avoided indefinitely. For our setting this would not work, because tests are entered probabilistically; the (now probabilistic) choice whether to continue simulation or do a test cannot avoid testing infinitely often.

The other way around, our undecidability results carry over to the setting of [6]. Our Theorem 1 shows a somewhat stronger result, since our PPTA forbid strict guards. Theorem 2 shows a new result on only two clocks, while often three clocks are necessary. Because of the strictly positive cost-rates, Theorem 3 also gives new insight in the setting of [6]. The innovative construction for Theorem 3 ensures that the time to reach the goal state is always 9 time units; it uses the third clock to measure the runtime. Theorem 2 also carries over to the game setting of [7] for two clocks and a lower bound.

Outlook. A possible continuation of this work is by having the slightly different notion of cost-bounded probabilistic reachability as in [4], namely to have $> p$ instead of $\geq p$ on reachability. The semi-algorithm in [4] does not necessarily terminate for $= p$ on probability. Can this crack between the two results be closed?

2 Preliminaries

A *probability distribution* over a finite set Q is a function $\mu : Q \to [0, 1]$ with $\sum_{q \in Q} \mu(q) = 1$. For set Q', let $\mathrm{Dist}(Q')$ be the set of distributions over finite subsets of Q'.

A *clock* is a real-valued variable that can be used to measure the elapse of time. A *clock valuation* is a mapping $\mathbb{X} \to \mathbb{R}_{\geq 0}$, assigning a value to each clock in some finite set $\mathbb{X}$. Let $\mathbb{R}^{\mathbb{X}}_{\geq 0}$ denote the set of all clock valuations. For $v \in \mathbb{R}^{\mathbb{X}}_{\geq 0}$ and $d \in \mathbb{R}_{\geq 0}$, let $v{+}d$ denote the clock valuation that maps each $x \in \mathbb{X}$ to $v(x) + d$. For $r \subseteq \mathbb{X}$, let $v[r{:=}0]$ denote the *reset* of the clocks in r, i. e. $v[r{:=}0](x)$ equals 0 if $x \in r$ and $v(x)$ otherwise. Valuation $v_{\mathsf{zero}} \in \mathbb{R}^{\mathbb{X}}_{\geq 0}$ assigns 0 to all clocks in $\mathbb{X}$.

A *zone* or *constraint* is a conjunction of non-strict inequalities where the value of a single clock is compared to an integer. Formally, for the set $\mathbb{X}$ of clocks the set $\mathit{Zones}(\mathbb{X})$ of zones Z is defined by the grammar: $Z ::= x \leq b \mid x \geq b \mid Z \wedge Z$, where $x \in \mathbb{X}$, $b \in \mathbb{N}$. Note that some other definitions [1] allow strict inequalities and inequalities on the difference between clocks, e. g. $x > 2$, $x - y < 3$.

2.1 Priced Probabilistic Timed Automata

The next definition a PPTA differs from [4] by: having no invariants, having only edges that incur cost 0, using our restricted notion of zones, and allowing negative cost-rates.

Definition 1. *A* PPTA *is a tuple* $(L, l_{\mathsf{init}}, \mathbb{X}, \mathsf{edges}, \dot{\$})$*, where* L *is a finite set of locations;* $l_{\mathsf{init}} \in L$ *is the initial location;* $\mathbb{X}$ *is a finite set of clocks;* $\mathsf{edges} \subseteq L \times \mathit{Zones}(\mathbb{X}) \times \mathrm{Dist}(2^{\mathbb{X}} \times L)$ *is a finite set of edges; and* $\dot{\$} : L \to \mathbb{Z}$ *associates a cost-rate with each location.*

For edge $(l, g, p) \in \mathsf{edges}$, l denotes the source location, g the guard (which is a zone), and p a distribution on pairs of a set of clocks to be reset and a destination location. Figure 1 shows a PPTA with clock x. The locations are represented by circles, with branching arrows between them denoting the edges of the PPTA. The initial location l_0 is marked with a dangling arrow. The cost-rates are written next to the locations. Guards (e. g. $x{\geq}1$) are next to the source location; the probabilities and resets are at the branches (e. g. probability 0.1 and $x{:=}\,0$.) Cost-rate 0, probability 1, and guards that always hold are omitted.

Intuitively, a PPTA behaves as follows. It always is in a state consisting of a location l, a clock valuation v and the amount of cost already incurred. A policy fills in the non-deterministic choice between the outgoing edges to take, or delaying. Only edges with guards satisfying the current valuation are available. Delaying will increase each clock by the amount of delay, and the accumulated cost by the amount of delay times the the cost-rate ($\dot{\$}(l)$). When taking an edge, one reset set and a destination location are chosen probabilistically, these clocks are reset and the system enters the destination.

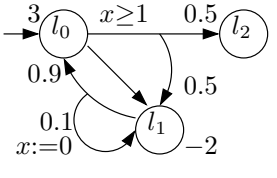

Fig. 1. Example PPTA

Definition 2. *A* Markov Decision Process *(MDP) is a tuple* $(S, s_{\mathsf{init}}, \mathsf{Act}, \pi)$*, where* S *is a set of states,* $s_{\mathsf{init}} \in S$ *is the initial state,* Act *is a set of action labels, and* $\pi \subseteq S \times \mathsf{Act} \times \mathrm{Dist}(S)$ *is a probabilistic transition relation such that for each* $s \in S$*, there exist* $a \in \mathsf{Act}$ *and* $\mu \in \mathrm{Dist}(S)$ *such that* $(s, a, \mu) \in \pi$*.*

A (in)finite *run* ω in an MDP $(S, s_{\mathsf{init}}, \mathsf{Act}, \pi)$ is a (in)finite sequence: $s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} s_2 \xrightarrow{a_2,\mu_2} \cdots$ such that $s_0 = s_{\mathsf{init}}$, $(s_i, a_i, \mu_i) \in \pi$, and $\mu_i(s_{i+1}) > 0$ for all i. Let ω_i denote the i-th state in the run ω, i. e. $\omega_i = s_i$. Let $\mathrm{last}(\omega)$ denote the last state in the finite run ω. A *policy* (also called scheduler, adversary, or strategy) is a function mapping every finite run ω in some MDP $(S, s_{\mathsf{init}}, \mathsf{Act}, \pi)$ to a pair $(a, \mu) \in \mathsf{Act} \times \mathrm{Dist}(S)$ such that $(\mathrm{last}(\omega), a, \mu) \in \pi$. For a policy A, let $Runs^A$ denote the set of all infinite runs that are induced by A. $Prob^A$ denotes the probability measure on $Runs^A$, defined using classical techniques [8].

Definition 3 (PPTA Semantics). *Given PPTA* $\mathsf{Aut} = (L, l_{\mathsf{init}}, \mathbb{X}, \mathsf{edges}, \dot{\$})$*, its semantics is the MDP:* $\mathsf{MDP}(\mathsf{Aut}) = (S, (l_{\mathsf{init}}, v_{\mathsf{zero}}, 0), \mathbb{R}_{\geq 0}, \pi)$*, where* $S = L \times \mathbb{R}^{\mathbb{X}}_{\geq 0} \times \mathbb{R}$ *so that a state consists of a location, a clock valuation, and the accumulated cost; and* $((l, v, c), d, \mu) \in \pi$ *if one of the following conditions holds:*

- *time transitions:* $d > 0$ *and* $\mu(l, v + d, c + \dot{\$}(l)d) = 1$
- *discrete transitions:* $d = 0$ *and there exists* $(l, g, p) \in \mathsf{edges}$ *such that* $v \models g$ *and for any* $(l', v', c) \in S$*:* $\mu(l', v', c) = \sum_{r \subseteq \mathbb{X} \wedge v' = v[r:=0]} p(r, l')$

Definition 4 (CBPR). *Given PPTA* $\mathsf{Aut} = (L, l_{\mathsf{init}}, \mathbb{X}, \mathsf{edges}, \dot{\$})$, cost-bounded probabilistic reachability *asks the question: "It is possible to reach location* $l_{\mathcal{G}} \in L$ *with probability at least* $p \in (0, 1]$ *and with cost at most* $\kappa \in \mathbb{N}$*.", denoted* $\exists \mathsf{P}_{\geq p} \mathsf{F}^{\leq \kappa} l_{\mathcal{G}}$*. It holds iff there exists a policy* A *of* $\mathsf{MDP}(\mathsf{Aut})$ *such that*

$$Prob^A\{\omega \in Runs^A \mid \exists i \in \mathbb{N}. \omega_i \in \{l_{\mathcal{G}}\} \times \mathbb{R}^{\mathbb{X}}_{\geq 0} \times (-\infty, \kappa]\} \geq p$$

3 Undecidability Results

Our undecidability results hold for restricted PPTA, called *simple* PPTA.

Definition 5 (Simple PPTA). *We call a PPTA* $\mathsf{Aut} = (L, l_{\mathsf{init}}, \mathbb{X}, \mathsf{edges}, \dot{\$})$ simple *if the resolution of probabilities does not influence the set of clocks being reset:* $\forall (l, g, p) \in \mathsf{edges}. \exists r \in 2^{\mathbb{X}}. \forall r' \in 2^{\mathbb{X}}. \forall l' \in L. p(r', l') > 0 \implies r' = r$.

Theorem 1. *CBPR of simple PPTA with three clocks and* $\dot{\$} : L \to \{0, 1\}$ *(stopwatch cost) is undecidable.*

Theorem 2. *CBPR of simple PPTA is undecidable even with two clocks.*

Theorem 3. *CBPR of simple PPTA with three clocks and* $\dot{\$} : L \to \mathbb{N}_{>0}$ *(strictly positive cost-rates) is undecidable.*

Note that our definition of policy is *deterministic:* a run is mapped to exactly one distribution. There exist other classes of policies, for which the undecidability results will hold in case the class allows the deterministic policies we have used.

The rest of this work contains the proofs. Sections 3.1–3.6 give the proof of Theorem 1. Sections 3.7 and 3.8 prove Theorems 2 and 3, respectively.

3.1 Proof of Theorem 1

Definition 6. *A* 2-counter Minsky machine *[9] is a computational model, consisting of a finite sequence of instructions, labeled* $l_1, l_2, \ldots, l_H$*. Computation starts at* l_1 *and halts at* l_H*. Instructions* $l_1, \ldots, l_{H-1}$ *are of the following two types, where* $c \in \{a, b\}$ *is one of the counters:*

$$\textbf{increment } c \qquad l_i : c := c + 1;\ \texttt{goto}\ l_j;$$

$$\textbf{test-and-decrement } c \qquad l_i : \texttt{if}\ c = 0\ \texttt{then goto}\ l_k;\ \texttt{else}\ c := c - 1;\ \texttt{goto}\ l_j;$$

We will encode the halting problem for 2-counter Minsky machine $\mathcal{M}$ using a PPTA Aut with a special goal location $l_{\mathcal{G}}$ that satisfies the following property:

$$\exists \mathsf{P}_{\geq 1}\mathsf{F}^{\leq 8} l_{\mathcal{G}} \text{ holds for } \mathsf{Aut} \iff \neg(\mathcal{M} \text{ terminates})$$

Aut has one location for each instruction label $l_1, \ldots, l_H$. Each transition, when taken at the correct time, corresponds to the execution of one instruction. After the transition, a test may check whether the right edge was taken at the correct time. There is a unique policy that chooses the correct time and edge in every state and so simulates the execution of $\mathcal{M}$; we call it the *fulfilling policy.* Any other policy will fail some test, which implies that it misses $l_{\mathcal{G}}$ or the cost bound with positive probability. So, the fulfilling policy is the only one that may satisfy CBPR. However, if $\mathcal{M}$ terminates, it leads to l_H with positive probability, so the maximal probability to reach $l_{\mathcal{G}}$ is still < 1. It is well-known that termination of a 2-counter machine is undecidable, implying Theorem 1.

Aut uses only 3 clocks x, y, z; it is not simple, and it allows resets of the form $x{:=}y$, where clock x is set to the value of clock y. Section 3.6 shows how Aut can be changed to a simple PPTA with only resets to zero.

Upon entering location l_i (under the fulfilling policy), auxiliary clock $y = 0$, and the values of the counters a and b are encoded by x as: $x = 2^{-a} \cdot 3^{-b}$. A value for x uniquely determines a and b. Since both counters start at 0, we have initial location l_0 with an edge to l_1 guarded by $x = 1$ and reset $y{:=}0$.

CBPR for $p < 1$ (e. g., $\mathsf{Aut} \models \exists \mathsf{P}_{\geq 0.7}\mathsf{F}^{\leq c} l_{\mathcal{G}}$) is also undecidable. Just add a probabilistic choice to the edge from l_0: enter l_1 with probability p, and let the remaining probability of $1 - p$ go to l_H.

In the rest of this section we assume a uniform distribution on all edges, and a cost-rate of 0 in every location, unless a different cost-rate is explicitly given. We now discuss the subautomata needed to let the fulfilling policy simulate the 2-counter machine.

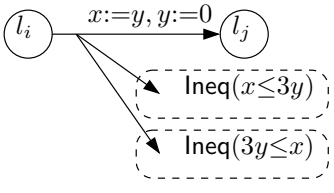

Fig. 2. Automaton for incrementing counter a

3.2 Increment Subautomata

Figure 2 shows the subautomaton for incrementing counter a. We denote the value of x and y upon entering l_i by x_i and y_i, respectively. Assume $x_i = 2^{-a} \cdot 3^{-b}$ (for

some $a, b \in \mathbb{N}$) and $y_i = 0$. (The fulfilling policy guarantees these assumptions.) The automaton ensures that under the fulfilling policy, the value upon entering l_j is $x_j = \frac{1}{2}x_i = 2^{-(a+1)} \cdot 3^{-b}$, which indeed encodes an increment on counter a: Let d_i be the time spent in l_i, and x_{Ineq}, y_{Ineq} be the values of the clocks upon entering the Ineq subautomata.

In subautomaton $\mathsf{Ineq}(\varphi)$, $l_{\mathcal{G}}$ is reachable with probability 1 within the cost bound only if φ holds and $0 \le y_{\mathsf{Ineq}} \le x_{\mathsf{Ineq}} \le 2$. Thus the fulfilling policy will only take the edge at a time when $x_{\mathsf{Ineq}} = 3y_{\mathsf{Ineq}}$. Now $y_{\mathsf{Ineq}} = d_i$ and $0 \le y_{\mathsf{Ineq}} \le x_{\mathsf{Ineq}}$ due to $y_i = 0$. Since $x_j = y_{\mathsf{Ineq}}$ due to reset $x{:=}y$, we have:

$$x_i = x_{\mathsf{Ineq}} - d_i = 3y_{\mathsf{Ineq}} - d_i = 3d_i - d_i = 2d_i = 2y_{\mathsf{Ineq}} = 2x_j \tag{1}$$

and $x_{\mathsf{Ineq}} = 3y_{\mathsf{Ineq}} = \frac{3}{2}x_i \le \frac{3}{2}$. The automaton for incrementing b is the same with the exception that we test for $x_i = 3x_j$ with $\mathsf{Ineq}(x \le 4y)$ and $\mathsf{Ineq}(4y \le x)$.

3.3 Power Subautomata

We now introduce an auxiliary automaton called $\mathsf{Power}(k)$. The fulfilling policy will multiply x with a power of $k \in \mathbb{N}$. In particular, a concatenation of $\mathsf{Power}(2)$ and $\mathsf{Power}(3)$ is used to check whether x has the form $2^{-a} \cdot 3^{-b}$: under the fulfilling policy x is doubled a times, leading to $x = 3^{-b}$, and then tripled b times, leading to $x = 1$. If x does not have the required form, it is impossible to reach $x = 1$.

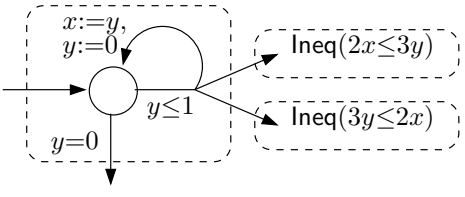

Fig. 3. $\mathsf{Power}(2)$: automaton for multiplying x a number of times by 2

Figure 3 shows $\mathsf{Power}(2)$. The number of times x is doubled is the number of times the loop is taken. The guard $y \le 1$ excludes a policy that always doubles and never takes the exit edge. Such a policy would pass a test with probability 1, because the probability to stay in the loop indefinitely is 0.

Let x^i be the value of x when entering the location for the i-th time. A similar argument as for Eq. 1 shows that $x^i = \frac{1}{2}x^{i+1}$. The power automaton can only be left with $x = x^i$ for some i, because of the guard $y = 0$, so x upon leaving the power automaton is $2^{i-1} \cdot x^1$. For $\mathsf{Power}(3)$ and $\mathsf{Power}(5)$ (used later), the corresponding tests are $3x = 4y$ and $5x = 6y$, respectively.

3.4 Decrement Subautomata

Figure 4 shows the subautomaton for test-and-decrement of counter a. In location l_i, the fulfilling policy takes the edge from l_i to l_k only if $a = 0$, because the test branch only succeeds if x_i has the form $2^0 \cdot 3^{-b}$. Below, we will see that the fulfilling policy takes the other edge only if $a > 0$.

Decrementing a is very similar to incrementing counters. With the same notations as in Sect. 3.2, assume $x_i = 2^{-a} \cdot 3^{-b}$ (for some $a, b \in \mathbb{N}$) and $y_i = 0$. Then, the Ineq subautomata ensure that the fulfilling policy lets $x_i = \frac{1}{2}x_j$.

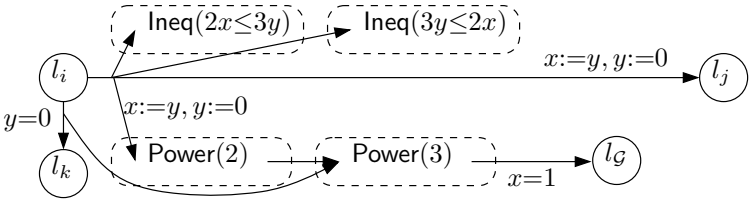

Fig. 4. Automaton for test-and-decrement of counter a

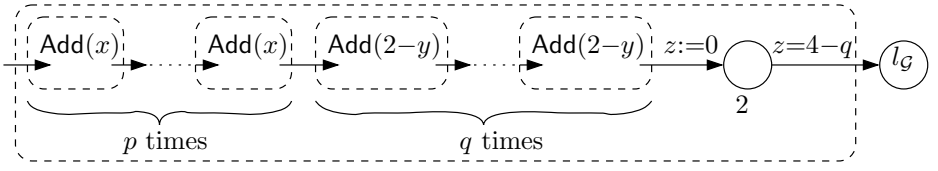

Fig. 5. Ineq($px \leq qy$): automaton for testing $px \leq qy$

The fulfilling policy will take the edge from l_i to l_j only if $a \neq 0$. Otherwise, assume the edge is taken while $a = 0$, then $x_i = 3^{-b}$. Recall that the fulfilling policy will ensure that $x_j = 2x_i = 2 \cdot 3^{-b}$. But then the branch leading from l_i to Power(2) will not reach $l_{\mathcal{G}}$, since it would have to divide x by 2.

The construction for test-and-decrement of counter b is very similar: on the edge from l_i to l_k, one would test for $x_i = 2^{-a} \cdot 3^0$, and on the edge from l_i to l_j, one would test for $3x = 4y$.

3.5 Ineq Subautomata

Figure 5 shows the Ineq($px \leq qy$) subautomaton. Location $l_{\mathcal{G}}$ is reached within the cost bound of 8 under a policy only if the clocks satisfy $px \leq qy$ and $0 \leq y \leq x \leq 2$ upon entering Ineq. Subautomata Add(x) and Add($2{-}y$) are used to add x respectively $2{-}y$ to the accumulated cost of a run under any policy that enters and exits the subautomaton, while all clocks have the same values upon exiting as upon entering. The accumulated cost when entering $l_{\mathcal{G}}$ is:

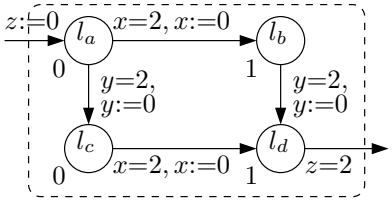

Fig. 6. Add(x): automaton adding x to the accumulated cost

$$px + q(2 - y) + (4 - q)2 = px - qy + 8$$

So $l_{\mathcal{G}}$ is reached within the cost bound of 8 only if $px \leq qy$.

Figure 6 depicts Add(x). The automaton has the same effect as in [6][1]. Subautomaton Add($2{-}x$) is easily obtained by swapping the cost-rates 0 and 1. The reader easily verifies the needed effect of passing through Add(x) or Add($2{-}x$).

[1] Add(x) in [6] contains a glitch: when $y = 1$ on entrance, possibly $y = 0$ on exit.

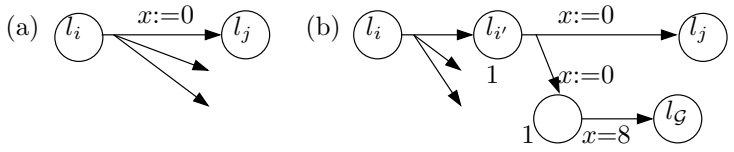

Fig. 7. Removing probabilistic resets

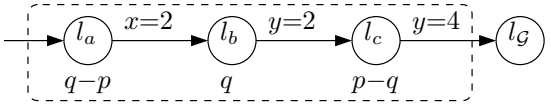

Fig. 8. $\mathsf{Ineq}(px \leq qy)$: automaton for testing $px \leq qy$

3.6 Adaption to Simple PPTA

To render the PPTA simple, we change the encoding as follows. The resets $x{:=}y, y{:=}0$ are replaced by $x{:=}0$. This swaps the role of x and y in the target location, i. e. y now encodes the counters. The fact that the clocks are swapped in some location will be captured by a copy of that location, where x and y are swapped on all guards and resets of outgoing edges.

The obtained PPTA still has resets $x{:=}0$ that depend on the resolution of probability. Figure 7a shows such an edge, and Fig. 7b shows how we can replace it using an intermediate location $l_{i'}$ and a reset that does not depend on the resolution of probability. The fulfilling policy will not let time advance in $l_{i'}$, because this incurs cost, and upon leaving $l_{i'}$, a test may be invoked to check whether the cost incurred up to that point is still 0.

3.7 Proof of Theorem 2

In this section, we allow PPTA to have any positive or negative integer cost rate. This relaxation will allow us to encode the 2-counter machine with two clocks only, because we can simplify the Ineq subautomata.

Figure 8 shows the alternative Ineq subautomaton. The cost-bound of the CBPR problem is changed to 0. (It may happen that a run exceeds the cost bound temporarily; however, upon entry into $l_{\mathcal{G}}$, its cost has to be ≤ 0.) Let d_a, d_b, d_c denote the time that elapses in locations l_a, l_b, l_c respectively. Let x_a, y_a, c_a denote the values of the clocks and accumulated cost when entering l_a. A run that reaches $l_{\mathcal{G}}$ has the following accumulated cost:

$$c_a + (q-p)d_a + qd_b + (p-q)d_c \tag{2}$$

Since all the locations a run visits before entering l_a have cost-rate 0 we have $c_a = 0$. We need to ensure that d_a, d_b, d_c are nonnegative (under the fulfilling policy). $d_a = 2 - x_a$, and non-negativity follows from the fact that when $x_a > 1$ the encoding of the counters is incorrect, which is only possible under a non-fulfilling policy. $d_b = 2 - (y_a + d_a) = x_a - y_a$, and non-negativity follows from the fact

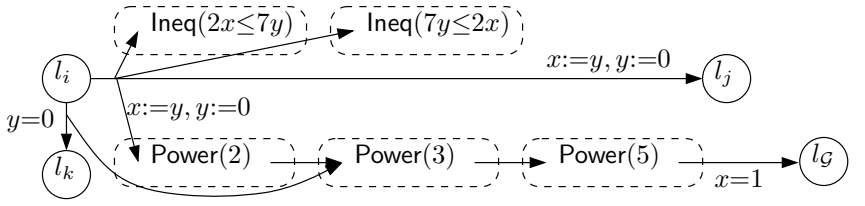

Fig. 9. New automaton for test-and-decrement of counter a

that $y_a \leq x_a$ whenever Ineq is entered. Clearly $d_c = 2$. By filling in Eq. 2 we get the following accumulated cost: $(q{-}p)(2{-}x_a) + q(x_a{-}y_a) + (p{-}q)2 = px_a - qy_a$. Therefore, Ineq$(px \leq qy)$ reaches $l_{\mathcal{G}}$ with cost ≤ 0 iff $px \leq qy$ upon entering.

3.8 Proof of Theorem 3

We now want to construct a simple PPTA with only strictly positive cost rates. As a starting point, we take the PPTA obtained in the previous section. We will again add a third clock z, but now, z is never reset, so it equals the duration of the run in all states.

The PPTA is adapted by adding 6 to all cost-rates. For all locations that had cost-rate 0 this clearly enforces a strictly positive cost-rate. The only negative cost-rates appear in Ineq subautomata (Fig. 8), but they are all larger than -6, so the new rates are all strictly positive.

All runs of the fulfilling policy that reach $l_{\mathcal{G}}$ should have an accumulated cost below the cost bound of the cost-bounded reachability problem. Because of the strictly positive cost-rate, we therefore need an overall time bound for all these runs, which we will show later to be 9. To accomodate the time bound, next to the counters a and b, clock x will encode the integer n, which is used to count the number of times a test-and-decrement instruction decremented any of the two counters. The encoding becomes: $x = 2^{-a} \cdot 3^{-b} \cdot 5^{-n}$.

The new test-and-decrement automaton is shown in Fig. 9. The values for the two Ineq automata are changed to accommodate that on entering l_j: $x_j = \frac{2}{5}x_i$ (which corresponds to decrementing a and incrementing n.) From Power(3) there is now an edge to Power(5) which has the edge guarded by $x = 1$ to $l_{\mathcal{G}}$. The Power(5) automaton is needed here, because this part was used to check the correctness of the encoding by x, which now includes the factor 5^{-n}.

The final change to the total automaton is that on every run where $l_{\mathcal{G}}$ was entered, the run now has to pass by a new location $l'_{\mathcal{G}}$. Figure 10 depicts $l'_{\mathcal{G}}$ and how from there $l_{\mathcal{G}}$ is reachable. Because z measures the duration of a run, which is bounded by 9 (as explained below), and time is spent in $l'_{\mathcal{G}}$ until z becomes 9, the additional cost for any run is: $9 \cdot 6 = 54$. Indeed the cost bound for the CBPR problem is changed to 54.

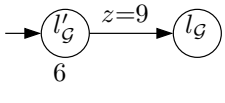

Fig. 10. Subautomaton to reach $l_{\mathcal{G}}$ in exactly 9 time units

We will now show that every run that enters $l'_{\mathcal{G}}$ has a duration bounded by 9. First of all 1 time unit is spent in l_0. Under the fulfilling policy, as long as

no Ineq or Power subautomaton is entered, every passage through an increment or decrement subautomaton multiplies x with $\frac{1}{2}$, $\frac{1}{3}$, $\frac{2}{5}$ or $\frac{3}{5}$, so the new value of x is at most $\frac{3}{5}$ times its old value. Further, the time spent in some subautomaton is equal to the new value of x. (If in a test-and-decrement subautomaton, the tested counter is $= 0$, then x is not changed and no time is spent in the subautomaton, so we can ignore this case in the runtime calculation.) Therefore, the total runtime until entering some Power or Ineq automaton is less than $1 + \sum_{i=1}^{\infty}(\frac{3}{5})^i = 2\frac{1}{2}$.

Similarly, one can see that each iteration in a concatenation of Power subautomata takes at most $\frac{1}{2}$ times the time of the next iteration, and the last iteration (all under the fulfilling policy) takes time 1. Therefore, the maximal time spent in Power subautomata is $\sum_{i=0}^{\infty}(\frac{1}{2})^i = 2$.

Finally, an Ineq subautomaton takes at most 4 time units. Summing up, we get a total upper bound on the runtime of $2\frac{1}{2} + 2 + 4 \leq 9$ upon entering $l'_{\mathcal{G}}$.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
2. Alur, R., Torre, S.L., Pappas, G.J.: Optimal paths in weighted timed automata. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 49–62. Springer, Heidelberg (2001)
3. Behrmann, G., Fehnker, A., Hune, T., Larsen, K.G., Pettersson, P., Romijn, J., Vaandrager, F.W.: Minimum-cost reachability for priced timed automata. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 147–161. Springer, Heidelberg (2001)
4. Berendsen, J., Jansen, D.N., Katoen, J.P.: Probably on time and within budget: On reachability in priced probabilistic timed automata. In: QEST, pp. 311–322. IEEE Computer Society Press, Los Alamitos (2006)
5. Kwiatkowska, M., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. Theoretical Computer Science 282(1), 101–150 (2002)
6. Bouyer, P., Brihaye, T., Markey, N.: Improved undecidability results on weighted timed automata. Inf. Process. Lett. 98(5), 188–194 (2006)
7. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Srba, J.: Infinite runs in weighted timed automata with energy constraints. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 33–47. Springer, Heidelberg (2008)
8. Kemeny, J., Snell, J., Knapp, A.: Denumerable Markov Chains, 2nd edn. Springer, New York (1976)
9. Minsky, M.L.: Computation: finite and infinite machines. Prentice-Hall, Inc., Upper Saddle River (1967)