Reachability in Weighted Probabilistic Timed Automata

J. K. Berendsen

Master of Science In Computer Science
Thesis

December 23, 2005

University of Twente
Faculty of EEMCS
Formal Methods and Tools group
Enschede, the Netherlands

Thesis Committee:
prof. dr. ir. J.P. Katoen
dr. D.N. Jansen
dr. ir. R. Langerak

Abstract

This thesis introduces the model of Weighted Probabilistic Timed Automata (WPTA). WPTA are an extension of the well known Timed Automata. With WPTA we can make a specification of systems that have discrete control, and continuous real-time behaviour. Moreover we can model cost-per-time, depending on the discrete state, and we can model probability on the discrete behaviour. An example of a system that can be modeled as WPTA is a task-scheduling problem with cost-per-time on resources, and resources being available with a certain probability.

The question is whether model checking, i. e. the automatic verification of certain properties, is possible on WPTA. Our research focused on methods for deciding whether a certain discrete state is reachable from the stating state, with cost below a certain boundary, and a certain minimal probability. Our research resulted in three algorithms. We assume a given cost-bound. The first algorithm is a backward algorithm that uses priced zones. It solves the problem when we are only interested in a positive probability instead of the exact value. The second algorithm constructs an upper bound on the probability for which the property holds. The third algorithm possibly does not terminate, but it generates an ascending sequence of values that converges to the maximum probability. In this way if a state exists such that the property holds, then we can conclude this. But if such a state does not exist we cannot give a verdict.

Samenvatting

Dit verslag introduceert het model: Weighted Probabilistic Timed Automata (WPTA). WPTA zijn een uitbreiding op de welbekende Timed Automata. Met behulp van WPTA kan een specificatie worden opgesteld van systemen met een discrete besturing die continu tijdsgedrag vertonen. Er kunnen kosten-per-tijd worden gemodelleerd, afhankelijk van de discrete toestand. Verder kunnen er kansen worden verbonden aan het discrete gedrag. Een voorbeeld van een systeem dat als WPTA gemodelleerd kan worden is een task-scheduling probleem waaraan kosten-per-tijd zijn verbonden aan resources en de resources met een zekere kans beschikbaar zijn.

De vraag is nu of model checking, dat wil zeggen automatische verficatie van zekere eigenschappen, mogelijk is op WPTA. Het onderzoek heeft zich beperkt tot methoden waarmee gekeken kan worden of een zekere discrete toestand bereikbaar is vanaf de begintoestand, waarbij de kosten beneden een bepaalde grens blijven met kans boven een bepaald minimum. Dit onderzoek heeft geresulteerd in drie algoritmen. We veronderstellen een vaste kostengrens. Het eerste algoritme dat gebaseerd is op terugwaardse exploratie en priced zones bepaalt of het met positieve kans mogelijk is. Het tweede algoritme vindt een bovengrens op de kans waarvoor het mogelijk is. Het derde algoritme termineert mogelijk niet, maar genereert een stijgende reeks van waarden die naar de maximum kans convergeren. Als er nu een toestand is die aan de randvoorwaarden voldoet, dan kunnen we dit met zekerheid concluderen. Echter als er niet zo'n toestand bestaat kunnen we geen uitspraak doen.

Preface

I am happy that I can hand over this thesis. The work on this thesis was the most challenging part of my study. I have learned a lot of new things, and practiced my mathematical skills more than ever. The results of my research were not clear beforehand, and a lot of attempts were made to solve the problems, of which not all are published in this report. Especially my ideas in the last sections, involved a lot of time, but in the end there was not enough time to complete them in a formal way. I am very grateful to Joost-Pieter and David, my supervisors and coaches. My meetings with them helped me a lot, and resulted in some good discussions. Joost-Pieter kept the general overview very well, including the time-schedule of the project, and constrained me in not diving in too much topics. David was always ready for me when I had some questions, helping me on the way when I was stuck. I would like to thank my fellow master students. Drinking tea with them was a welcome distraction. Furthermore I would like to thank my friends and fellow house mates for the necessary social distraction. Finally I would like to thank my family, my brothers Floris, Rikkert and Arnout, and my parents Evert and Louise, for their never faltering support and believe in me.

Enschede, December 23, 2005,
Jasper Berendsen

Contents

Chapter 1

Introduction

1.1 Reachability Analysis

System verification is checking if certain properties in a system hold. As some systems can get very complex we would like to do system verification in an automated way. To that end, both the system and the properties to be checked have to be formulated in a formal language. Model checking concerns the automatic verification of a formal model against formal properties.

There are a number of combinations of models and properties for which model checking is possible. An important property to be checked is if a certain state of the model, and thus of the system, is reachable from some other state. These properties are called reachability properties.

1.2 Formal System Models

A well studied formal model of systems are timed automata [AD94]. With timed automata it is possible to describe systems that have a notion of time. On timed automata model checking of reachability properties is possible. Timed automata can for example be used to model a number of tasks, as well as their commonly used resources, together with various time constraints. The question whether there exists a schedule that fulfills all requirements (such as ordering of the tasks, the timing constraints between tasks and the deadline(s)) can be formulated as a time-constrained reachability problem, and can be solved using model checkers such as Uppaal. Recently, this approach has been extended such that also costs can be treated [BFH$^+$01, ATP01]. Costs could model prices for the usage of resources, or the like. By means of cost-and-time constrained reachability one is able to answer the question: "Can a certain number of tasks be scheduled within a certain deadline with maximally cost κ?"

Another recent extension of timed automata is by including probabilities in the model [KNSS02]. Now reachability properties are extended to probabilistic reachability properties. A probabilistic reachability property for example states that a certain system state is reachable with probability λ or higher. Timed

automata with probabilities are suited to model protocols, because protocols have timing constraints and some probability of message loss.

1.3 A New Model

We define the new model of Weighted Probabilistic Timed Automata (WPTA), that extends the model of timed automata, by adding both costs and probability. The new model will be useful for scheduling problems where resources have prices on usage but also are not 100% available. Another application could be in protocols that consume different amounts of energy in different states of execution, where energy consumption can be modeled as cost.

In the model we want to check whether some system state is reachable with probability λ or higher, and cost at most κ. We call these *cost-bounded probabilistic reachability* properties. Our first step in solving the problem was to study [BFH+01] and [KNSS02]. In these papers, two problems related to ours are solved, namely: minimal cost reachability on timed automata with cost, and probabilistic reachability on timed automata with probabilities. At first, with a combination of these two approaches a solution to our problem seemed easy but this is not the case.

As the semantics of our model is described by an infinite structure, the next step concerned the construction of finite abstractions of this semantics. Finite abstractions based on the *priced regions* of [BFH+01], or the *priced zones* of [LBB+01], were investigated. Here the problem of adding both costs and probabilities becomes apparent. We will see that these kind of finite abstractions do not exist for WPTA, at least not for the problem investigated. We are, however, able to construct an algorithm that uses priced regions to compute an upper bound on the probability to reach a certain state with cost at most κ. We are also able to construct an efficient algorithm that decides (non-probabilistic) reachability of some state with cost at most κ, using backward analysis.

A new kind of abstraction with classes called *multi-priced zones* is introduced. Multi-priced zones are a subclass of multi-dimensional polyhedra. Multi-priced zones do not give a finite abstraction of the semantics, but with multi-priced zones we can use an altered version of the algorithm of [KNS03] to get an algorithm which iteratively approximates the cost-bounded *maximal* probabilistic reachability. Being able to approximate the maximum probability, we are able to give a partial solution to the problem of cost-bounded probabilistic reachability. The problem is solved partially in the sense that if the property holds the algorithm will give the answer "yes" after a finite number of iterations, but the algorithm can never given the answer "no" for certain. Finally we present some ideas on how to compute answer "no".

Chapter 2 contains the preliminaries for the rest of this thesis. It will introduce the model of timed automata, its variants with cost and probability, and reachability verification results on these models. Chapter 3 introduces the model of WPTA and its semantics. Finite abstractions for the infinite semantics are given, and some first easy results are presented. Moreover we present the new efficient backward algorithm to model check non-probabilistic reachability for

WPTA. Chapter 4 formalizes the property of cost-bounded probabilistic reachability. Next, we present our first naive algorithm based on priced regions, and prove its incorrectness by a counter-example. We analyze the problem and conclude with proving that the algorithm is capable of computing an upper bound on the probability. Chapter 5 presents our iterative algorithm, that returns a monotonous sequence that converges to the maximum on probability λ for which some cost-bounded probabilistic reachability property may still hold. As our algorithm may not converge, this solves the problem partially. The chapter finishes with some ideas on how to solve the problem fully. Finally this thesis ends with a conclusion, and directions for further research.

1.4 Related Work

The work most closely related to ours is that of J. Sproston. In [Spr00] model checking on probabilistic linear hybrid automata is presented. Reachability properties are part of their model checking properties, and WPTA are a subclass of probabilistic linear hybrid automata. The difference is that they only handle model checking on automata that have a finite bisimulation quotient, whereas WPTA in general do not have a finite bisimulation quotient.

Chapter 2

Preliminaries

A number of results and definitions from the literature are relevant for the problem investigated. Three models and important results on these models are introduced: timed automata, linear priced timed automata, and probabilistic timed automata.

2.1 Timed Automata

Timed automata are discrete transition systems extended with a notion of time. With timed automata it is possible to model systems that incorporate timing aspects. For timed automata there are a number of model checkers available, that can check properties like for example: "Within 5ms after receiving a packet, the system sends an acknowledgment," or "Every 3ms the system empties its buffer." Formalization of the properties to be checked is often done in some temporal logic (TCTL [ACD93]). Before giving a more detailed description of timed automata we need to give some definitions.

Clocks and clock valuations

A *clock* x is a variable that denotes time. Clocks can have a value from the positive reals including zero, denoted $\mathbb{R}_+$. Time is expressed in some unit of choice, which is unimportant for the functioning of the model. We consider $\mathbb{X}$ to be a finite set of clocks. All clocks in $\mathbb{X}$ increase with the same rate. This means that without reset, if one clock increases with say d time units then all clocks increase with d. A *clock valuation* is an assignment of values to all clocks. Thus a clock valuation is a mapping $v : \mathbb{X} \to \mathbb{R}_+$. The set of all clock valuations is called the *clock valuation space* and is denoted as $\mathbb{R}_+^{\mathbb{X}}$. Let v be a clock valuation. For $d \in \mathbb{R}_+$ ($u \in \mathbb{R}_+^{\mathbb{X}}$), let $v + d$ ($v + u$) denote the clock valuation that maps each $x \in \mathbb{X}$ to $v(x) + d$ ($v(x) + u(x)$). Define $v - d$ and $v - u$ analogously, but these may not be valid clock valuations, as they possibly map a clock to a negative value. For $r \subseteq \mathbb{X}$, let $v[r := 0]$ denote the *reset* of the

clocks in r such that:

$$v[r := 0](x) = \begin{cases} 0 & \text{if } x \in r \\ v(x) & \text{otherwise} \end{cases}$$

Clock valuation $\bar{0}$ is the special clock valuation that maps all clocks to zero, i. e. for all $x \in \mathbb{X}$, $\bar{0}(x) = 0$.

A set of clock valuations $V \subseteq \mathbb{R}_+^{\mathbb{X}}$ is called *bounded w. r. t. x* if there exists $a \in \mathbb{R}_+$ such that for all $v \in V$ we have that $v(x) \leq a$. When V is not bounded w. r. t. x it is called *unbounded w. r. t. x*. A set of clock valuations is called *bounded* (*unbounded*) if it is (not) bounded for all clocks.

Clock constraints

Clock constraints are conditions on the values of clocks. They are limited to the constraints in which a single clock or the difference between two clocks is compared to an integer. Furthermore, a constraint can have a conjunction of these comparisons which must all hold for the constraint to be met. Formally, for the set of clocks $\mathbb{X}$ the set Cons($\mathbb{X}$) of clock constraints ϕ is defined by the grammar:

$$\phi ::= x \bowtie b \mid x - y \bowtie b \mid \phi \wedge \phi \mid \mathit{true} \quad \text{where } x, y \in \mathbb{X}, b \in \mathbb{Z}, \bowtie \in \{<, \leq, \geq, >\}$$

A subset is formed by the *diagonal-free* clock constraints. These constraints do not have bounds on the difference between two clocks, and are given by the grammar:

$$\phi ::= x \bowtie b \mid \phi \wedge \phi \mid \mathit{true} \quad \text{where } x, y \in \mathbb{X}, b \in \mathbb{Z}, \bowtie \in \{<, \leq, \geq, >\}$$

Clock constraints can be *closed*, which means that all bounds are closed ($\bowtie \in \{\leq, \geq\}$). We write $v \in \phi$ if clock valuation v satisfies clock constraint ϕ.

Transition systems and paths

A *labeled transition system* is a triple $(S, \mathit{Act}, \rightarrow)$, with S a set of states, Act a finite set of (action) labels, and transition relation $\rightarrow \subseteq S \times \mathit{Act} \times S$.

A *path* in a labeled transition system is a sequence of transitions: $\omega = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$, such that $s_i \xrightarrow{a_i} s_{i+1}$ is a transition in the labeled transition system, for all $i \geq 0$. If ω is finite and the last state is s_n, then $|\omega| = n$ denotes the length of ω. A path of length zero contains only a single state. We write $\omega(i) = s_i$, and last(ω) for the last state of ω if it is finite.

A *timed transition system* is written as a triple $(S, \mathit{Act}, \rightarrow)$, and is the labeled transition system $(S, \mathit{Act} \cup \mathbb{R}_+, \rightarrow)$, where $\mathit{Act} \cap \mathbb{R}_+ = \emptyset$. Labels on the transitions are now elements of the set $\mathit{Act} \cup \mathbb{R}_+$. A transition $s \xrightarrow{a} t$, with $t, s \in S$, is called a *discrete transition* if $a \in \mathit{Act}$, and a *time transition* if $a \in \mathbb{R}_+$. For time transitions the following axioms [Sto02] must always hold:

time determinism: if $s \xrightarrow{d} t$ and $s \xrightarrow{d} t'$ for $d \in \mathbb{R}_+$ then $t = t'$,

Wang's Axiom: $s \xrightarrow{d} t$, with $d > 0$, if and only if there exists s' and $d' < d$ such that $s \xrightarrow{d'} s'$ and $s' \xrightarrow{d-d'} t$.

zero delay: $s \xrightarrow{0} t$ if and only if $s = t$. [1]

Forward & backward exploration

When we have a labeled transition system $LTS = (S, Act, \rightarrow)$, reachability analysis concerns answering the question whether some state in a set of target states is reachable from some starting state. There are two main methods to perform reachability analysis: forward exploration and backward exploration. Forward exploration starts with a set containing only the starting state. In operation of the procedure, the set always contains states that are reachable from the starting state. In steps, the set is extended with new states that are reachable from the set using some transition, until a target state is added. Backward exploration works in opposite direction. The set is initialized with the target states and is extended by states that can reach some state in the set, and thus the set contains only states that can reach the target. This continues until the starting state is added. Both forward and backward exploration could also work with a *set of* starting states.

If the transition system is given in advance then it is clear that, for a method to be computable, S must be finite. However if only the transition relation $\rightarrow$ is given, the states can be generated from the previously generated ones, and are contained in the set that both methods maintain. In this way only a subset of the state space is generated. For forward exploration $\rightarrow$ needs to be finitely branching, meaning that from a state there are only a finite number of transitions possible. With backward exploration each state should only be reachable by a finite number of transitions. Note that these conditions are always true when S is finite.

2.1.1 The model

A timed automaton is a discrete transition system, or equivalently a directed graph, with a set of clocks $\mathbb{X}$. The transitions take place between what are called *locations*. A state in a timed automaton is a pair consisting of a location and a clock valuation. Execution of a timed automaton starts in one location, with all clocks set to zero. In a location a certain amount of time units can be spent. This does not need to be a discrete amount but can be a real number. The effect of spending $d \in \mathbb{R}_+$ time units in a location is that all clocks are increased by d. Edges between locations are enabled if a constraint on the clocks, called the *guard*, is satisfied. Taking an edge induces no time, so none of the clocks is changed, unless the clock is reset to zero by the edge. An edge can reset any number of clocks. All locations have a constraint that is called

[1] This condition is needed because we allow $d = 0$.

an *invariant*. When taking edges and letting time progress, the invariant in the current location must hold.

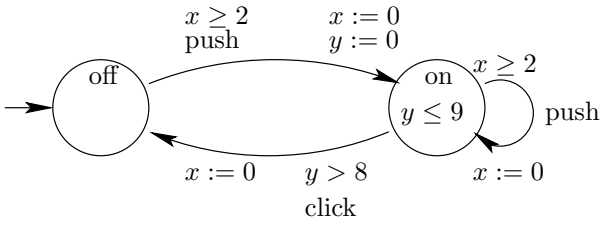
A timed automaton can be represented by a directed graph augmented by operations on the clocks. Figure 2.1 gives an example of such a representation. The timed automaton models a light switch. The automaton has two clocks x and y. The circles represent the locations of the timed automaton. The initial location 'off' is marked by a dangling incoming arrow. The invariants are written in the locations, or left out when there is no restriction (i. e. the invariant equals true). The guard and resets of a transition are written next to the edge. Guards are left out when the transition is always enabled.

In location 'off', the light is off. To turn the light on, the edge marked 'push' may be taken. This can only be done when the guard $x \geq 2$ is satisfied, thus at least 2 time units should be spent in location 'off'. When taking the edge, both clocks x, y are reset to zero. When the light is on, at most every 2 time units the button can be pushed again, as can be seen from the looping edge from location 'on'. The invariant in location 'on' makes that the light is on for at most 9 time units. The edge marked 'click' makes that the light may switch automatically off after more than 8 time units. Thus the light stays on for more than 8, but at most 9 time units.

We can write a possible execution as a path in a timed transition system, where states consists of the current location and the values of x and y. We have for example:

$$\begin{aligned}\rightarrow (\text{off}, 0, 0) &\xrightarrow{1.3} (\text{off}, 1.3, 1.3) \xrightarrow{1} (\text{off}, 2.3, 2.3) \xrightarrow{\text{push}} (\text{on}, 0, 0) \\ &\xrightarrow{4.3} (\text{on}, 4.3, 4.3) \xrightarrow{\text{push}} (\text{on}, 0, 4.3) \xrightarrow{4} (\text{on}, 4, 8.3) \xrightarrow{\text{click}} (\text{off}, 0, 8.3)\end{aligned}$$

Figure 2.1: An example timed automaton



Definition 2.1.1 A **Timed Automaton** [AD94] is a tuple $(L, l_0, \mathbb{X}, inv, E)$ with the following properties:

- L - finite set of locations.
- $l_0 \in L$ - single starting location.
- $\mathbb{X}$ - finite set of clocks.
- $inv : L \to \text{Cons}(\mathbb{X})$ - function assigning an invariant to each location.
- $E \subseteq L \times (\text{Cons}(\mathbb{X}) \times 2^{\mathbb{X}}) \times L$ - edges.

□

We shall write $l \xrightarrow{g,r} l'$ when $(l, g, r, l') \in E$. A timed automaton is said to be *diagonal-free*, if all its invariants and guards are diagonal-free. In the same way a timed automaton is *closed*, if all its invariants and guards are closed.

Definition 2.1.2 (Timed Automata Semantics) The semantics of a timed automaton $TA = (L, l_0, \mathbb{X}, inv, E)$ are given by a timed transition system $TTS = (S, Act, \to)$, where $Act = E, S = \{(l, v) \mid l \in L \wedge v \in inv(l)\}$. The *starting state* is $(l_0, \bar{0})$. Transitions are defined by letting time pass, or taking an edge:

time transitions: $(l, v) \xrightarrow{d} (l, v+d)$ if $v \in inv(l)$ and $(v+d) \in inv(l)$ for $d \in \mathbb{R}_+$,

discrete transitions: $(l, v) \xrightarrow{a} (l', v')$ if $a = (l \xrightarrow{g,r} l') \in E, v \in g, v' = v[r := 0]$ and $v' \in inv(l')$.

For a state s of TA, $loc(s)$ denotes its location. □

The definition of time transitions in Definition 2.1.2 is somewhat relaxed with respect to the regular definition that states: $(l, v) \xrightarrow{d} (l, v + d)$ for all $d' \leq d, (v + d') \in inv(l)$. Here, all time moments in the interval $[0, d]$ must satisfy the invariant. Note that our definition implies the traditional one, since the invariant can be seen as a convex polyhedron (see 2.1.9). Now if both v and $v + d$ satisfy the invariant, this implies that $v + d'$, for any $d' \leq d$ must satisfy the invariant.

2.1.2 Reachability analysis on timed automata

Reachability analysis on timed automata answers the question whether some location of a set of target locations is reachable from the starting state of the timed automaton. There are a number of methods for reachability analysis on timed automata. In the next sections we will discuss some different approaches. Due to the presence of real valued clocks the state space of a timed automaton, i. e. all possible states of its semantics, is uncountable. Infinite state systems are inadequate for automated verification. All approaches rely on a finite abstraction of the infinite semantics.

Region equivalence

[AD94]. Reachability analysis does not concern the values of the clocks in the target locations, but the values of the clocks are of importance for the behavior, as there are constraints on the clocks in a timed automaton. A *time-abstracting bisimulation* is an equivalence relation on the states in a timed transition system. Recall that a timed transition system is used to define the semantics of a timed automaton. A time-abstracting bisimulation abstracts away from the exact amount of time that elapses, as long as the equivalent states have the same results on discrete transitions. With a time abstracting bisimulation it is possible to construct an abstract transition system on the equivalence classes. There are several sorts of time-abstracting bisimulations [TY01], but this thesis always refers to *strong* time-abstracting bisimulations. Region equivalence is an equivalence relation defined on clock valuations and was first discovered by Alur and Dill [AD94]. Region equivalence is a used to define a special kind of finite time-abstracting bisimulation for timed automata.

Definition 2.1.3 [TY01]. Let $TTS = (S, Act, \rightarrow)$ be a timed transition system. A **(Strong) Time-Abstracting Bisimulation** of TTS is an equivalence relation $\simeq \subseteq S \times S$ such that for all $s_1, s_2 \in S, s_1 \simeq s_2$,

- when $s_1 \xrightarrow{a} s_1'$, there exists $s_2' \in S$ such that $s_2 \xrightarrow{a} s_2'$ and $s_1' \simeq s_2'$,
- when $s_1 \xrightarrow{d} s_1'$ with $d \in \mathbb{R}_+$, there exists $d' \in \mathbb{R}_+$ and $s_2' \in S$ such that $s_2 \xrightarrow{d'} s_2'$ and $s_1' \simeq s_2'$.

A time-abstracting bisimulation that has a finite number of equivalence classes is called *finite*. □

Definition 2.1.4 [TY01]. Let $TTS = (S, Act, \rightarrow)$ be a timed transition system, and $\simeq$ a time-abstracting bisimulation of TTS. $\simeq$ induces a labeled transition system $TTS/_{\simeq} = (S/_{\simeq}, Act \cup \{\epsilon\}, \rightarrow /_{\simeq})$, that is called a **Time-Abstracting Quotient**, with:

- $S/_{\simeq}$ is a partitioning of S containing the equivalence classes of $\simeq$,
- $\sigma \xrightarrow{a} /_{\simeq} \tau$ if for all $s \in \sigma$ there exists $t \in \tau$ such that $s \xrightarrow{a} t$,
- $\sigma \xrightarrow{\epsilon} /_{\simeq} \tau$ if for all $s \in \sigma$ there exists $d \in \mathbb{R}_+$ and $t \in \tau$ such that $s \xrightarrow{d} t$ and for all $s \xrightarrow{d'} t'$, with $d' < d$ we have that $t' \in \sigma \cup \tau$.

□

Definition 2.1.5 (Clock Ceiling) Given a diagonal-free timed automaton $TA = (L, l_0, \mathbb{X}, inv, E)$, let $k : \mathbb{X} \rightarrow \mathbb{N}$ be a function mapping each clock to a natural number representing its *ceiling* in TA. The ceiling of a clock is the upper bound used as a constraint in TA. Thus for some clock $x \in \mathbb{X} : x < k(x)$ or $x \leq k(x)$ are part of some constraint in TA. □

Definition 2.1.6 (Region Equivalence) Given a diagonal-free timed automaton $TA = (L, l_0, \mathbb{X}, inv, E)$, with $k : \mathbb{X} \to \mathbb{N}$ mapping each clock to its ceiling. For a real number r, let $\text{frac}(r)$ denote the fractional part of r, and $\lfloor r \rfloor$ denote its integer part. Two clock valuations u, v are region equivalent, denoted $u \simeq_k v$, if and only if all of the following conditions hold:

1. for all x, either $\lfloor u(x) \rfloor = \lfloor v(x) \rfloor$ or both $u(x) > k(x)$ and $v(x) > k(x)$,
2. for all x, if $u(x) \leq k(x)$ then $\text{frac}(u(x)) = 0$ iff $\text{frac}(v(x)) = 0$,
3. for all x, y if $u(x) \leq k(x)$ and $u(y) \leq k(y)$ then $\text{frac}(u(x)) \leq \text{frac}(u(y))$ iff $\text{frac}(v(x)) \leq \text{frac}(v(y))$.

□

The $norm_k$ operation can be used to construct the set of equivalent valuations for some arbitrary set of valuations, and ceiling function.

Definition 2.1.7 (k-Normalization) [BY03]. Let $V \subseteq \mathbb{R}_+^{\mathbb{X}}$ be a set of clock valuations, and $k : \mathbb{X} \to \mathbb{N}$ mapping each clock to its ceiling. The k-normalization operation is defined as follows: $norm_k(V) = \{u \mid \forall v \in V . u \simeq_k v\}$. □

Region equivalence defines equivalence classes on the valuation space that are called regions. Regions are *unbounded* if and only if there exists a clock x, such that for all valuations v we have that $v(x) > k(x)$. Bounded regions can be described using Definition 2.1.8. Unbounded regions can be described by using Definition 2.1.8 to define a region that is bounded but has a clock valuation that assigns a clock to a value beyond its ceiling, and then applying the normalization procedure to this region. Note that by using this approach there are infinite possibilities of representing an unbounded region. Although it is not hard to find a canonical representation, we do not need it in this context.

Definition 2.1.8 (Representation for Bounded Regions) A region can be represented as $R = (h, [r_0, \ldots, r_n]) \subseteq (\mathbb{X} \to \mathbb{N}) \times \text{Seq}(2^{\mathbb{X}})$, where $\text{Seq}(S)$ denotes the set of finite sequences of elements of set S; $[r_0 \ldots r_n]$ is a partitioning of $\mathbb{X}$, where $i > 0$ implies that $r_i \neq \emptyset$. Given a clock valuation $v \in \mathbb{R}_+^{\mathbb{X}}$, and $v \in R$ the following conditions hold:

- for all $x, \lfloor v(x) \rfloor = h(x)$,
- $x \in r_0$ iff $\text{frac}(v(x)) = 0$,
- $x, y \in r_i$ iff $\text{frac}(v(x)) = \text{frac}(v(y))$,
- $x \in r_i, y \in r_j$, with $i < j$ iff $\text{frac}\, v(x) < \text{frac}\, v(y)$.

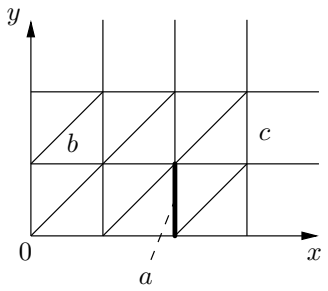
□

Consider a timed automaton with two clocks x and y, with ceilings $k(x) = 3$ and $k(y) = 2$. Then the possible regions of the automata are depicted in figure 2.2, and consist of: all corner points (including those on the axes), all line segments between points, and all spaces defined by the segments. The total number of possible regions is 60. Three regions are marked a, b and c, where a is a line segment. The representations are as follows, with two possible representations

of c:

$$\begin{aligned} a &= (\{x \mapsto 2, y \mapsto 0\}, [\{x\}, \{y\}]) \\ b &= (\{x \mapsto 0, y \mapsto 1\}, [\emptyset, \{y\}, \{x\}]) \\ c &= norm_k(\{x \mapsto 3, y \mapsto 1\}, [\emptyset, \{y, x\}]) \\ \text{or } c &= norm_k(\{x \mapsto 4, y \mapsto 1\}, [\emptyset, \{x\}, \{y\}]) \end{aligned}$$

Figure 2.2: Regions for a system with two clocks

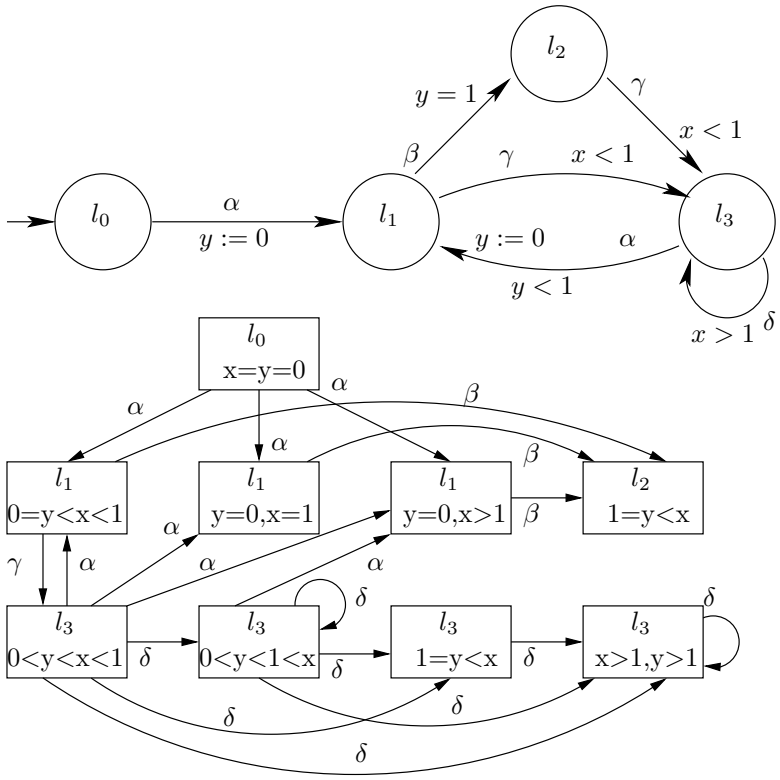


The importance of region equivalence is that it can be lifted to a time-abstracting bisimulation on the states of a timed automaton, by defining that two states are bisimilar only if they have the same location *and* their clock valuations are region equivalent [TY01]. Now by using Definition 2.1.4 it is possible to construct a finite quotient that is finite due to the finiteness of the number of locations. The new system is called the region graph. The states of the region graph are called *symbolic states*, and are described by tuples (l, R), where l is a location, and $R \subseteq \mathbb{R}_+^{\mathbb{X}}$ is a region. More generally symbolic states are sets of states that can be represented as a tuple. The first of the tuple part contains the discrete part that is the same for all states. In this thesis the discrete part will consist of a location, and we write $loc(l, R) = l$. The second part is a set containing all continuous parts of the states. The definition of region equivalence can be altered such that it is an equivalence relation on timed automata including difference constraints. For more information see [BDFP00].

On the region graph a number of properties can be model checked, including reachability. Forward exploration and backward exploration are two methods for checking reachability. The region graph captures all possible behavior of a timed automaton, even of states that are not reachable from the starting state, or states that will never reach a target location. This results in redundant symbolic states. When using forward exploration it is possible to generate only the symbolic states that are reachable from the starting symbolic state. With backward exploration we only have to generate the symbolic states that can reach a target location.

Figure 2.3 gives a timed automaton and its region graph (from [AD94]). The timed automaton has extra labels $\alpha, \beta, \gamma, \delta$ on its edges. And the corresponding edges in the region graph have the same label. Note that all invariants are *true*, allowing all clock valuations in locations.

Figure 2.3: An example timed automaton and its region graph [AD94].



Next to region equivalence, other time-abstracting bisimulations on timed automata are possible. For reachability, more generally than target locations, we can use a target set of states. When a time-abstracting bisimulation is used to calculate reachability, it must be such that its equivalence classes respect the partitioning between target states and non-target states. There must be no equivalence class containing target states as well as non-target states, because then these states will be regarded equivalent. Of interest is the *coarsest bisimulation*, i. e. the time-abstracting bisimulation with the least equivalence classes respecting the partitioning of target states.

Forward reachability analysis

With region equivalence, in the worst case the number of regions is exponential in the number of clocks, and constants $k(\cdot)$ of the ceiling function. In the simple system of figure 2.2 we already had 60 regions. Using forward or backward exploration to generate a subset of the symbolic state space will not help very much.

Forward reachability analysis is another method of constructing a finite abstraction of a timed automaton, by using symbolic states with zones instead of

regions. On the symbolic states a successor operation is defined. A successor symbolic state contains all states reachable by delaying some amount of time, and taking an edge. The successor operation implements forward exploration on a (infinite) set of states that can be described by a zone. Zones are, like regions, sets of clock valuations, in fact they are a convex set of regions. The number of reachable zones is in most cases smaller than the number of regions.

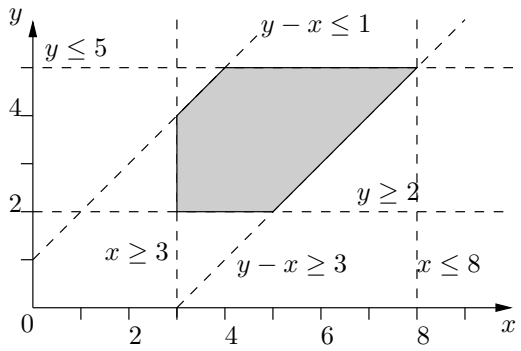
Definition 2.1.9 A **Zone** is the set of all possible clock valuations that respect some clock constraint. A zone takes the form of a convex polyhedron in the space $\mathbb{R}_+^{\mathbb{X}}$. The set of all possible zones is closed under intersection, as clock constraints can be a conjunction of clock constraints.

Different clock constraints can describe the same zone. The constraint of a zone has a canonical form, where the number of equations of the form $(x \bowtie b)$ or $(x - y \bowtie b)$ is maximal, and those equations can't be strengthened without changing the zone it describes. Strengthening an equation means that b is changed such that less clock valuations satisfy the equation, for example equation $x - y \leq 2$ can be strengthened to $x - y \leq 1$. When regarding the constraint of a zone we will always use its canonical form. A clock constraint is said to be part of a zone, if it is part of the canonical constraint that induces the zone. Zones can effectively be represented and stored in memory by using difference bound matrices [Dil89].

Let N be a zone, and $b \in \mathrm{Cons}(\mathbb{X})$, then $N \wedge b$ denotes the zone that is induced by the constraint of N strengthened by constraint b. b can also be seen as a zone on itself, in that way: $N \wedge b = N \cap b$. A zone is called *closed* when it is induced by a closed clock constraint. For a closed zone Z, the *offset clock valuation* $\Delta_Z \in Z$ is the infimum clock valuation on the zone, formally: $\forall v \in Z. \forall x \in \mathbb{X}. \Delta_Z(x) \leq v(x)$. □

Example 2.1.10 Figure 2.4 gives an example of a zone on two clocks x and y. The borders of the zone are formed by the equations of its clock constraint. □

Figure 2.4: An example zone



Like regions, zones can be used to construct symbolic states (l, Z) where l is a location, and $Z \subseteq \mathbb{R}_+^{\mathbb{X}}$ is a zone, with $Z \subseteq \mathit{inv}(l)$. Forward reachability

analysis starts with a symbolic state representing the starting state. Then the successor operation is used to compute symbolic states that are reachable from the starting symbolic state, by letting some amount of time progress, and then taking an edge in the timed automaton. Now for the new symbolic states the successors are computed. This process is repeated until no new symbolic states are generated. Forward analysis only works correctly for diagonal-free timed automata, for timed automata including difference constraints, special measures should be taken [BDFP00].

As the clocks can take arbitrary high values, there is a possibility of generating an infinite number of symbolic states. To ensure termination, zones are normalized using the $norm_k$ operation of Definition 2.1.7. The intuition is that, like for regions, all values for a clock that are above the ceiling for that clock are equivalent. Now zones that have bounds beyond the ceiling can be transformed to equivalent zones by removing these bounds. As the number of locations is finite, in this way only a finite number of symbolic states will be generated using forward analysis. Note that forward analysis does not construct a partition of the state space, because the zones can overlap.

The next definition is a formal definition of the successor operations on sets of states in a timed transition system. Note that this can be any timed transition system, it is not necessarily the semantics of some timed automaton. As the number of states in a timed transition system can be infinite, computation cannot be performed analogously to the definition.

Definition 2.1.11 (General Successor Operations) Let $TTS = (S, Act, \rightarrow)$ be a timed transition system, and subset $S' \subseteq S$. Define the following operations:

time successor: $tsucc(S') = \{t \mid s \xrightarrow{d} t \text{ for some } s \in S' \text{ and } d \geq 0\}$,

discrete successor: $dsucc_a(S') = \{t \mid s \xrightarrow{a} t \text{ for some } s \in S'\}$, for $a \in Act$,

successor: $succ_a = tsucc \circ dsucc_a$.

□

For timed automata the semantics are defined by a timed transition system with an infinite number of states. Using symbolic states with zones, it is possible to define successor operations for timed automata that are computable. Figure 2.5 gives an example of the operations $^{\uparrow}$ and $[r := 0]$.

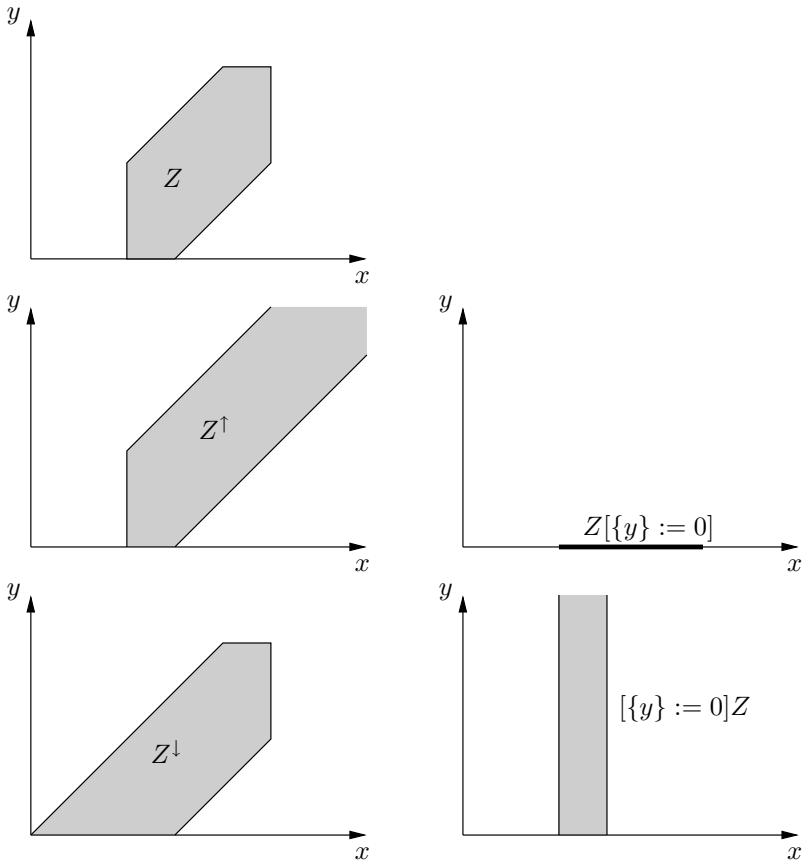
Definition 2.1.12 (Timed Automata Successor Operations) [BY03]. Let $TA = (L, l_0, \mathbb{X}, inv, E)$ be a timed automaton. Let Z be a zone and r a set of clocks. We define $Z^{\uparrow} = \{v + d \mid v \in Z \wedge d \in \mathbb{R}_+\}$ and $Z[r := 0] = \{v[r := 0] \mid v \in Z\}$. Redefine the following operations:

time successor: $tsucc(l, Z) = (l, norm_k(Z^{\uparrow} \cap inv(l)))$,

discrete successor: $dsucc_{(l,g,r,l')}(l, Z) = (l', (Z \cap g)[r := 0]) \cap inv(l')$, if $l \xrightarrow{g,r} l' \in E$.

□

Figure 2.5: Example of operations on a zone Z.



Backward reachability analysis

Like forward reachability, backward reachability uses zones instead of regions as finite abstraction. Backward reachability starts with a set of symbolic states representing the target locations. The predecessor operation is used to compute symbolic states that can reach target symbolic states, by taking an edge in the timed automaton, and then letting some amount of time progress. In this way new symbolic states are generated from previous generated ones, until no new symbolic states can be generated. For termination, in contrast to forward reachability analysis, the ceiling of the automaton does not need to be taken into account [HNSY92].

Like Definition 2.1.11 did for successor operations, we now give a general definition for predecessor operations.

Definition 2.1.13 (General Predecessor Operations) [HNSY92]. Let $TTS = (S, Act, \rightarrow)$ be a timed transition system, and subset $S' \subseteq S$. Define the following operations:

time predecessor: $tpre(S') = \{s \mid s \xrightarrow{d} t \text{ for some } t \in S' \text{ and } d \geq 0\}$,

discrete predecessor: $dpre_a(S') = \{s \mid s \xrightarrow{a} t \text{ for some } t \in S'\}$, for $a \in Act$,

predecessor: $pre_a = tpre \circ dpre_a$.

□

Like Definition 2.1.12 did for successor operations, we now give a computable definition of predecessor operations, that work on zones. The definition uses operation $^{\downarrow}$, called the *past-operator*, and $[r := 0]\cdot$. Figure 2.5 gives an example of both operations.

Definition 2.1.14 (Timed Automata Predecessor Operations) [HNSY92, KNS03]. Let $TA = (L, l_0, \mathbb{X}, inv, E)$ be a timed automaton. Let Z be a zone and r a set of clocks. We define $Z^{\downarrow} = \{v \mid v + d \in Z \text{ for some } d \in \mathbb{R}_+\}$ and $[r := 0]Z = \{v \mid v[r := 0] \in Z\}$. Redefine the following operations:

time predecessor: $tpre(l, Z) = (l, Z^{\downarrow} \cap inv(l))$,

discrete predecessor: $dpre_{(l,g,r,l')}(l', Z) = (l, ([r := 0]Z) \cap g \cap inv(l))$, if $l \xrightarrow{g,r} l' \in E$.

□

Lemma 2.1.15 Given TTS and set of clock valuations V that are all possible in TTS then

- if $v \in V^{\downarrow}$ and $\exists d \geq 0. v + d \in V$ we have that $v + d' \in V$ for all $0 \leq d' < d$.
- $V \subseteq V^{\downarrow}$

Proof: by definition of $^{\downarrow}$ and Wang's Axiom in definition of TTS. □

2.2 Linear Priced Timed Automata

Linear priced timed automata (LPTA) are timed automata extended with a variable that can be used to model the cost of execution. This model has been independently introduced in [BFH$^+$01] and [ATP01] (uses the name Weighted Timed Automata). The cost variable is incremented due to prices on certain behavior. Edges have a price for taking the edge. Locations have a price that specifies a cost per time-unit for time spent in the location. To make things clear first an example is given, and then the formal definition and its semantics.

Example 2.2.1 Consider the simple static scheduling problem represented by the LPTA in figure 2.6 (based on [BFH$^+$01]), which contains 3 tasks $\{A, B, C\}$. D denotes the 'goal'. Task A is the first task to be executed. At beginning of A all clocks have value zero, and the cost variable is also zero. The edge from

A to B makes that task B will commence execution only when task A finishes. It is clear that task B is the only task that can be executed multiple times. The price of the tasks are written in the corresponding locations. They model the cost per time unit of executing the task. Prices on edges are written next to the corresponding arrow. The cost variable can model for example power consumption, or cost on some other resource. None of the locations has an invariant, meaning there is no restriction on the time that is spent in a location.

A possible path in the underlying timed transition system leading to location D is:

$$\begin{aligned}
&(A,(0,0),0) \xrightarrow{1} (A,(1,1),2)\\
&\rightarrow (B,(0,0),3) \xrightarrow{0.3} (B,(0.3,0.3),3.3)\\
&\rightarrow (C,(0.3,0.3),3.3) \xrightarrow{2.7} (C,(3,3),11.4)\\
&\rightarrow (D,(3,3),15.4)
\end{aligned}$$

Note that states are now triples (l,v,c) where c is the cumulated cost so far. The given path is not optimal when minimizing cost. The next path is better, because more time is spent on task B than on C, and B is 'cheaper' than C.

$$\begin{aligned}
&(A,(0,0),0) \xrightarrow{1} (A,(1,1),2)\\
&\rightarrow (B,(0,0),3) \xrightarrow{1} (B,(1,1),4)\\
&\rightarrow (C,(1,1),4) \xrightarrow{2} (C,(3,3),10)\\
&\rightarrow (D,(3,3),14)
\end{aligned}$$

An optimal path leading to 'goal' D, is obtained by executing task B two times:

$$\begin{aligned}
&(A,(0,0),0) \xrightarrow{1} (A,(1,1),2)\\
&\rightarrow (B,(0,0),3) \xrightarrow{3} (B,(3,3),6)\\
&\rightarrow (B,(0,3),8)\\
&\rightarrow (C,(0,3),8)\\
&\rightarrow (D,(0,3),12)
\end{aligned}$$

□

Definition 2.2.2 A **Linear Priced Timed Automaton** is a tuple $(L,l_0,\mathbb{X},inv,E,c,\dot{\$},\$)$ with the following properties:

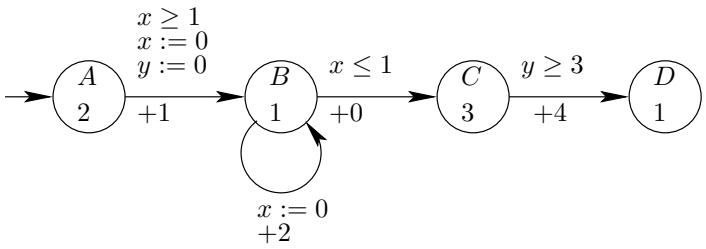
$(L,l_0,\mathbb{X},inv,E)$ - is a diagonal-free timed automaton following Definition 2.1.1.

c - the cost variable.

$\dot{\$} : L \rightarrow \mathbb{N}$ - function assigning a price to each location.

$\$: E \rightarrow \mathbb{N}$ - function assigning prices to transitions. □

Figure 2.6: A LPTA of a scheduling problem with prices



The semantics are easy to understand. The cost variable c is only used as an observable, meaning that it is not used in guards or invariants. If we only regard the value of the clocks the execution is exactly the same as that of the underlying timed automaton. Initially the value of c is zero. When taking an edge the value increments by the price on that edge. When staying for t time units in some location l, c increments by $\dot{\$}(l) \cdot t$.

Definition 2.2.3 (LPTA Semantics) The semantics of a linear priced timed automaton $LPTA = (L, l_0, \mathbb{X}, inv, E, c, \dot{\$}, \$)$ is given by the timed transition system $TTS = (S, Act, \rightarrow)$, where $Act = E, S = \{(l, v, c) \mid l \in L \wedge v \in inv(l) \wedge c \in \mathbb{R}_+\}$. Transitions are defined by letting time pass or taking an edge:

timed transitions: $(l, v, c) \xrightarrow{d} (l, v+d, c+d \cdot \dot{\$}(l))$ if $v \in inv(l)$ and $(v+d) \in inv(l)$,

discrete transitions: $(l, v, c) \xrightarrow{a} (l', v', c + \$(l \xrightarrow{g,r} l'))$ if $a = (l \xrightarrow{g,r} l') \in E, v \in g, v' = v[r := 0]$ and $v' \in inv(l')$.

□

2.2.1 Verification of LPTA

From Definition 2.2.2, we see that a LPTA is in fact a normal timed automaton extended with prices. Reachability analysis is possible on LPTA through reachability analysis of the underlying timed automaton. This only answers the question if a certain set of locations is reachable. It doesn't minimize cost. As we have seen in example 2.2.1, all paths would lead to 'goal' D, but only the last path gives the minimal cost.

In contrast to timed automata, in general, LPTA do not have a finite bisimulation [Hen95]. Despite this fact, *minimum cost reachability* is proven to be decidable by [BFH+01] and [ATP01]. [BFH+01] shows this result for bounded LPTA. A LPTA is bounded if the clocks can't get arbitrary high, thus all regions will be bounded. This limitation is no problem, as they show that every LPTA can be transformed to a bounded LPTA that is equivalent. It is equivalent in the sense that every location is reachable with some path in both LPTA, that has exactly the same cost. The bounded LPTA has constant updates on clocks. Edges may now reset a clock to some natural number. It is easy to alter the

reset operation of Definition 2.2.7 to incorporate this new functionality; just let h' in the definition have the new constant value for the clock that was updated. The recursive definition for reseting a set of clocks holds for a set of updates.

Minimum cost reachability answers the question whether a set of locations is reachable from the starting state, and if it is reachable, by what minimum cost. Being able to compute minimal cost reachability, it is also possible to compute *cost-bounded reachability*. Cost-Bounded reachability answers the question whether a set of locations is reachable with an upper bound on the cost.

Priced regions

In [BFH+01] decidability of minimum cost reachability is proven by using priced regions. Priced regions are regions, where cost is associated with the clock valuations in the region.

Definition 2.2.4 [BFH+01]. A **(Bounded) Priced Region** $R = (h, [r_0 \dots r_n], [c_0 \dots c_n])$ is an element of $(\mathbb{X} \to \mathbb{N}) \times \text{Seq}(2^{\mathbb{X}}) \times \text{Seq}(\mathbb{N})$, where $\text{Seq}(S)$ denotes the set of sequences of elements of set S; The first two components: h and $[r_0, \dots, r_n]$, make a bounded region. □

The closure of a bounded region consists of all valuations in the bounded region, and the valuations that are not part of the region, but which values can be approached arbitrary close. The closure of the bounded region is a convex polyhedron. For a priced region $R = (h, [r_0 \dots r_n], [c_0 \dots c_n])$, the naturals $c_0 \dots c_n$ are associated with the vertices of the bounds of the closure of $(h, r_0, \dots, r_n)$. c_0 is associated with the vertex that is the infimum for the value of all clocks. In case of two dimensions, this will be the left-most lower vertex. By adding one time unit to all the clocks of this valuation that appear in r_n we get the next vertex, to which we assign cost c_1. Now this process is repeated by adding one time unit to all clocks appearing in r_{n-1} etcetera. The costs span a linear cost plane on the n-dimensional unpriced region. The cost of clock valuations in the priced region are given by the next definition. Figure 2.7 gives an example, for priced region $R = (h, [r_0 \dots r_n], [c_0 \dots c_n])$, where $h(x) = 1, h(y) = 0, r_0 = \emptyset, r_1 = \{y\}, r_2 = \{x\}, c_0 = 1, c_1 = 4, c_2 = 3$.

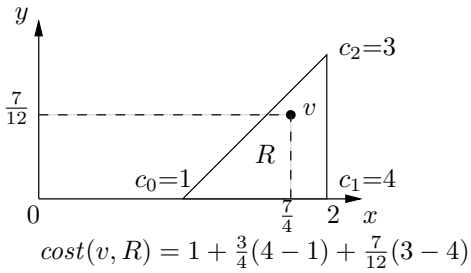
Definition 2.2.5 (Cost inside Priced Regions) [BFH+01]. Given priced region $R = (h, [r_0 \dots r_n], [c_0 \dots c_n])$ and clock valuation $v \in \mathbb{R}_+^{\mathbb{X}}$, the cost of v in R is defined as:

$$cost(v, R) = c_0 + \sum_{i=0}^{n-1} \text{frac}(v(x_{n-i})) \cdot (c_{i+1} - c_i)$$

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

The previous definition can be used to calculate the cost of clock valuations inside the priced region. The definition differs from [BFH+01], such that it is also possible to calculate the cost of valuations associated with the region that lie outside the priced region. The cost function is extended beyond the boundaries of the region. This useful property is needed for calculating the cost

Figure 2.7: Example cost inside a priced region.



of valuations in unbounded priced regions.

Next we recall the operations on priced regions from [BFH+01], which later will be used to define the symbolic semantics for LPTA. The delay operation computes a time successor for a priced region. The unpriced region of the successor is the same as the unpriced region we would get by taking a time-abstracting ϵ-transition in the region graph. The two cases in the operation are illustrated in Figure 2.8.

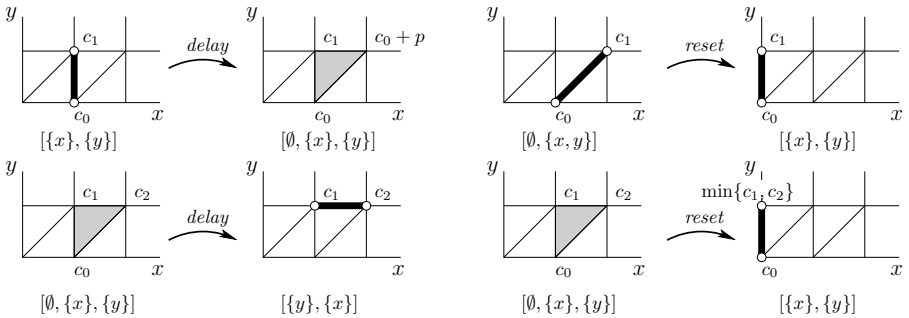
Definition 2.2.6 (Priced Region Delay) [BFH+01]. Given a priced region $R = (h, [r_0 \dots r_n], [c_0 \dots c_n])$ and a price $q \in \mathbb{N}$, the function *delay* is defined as follows:

- if $r_0 \neq \emptyset$ then $delay(R, q) = (h, [\emptyset, r_0, \dots, r_n], [c_0, \dots, c_n, c_0 + q])$,
- if $r_0 = \emptyset$ then $delay(R, q) = (h', [r_n, r_1, \dots, r_{n-1}], [c_1, \dots, c_n])$, where

$$h'(x) = \begin{cases} h(x) + 1 & \text{if } x \in r_n \\ h(x) & \text{otherwise.} \end{cases}$$

□

Figure 2.8: Example operations on priced regions [BFH+01].



When resetting a clock, a region may lose a dimension. Two vertices collapse

to one new vertex. In a priced region a choice must be made for the cost on the new vertex. We chose the minimum of the cost on the two vertices, as we are interested in minimum cost reachability. Two of the three cases in the operation are illustrated in Figure 2.8.

Definition 2.2.7 (Priced Region Reset) [BFH+01]. Given a priced region $R = (h, [r_0 \dots r_n], [c_0 \dots c_n])$ and a clock $x \in r_i$, the function *reset* is defined as follows:

- if $i = 0$ then $reset(x, R) = (h[\{x\} := 0], [r_0, \dots, r_n], [c_0, \dots, c_n])$,
- if $i > 0$ and $r_i \neq \{x\}$, then $reset(x, R) = (h[\{x\} := 0], [r_0 \cup \{x\}, \dots, r_i \backslash \{x\}, \dots, r_n], [c_0, \dots, c_n])$,
- if $i > 0$ and $r_i = \{x\}$, then $reset(x, R) = (h[\{x\} := 0], [r_0 \cup \{x\}, \dots, r_{i-1}, r_{i+1}, \dots, r_n], [c_0, \dots, c_{n-i-1}, c', c_{n-i+2}, \dots, c_n])$, where $c' = \min(c_{n-i}, c_{n-i+1})$.

The reset operation on a set r of clocks is defined inductively by: $reset(r \cup \{x\}, R) = reset(r, reset(x, R))$, and $reset(\emptyset, R) = R$. □

The increment operation is used to model a discrete cost increment on an edge.

Definition 2.2.8 (Priced Region Increment) [BFH+01]. Given a priced region $R = (h, [r_0 \dots r_n], [c_0 \dots c_n])$ and a price $q \in \mathbb{N}$, the increment of R with respect to q is the priced region $inc(R, q) = (h, [r_0, \dots, r_n], [c'_0, \dots, c'_n])$ where $c'_i = c_i + q$. □

If in priced region R, no clock has fractional part 0, then time may pass in R. When R is associated with location l, the costs on the vertices are determined by $\dot{\$}(l)$. By taking an edge, a new location l' may become associated with R. All time spent in l, can also be spent in l'. The next operation, is used to calculate the cheapest possible way of letting time elapse.

Definition 2.2.9 (Priced Region Self) [BFH+01]. Given a priced region $R = (h, [r_0 \dots r_n], [c_0 \dots c_n])$ and a price $q \in \mathbb{N}$, the function *self* is defined as follows:

- if $r_0 \neq \emptyset$ then $self(R, q) = R$,
- if $r_0 = \emptyset$ then $self(R, q) = (h, [r_0, \dots, r_n], [c_0, \dots, c_{n-1}, c'])$, where $c' = \min(c_n, c_0 + q)$.

□

Symbolic semantics

With definitions 2.2.6 2.2.7, 2.2.8, and 2.2.9, it is possible to define a symbolic transition system for a LPTA. The states of the transition system consist of a location, and a priced region. The big difference with the symbolic states for timed automata is that now symbolic states are *not* a set of states of some LPTA. A symbolic state only captures states without cost of a LPTA, i. e. a location and possible clock valuations, and defines a minimal cost function on

those states. A Symbolic state is regarded 'cheaper' than some other symbolic state, if all possible clock valuations have a lower cost. The state system captures all cost minimal paths; symbolic states do *not* model a set of states of the LPTA that are bisimilar.

Definition 2.2.10 [BFH+01]. The **Symbolic semantics of a** $LPTA = (L, l_0, \mathbb{X}, inv, E, c, \dot{\$}, \$)$ is defined as a labeled transition system where symbolic states are of the form (l, R), with l a location and R a priced region satisfying $inv(l)$. The starting symbolic state is $(l_0, (\bar{0}, [\mathbb{X}], [0]))$. The transition relation is as follows:

- $(l, R) \xrightarrow{\epsilon} (l, delay(R, \dot{\$}(l)))$ if $delay(R, \dot{\$}(l)) \in inv(l)$,
- $(l, R) \xrightarrow{(l,g,r,l')} (l', R')$ if there exists g, r such that $(l \xrightarrow{g,r} l') \in E, R \subseteq g, R' = self(inc(reset(r, R), \$(l, g, r, l')), \dot{\$}(l'))$.

□

The definition of the symbolic semantics is slightly different from [BFH+01], where there were three sorts of transitions including a 'self'-transition. In Definition 2.2.10 the 'self'-transition is not possible, but is encapsulated in first type of transitions. This generates less symbolic states, because if *self* generates a 'cheaper' symbolic state we do not need the original one; we are only interested in the minimal cost reachability. This approach is similar to that of the *reset* operation, where the minimum cost is taken on two vertices that collapse.

The priced regions represent the minimum cost for paths in the semantics leading to clock valuations in the region, when these paths take the same edges of the LPTA used in generating the priced region from the starting priced region. $cost(v, R)$ does not give the exact minimum cost of paths to v, but rather gives the infimum cost. Meaning that exact minimum cost on paths respecting the edges by which R was generated, may have a higher cost than $cost(v, R)$, but are arbitrary close to it.

The following lemma is sufficient to show that cost-bounded reachability is decidable on basis of the symbolic semantics of a LPTA.

Lemma 2.2.11 [BFH+01] For every reachable state (l, v, c) in the LPTA, there is a reachable symbolic state (l, R) such that $v \in R$ and $cost(v, R) \leq c$.
Proof: lemma 1 in [BFH+01], the integration of the 'self'-transition into the time-transitions does not alter the proof. □

Priced zones

The definition of zones in timed automata can also be extended with cost. Priced zones are closed zones extended with a linear cost function [LBB+01]. Like zones for timed automata, priced zones can be used to verify reachability of LPTA, but now including costs. In [LBB+01] priced zones are used in a forward exploration algorithm to compute minimum cost reachability. [LBB+01] only defines closed priced zones. Closed priced zones have the nice property that clock valuations with the infimum and supremum for the cost are element of

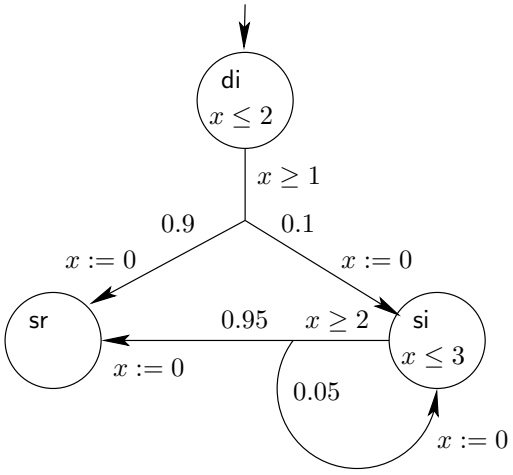
the priced zone. What modifications are necessary when non-closed LPTA and non-closed priced zones are used remains to be investigated. The successor operation on symbolic states with priced zones differs from that using normal zones, in that it generates a set of successor symbolic states. The successor symbolic states have disjoint priced zones.

2.3 Probabilistic Timed Automata

The model of Probabilistic timed automata (PTA) is introduced in [KNSS02]. PTA are timed automata extended with probabilistic edges. Like a normal edge, a probabilistic edge has a source location and a guard. Now the combination of reset and target location is chosen probabilistically from a set, with a discrete probability distribution defined on it.

Example 2.3.1 [KNS03] Consider the PTA modeling a simple probabilistic communication protocol given in Figure 2.9. The locations are named: di (sender has data, receiver is idle); si (sender sends data, receiver idle); and sr (sender has sent data, receiver has received). The automaton starts in location di in which the sender has data that needs to be sent. After between 1 and 2 time units, the protocol makes a transition either to sr with probability 0.9 (data received), or to si with probability 0.1 (data lost). In si after 2 to 3 time units, the protocol will attempt to resend the data, which again can be lost, this time with probability 0.05. □

Figure 2.9: A PTA modeling a probabilistic protocol.



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

Definition 2.3.2 A **Probabilistic Timed Automaton** is a tuple $(L, l_0, \mathbb{X}, inv, pE)$. $L, l_0, \mathbb{X}$, and inv, have the same meaning a in a normal timed automaton (Definition 2.1.1). $pE \subseteq L \times \text{Cons}(\mathbb{X}) \times \text{Dist}(2^{\mathbb{X}} \times L)$ is a probabilistic edge relation. All guards and invariants in PTA are diagonal free. E_{PTA} are the edges of the PTA. $(l, g, p, r, l') \in E_{PTA}$ if $(l, g, p) \in pE$, and $p(r, l') > 0$. □

2.3.1 Probabilistic Systems

A probabilistic system is exactly the same as a *discrete time Markov decision process*. Probabilistic systems are of great importance to the verification of PTA, as both the semantics of a PTA, as well as the symbolic semantics can be described as a probabilistic system.

Definition 2.3.3 [KNSW04]. A **Probabilistic System** is a Markov decision process $(S, Steps)$, where S is a set of states, $Steps \subseteq S \times \text{Dist}(S)$ is a probabilistic transition relation. □

For $s, s' \in S$, $s \xrightarrow{\mu} s'$ denotes a transition. The transition is made by first nondeterministically selecting a distribution $\mu \in \text{Dist}(S)$ such that $(s, \mu) \in Steps$, and then making a probabilistic choice of target state s' according to μ (given that $\mu(s') > 0$). We see that a transition resolves both nondeterminism and probability.

Now a *path* is denoted $\omega = s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} s_2 \xrightarrow{\mu_2} \ldots$. The probability on a finite path $\omega = s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_{n-1}} s_n$ is of course $P(\omega) = \mu_0(s_1) \cdot \mu_1(s_2) \cdot \ldots \cdot \mu_{n-1}(s_n)$.

An *adversary* of a probabilistic system is similar to a scheduler or policy in a Markov decision process. An adversary A is a function mapping every finite path ω in probabilistic system $PS = (S, Steps)$ to a distribution $\mu \in Dist(S)$ such that $\mu \in Steps(last(\omega))$. Hereby all nondeterminism is resolved, therefore an adversary applied to a probabilistic system in fact gives a discrete time Markov chain (DTMC). For any adversary A, let $Path^A_{\mathsf{full}}$ and $Path^A_{\mathsf{fin}}$ respectively denote the infinite paths and finite paths induced by the adversary. $Prob^A$ denotes the probability measure on infinite paths associated with adversary A, defined using classical techniques [KNS03, KSK76]. Adv_{PS} denotes all possible adversaries on PS.

Probabilistic reachability is the probability of reaching a certain set of target states in a finite number of transitions under some adversary. For probabilistic system $(S, Steps)$ with $s \in S$ and $F \subseteq S$, and adversary A let:

$$ProbReach^A(s, F) \stackrel{\mathsf{def}}{=} Prob^A\{\omega \in Path^A_{\mathsf{full}} \mid \omega(0) = s \wedge \exists i \in \mathbb{N}.\omega(i) \in F\}$$

, denote the probabilistic reachability defined on infinite paths as in [KNS03].

The probabilistic reachability depends on the non-deterministic choices made by the adversary. A non-deterministic choice can model branches in system execution, for which the probability distribution is not known, therefore we are interested in *maximal probabilistic reachability*, that gives the maximum if all choices are optimal. The maximal probabilistic reachability for probabilistic

system $(S, Steps)$ with $s \in S$ and $F \subseteq S$ is defined as:

$$MaxProbReach(s, F) \stackrel{\text{def}}{=} \sup_{A \in Adv_{PS}} ProbReach^A(s, F)$$

For finite state probabilistic systems (Markov decision processes), maximal and *minimal* probabilistic reachability are computable [dA99].

2.3.2 Timed Probabilistic Systems

Definition 2.3.4 [KNSW04]. A **Timed Probabilistic System (TPS)** is a probabilistic system of Definition 2.3.3 extended with two extra labels on the probabilistic edge relation. Now $Steps \subseteq S \times \mathbb{R}_+ \times \{\mathsf{time}, \mathsf{disc}, \mathsf{full}\} \times \mathrm{Dist}(S)$, such that if $(s, d, \iota, \mu) \in Steps$ then d denotes the *duration*. time and disc are used to mark *discrete*, and *time* transitions respectively, with the following rules:

non-probabilistic time: if $\iota = \mathsf{time}$ then μ is required to be a point distribution.

discrete transitions: if $\iota = \mathsf{disc}$ then $d = 0$,

time determinism: if $s \xrightarrow[\mathsf{time}]{d,\cdot} t$ and $s \xrightarrow[\mathsf{time}]{d,\cdot} t'$ then $t = t'$,

Wang's Axiom: $s \xrightarrow[\mathsf{time}]{d,\cdot} t$, with $d > 0$, if and only if there exists s' and $d' < d$ such that $s \xrightarrow[\mathsf{time}]{d',\cdot} s'$ and $s' \xrightarrow[\mathsf{time}]{d-d',\cdot} t$.

zero delay: $s \xrightarrow[\mathsf{time}]{0,\cdot} t$ if and only if $s = t$.

full denotes *full* transitions. Full transitions are a combination of a time transition and a discrete transition. Formally, there exist transitions $s \xrightarrow[\mathsf{time}]{d,\{t \mapsto 1\}} t$ and $t \xrightarrow[\mathsf{disc}]{0,\mu} u$ if and only if transition $s \xrightarrow[\mathsf{full}]{d,\mu'} u$ exists, where $\mu'(u) = \mu(u)$. □

Labels on transitions are sometimes left out when they are clear from the context. Now adversaries map finite paths to duration-distribution pairs. A path ω in a TPS has the form: $\omega = s_0 \xrightarrow[\iota_0]{d_0,\mu_0} s_1 \xrightarrow[\iota_1]{d_1,\mu_1} s_2 \xrightarrow[\iota_2]{d_2,\mu_2} \cdots$. The *duration* $\mathcal{D}$ of ω up to the $n+1$th state is $\mathcal{D}_\omega(n+1) = \sum_{i=0}^{n} t_i$. Path ω is *divergent* if for any $t \in \mathbb{R}_+$, there exists $j \in \mathbb{N}$ such that $\mathcal{D}_\omega(j) > t$.

Definition 2.3.5 (Divergent Adversary) [KNSS02]. An adversary A of a timed probabilistic system *TPS* is *divergent* if and only if $Prob^A\{\omega \in Path^A_{\mathsf{full}} \mid \omega \text{ is divergent} = 1\}$. □

The definition of divergent adversaries considers *probabilistic divergence*, as some paths may be non-divergent, but the probability on these paths is zero. This in contrast to a stronger definition that requires *all* paths of an adversary to be divergent. See [KNSS02] for the reason why probabilistic divergence is used.

Definition 2.3.6 A timed probabilistic system is **non-Zeno** if and only if there exists a divergent adversary. □

Definition 2.3.7 (PTA Semantics) The semantics of a probabilistic timed automaton $PTA = (L, l_0, \mathbb{X}, inv, pE)$ are given by a timed probabilistic system $TPS = (S, Steps)$, where $S = \{(l, v) \mid l \in L \wedge v \in inv(l)\}$. Now $((l, v), d, \iota, \mu) \in Steps$, if and only if one of the following conditions holds:

time transitions: $\mu(l, v + d) = 1$ (point distribution), $v + d \in inv(l)$, and $\iota = \mathsf{time}$,

discrete transitions: $d = 0$ and there exists $(l, g, p) \in pE$ such that $v \in g$, $\iota = \mathsf{disc}$, and for any $(l', v') \in S$: $\mu(l', v') = \sum_{r \subseteq \mathbb{X} \wedge v' = v[r:=0]} p(r, l')$.

□

A PTA is considered *non-Zeno* if the underlying TPS is *non-Zeno*. In verification of PTA only divergent adversaries are of interest as these model realizable behavior. A Zeno PTA contains no such adversaries and will typically contain a modeling error, as there is no possibility of letting time diverge. In real systems time will always proceed, it does not converge to some value.

2.3.3 Verification of PTA

All reachability results for timed automata also hold for PTA. This can simply be verified by the following observation: every probabilistic edge can be replaced with a set of normal edges, where every normal edges has one possibility of reset and target location from the distribution of the probabilistic edge. The following definition gives the construction of a timed automaton from some PTA.

Definition 2.3.8 For some $PTA = (L, l_0, \mathbb{X}, inv, pE)$ let PTA_{TA} denote the timed automaton $PTA_{TA} = (L, l_0, \mathbb{X}, inv, E)$, where $E = E_{PTA}$. □

Of course in addition, for PTA, we want to say something about the probability that certain behavior occurs. In [KNSS02] it is shown that by using region equivalence on PTA it is possible to construct a region graph with probabilistic edges. The probabilistic region graph is an abstraction that is sufficient for model checking the logic PTCTL, and therefore can also be used for probabilistic reachability of a set of locations. Section 4.4.3 of chapter 4 gives the formal definition.

In [KNSS02], also forward analysis on PTA using zones is discussed. Assume some PTA, on PTA_{TA} we can generate a (symbolic) state space Σ of zones with forward analysis of normal timed automata. Now construct a probabilistic system $PS_{fwd} = (\Sigma, Steps)$ on the state space, where $\mu \in Steps(l, Z)$ if and only if there exists g and p such that for all $(l', Z') \in \Sigma$:

$$\mu(l', Z') = \sum_{\substack{r \subseteq \mathbb{X} \\ \emptyset \neq (l', Z') = succ_{(l,g,p,r,l')}(l, Z)}} p(r, l')$$

With the probabilistic system PS_{fwd} only an upper bound on the maximum

reachability probability can be computed, see [KNSS02].

In [KNS01] backward analysis is used to compute exact maximal probabilistic reachability on symbolic probabilistic systems. Symbolic probabilistic systems are a very general class of probabilistic systems that have a symbolic representation. PTA are a possible instance of such a symbolic probabilistic system. In [KNS03] the approach of [KNS01] is specially tailored toward PTA. More important the problem of minimal probabilistic reachability is solved. The method requires the PTA to be non-Zeno. Due to a minimum and a maximum we can compute a probability interval for reachability using backward analysis, and it is possible to model check full PTCTL.

The backward analysis of [KNS01] and [KNS03] is based on the same predecessor operation as for normal timed automata. In addition, the intersection between certain symbolic states is added to the symbolic state space. Now also on these interactions predecessor operations are applied. In short, the intersections represent set of states from which different paths can lead to the target under the same adversary. The intersections are of interest, as more paths to the target will lead to a higher reachability probability. The problem could also be solved by using the coarsest bisimulation [Spr00]. For the quotient system under coarsest bisimulation, the same rules as for the probabilistic region graph would hold. The problem is that a lot of symbolic states are generated, because intersections, *and* differences between symbolic states are added. The method of [KNS01] and [KNS03] only adds intersections, and is capable of computing maximum and minimum probabilistic reachability.

Chapter 3

Weighted Probabilistic Timed Automata

Weighted probabilistic timed automata (WPTA) are probabilistic timed automata extended with prices in the locations, like linear priced timed automata. WPTA are as far as we know a new formalism, but WPTA can be seen as a specialization of probabilistic linear hybrid automata that are presented in [Spr00].

In section 3.1 we define WPTA formally, we give an example, and define the formal semantics as timed probabilistic system (TPS). Section 3.2 gives the formal definition of the cost-bounded probabilistic reachability problem. In section 3.3 known reachability results for WPTA are discussed. Section 3.4 presents our new symbolic states for WPTA based on priced zones, and a computable predecessor operation. Section 3.5 presents an algorithm for computing cost-bounded reachability using our symbolic states and predecessor operations. Finally section 3.6 argues why our symbolic states are not suitable, and presents symbolic states based on a new abstraction called *multi-priced zones*.

3.1 The Model

Definition 3.1.1 A **Weighted Probabilistic Timed Automaton** is a tuple $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$, where each element has the same meaning as in LPTA and PTA, and are defined as follows:

- L - finite set of locations.
- $l_0 \in L$ - single starting location.
- $\mathbb{X}$ - finite set of clocks.
- $inv : L \to \text{Cons}(\mathbb{X})$ - function assigning to each location an invariant condition.
- $pE \subseteq L \times \text{Cons}(\mathbb{X}) \times Dist(2^{\mathbb{X}} \times L)$ - probabilistic edge relation.
- c - the cost variable.
- $\dot{\$} : L \to \mathbb{N}$ - function assigning a price to each location.

Note that WPTA in contrast to LPTA do not have prices on edges. All guards and invariants in WPTA are diagonal-free. We define E_{WPTA}, the set of edges of the WPTA, as follows: $(l, g, p, r, l') \in E_{WPTA}$ if $(l, g, p) \in pE$ and $p(r, l') > 0$. Note that the tuple $(L, l_0, \mathbb{X}, inv, pE)$ in fact defines a PTA (Definition 2.3.2), and $\dot{\$}$ defines prices in this PTA. □

Figure 3.1 gives an example of a WPTA. It has five locations:
$L = \{l_0, l_1, l_2, \text{goal}, \text{fail}\}$. The set of clocks is $\mathbb{X} = \{x, y\}$. Locations are drawn as circles with their name and price inscribed, e. g. $\dot{\$}(l_1) = 2$. The invariant of each location is *true*, meaning there are no constraints on the values of clocks in a location. Probabilistic edges are drawn as arrows (possibly with multiple heads), and we use α, β, γ to point out which of the distributions is chosen from location l_0.

Like for PTA, the semantics of a WPTA can be defined by paths in a TPS, but now the states include the value for the cost variable. A WPTA is called *non-Zeno* if the TPS semantics is non-Zeno.

Definition 3.1.2 (WPTA Semantics) Let $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$ be a weighted probabilistic timed automaton. We define its semantics by the timed probabilistic system $TPS = (S, Steps)$, where $S = \{(l, v, c) \mid l \in L \wedge v \in inv(l) \wedge c \in \mathbb{R}_+\}$. A probabilistic transition $((l, v, c), d, \iota, \mu)$ is element of *Steps*, if and only if $\forall (l', v', c') \in \text{support}(\mu).c' = c + \$(l) \cdot d$, and one of the following conditions holds:

time transitions: $d \geq 0$, $\iota = \mathsf{time}$, $\mu(l, v + d, c') = 1$ (point distribution) and $v + d' \in inv(l)$ for all $0 \leq d' \leq d$.

discrete transitions: $d = 0$, $\iota = \mathsf{disc}$, and there exists $(l, g, p) \in pE$ such that $v \in g$, and for any $(l', v', c') \in S, \mu(l', v', c') = \sum_{r \subseteq \mathbb{X} \wedge v' = v[r:=0]} p(r, l')$.

□

We continue our example of figure 3.1. States are of the form $(l, (x, y), c)$. We

are interested in reaching location 'goal' from the starting state. When prices and probabilities are ignored we have a simple path:

$$(l_0,(0,0),0) \xrightarrow[\text{disc}]{\alpha} (l_2,(0,0),0) \xrightarrow[\text{time}]{6,\cdot} (l_2,(6,6),12) \xrightarrow[\text{disc}]{} (\text{goal},(6,6),12)$$

The probability of this path is as high as possible, namely 1. Note that when more than one time unit is spent in l_0, x should be reset by taking β or γ. Assume we are interested in reaching location 'goal' with cost at most 10. Now the cost of the above path, which is 12, is too high. The path with minimal cost is:

$$(l_0,(0,0),0) \xrightarrow[\text{time}]{6,\cdot} (l_0,(6,6),6) \xrightarrow[\text{disc}]{\gamma} (l_0,(0,6),6) \xrightarrow[\text{disc}]{\alpha} (l_2,(0,6),6) \xrightarrow[\text{disc}]{} (\text{goal},(0,6),6)$$

The probability on this path is 0.4. But is this probability maximal? The answer is no. In the next section we will see why the maximum probability is of interest. Take cost as well as probability into account. An adversary chooses in a location between taking a probabilistic edge or letting some amount of time pass. Figure 3.2 gives the tree of paths generated in this way by an adversary that maximizes the reach probability. For brevity, only the starting and ending state of the paths are given. Note that in this special case a finite tree suffices. The maximal probabilistic reachability respecting the cost bound is the summation over the paths: $0.8 + 0.16 + 0.016 = 0.976$.

Figure 3.1: An example WPTA.

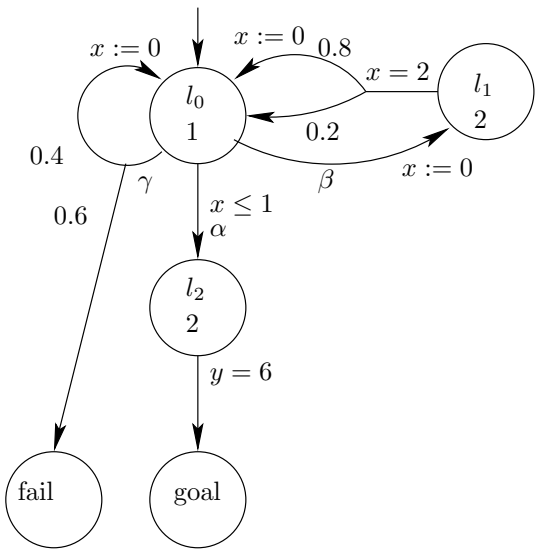
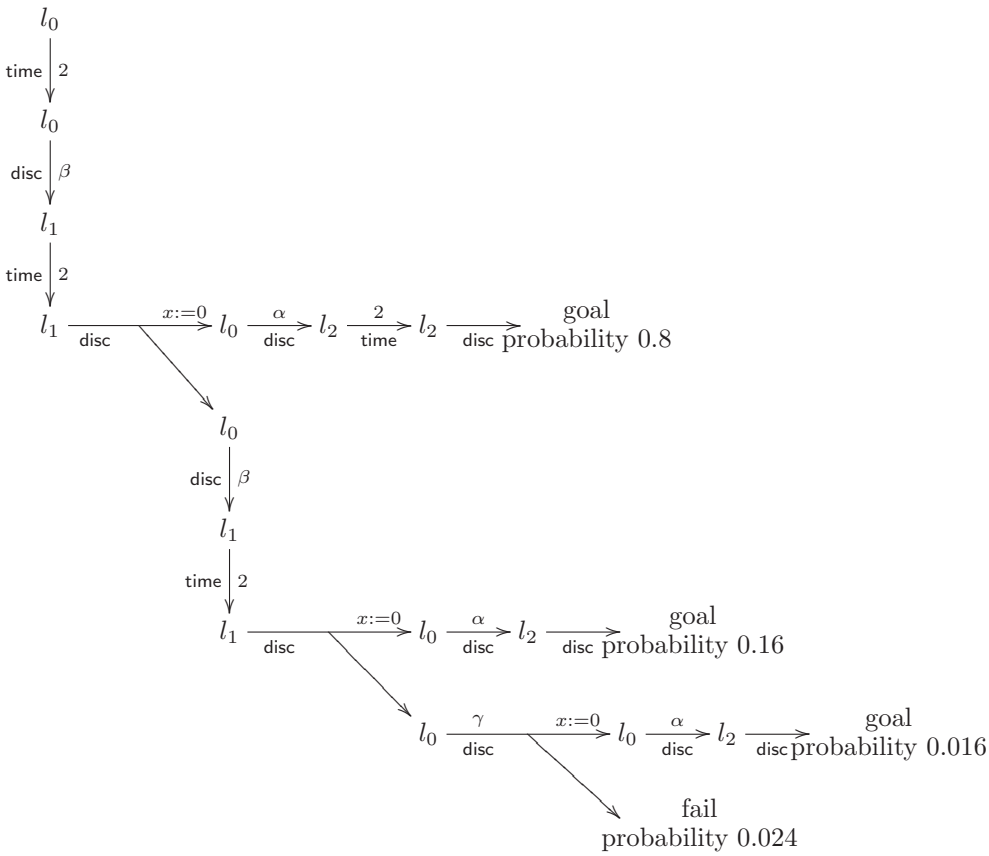


Figure 3.2: A tree of paths for the maximizing adversary.



3.2 Formal Problem Definition

We now present the formal definition for the cost-bounded probabilistic reachability problem. Like for PTA, only divergent adversaries are of interest, as they present realizable behaviour. However we can weaken this to T-divergent adversaries.

T-divergent paths in PTA are paths that are either divergent or reach a location from set T in some finite number of steps. A T-divergent adversary generates T-divergent paths with probability one [KNSS02]. T-divergence is a weaker notion than normal divergence. It is useful when there are no divergent paths that reach a target location, but we still want to compute probabilistic reachability for that target location. First we give the more general definition of F-divergent adversaries.

Definition 3.2.1 (F-divergence) Assume given a probabilistic system $PS = (S, Steps)$ and a set of states $F \subseteq S$. A path ω is *F-divergent* if it is either divergent or there exists $i \in \mathbb{N}$ such that $\omega(i) \in F$. An adversary of $A \in Adv_{PS}$ is F-divergent if and only if

$$Prob\{\omega \mid \omega \in Path^A_{\mathsf{full}} \text{ and } \omega \text{ is } F\text{-divergent}\} = 1$$

□

Let $T \subseteq L$ denote a *target set* of locations. An adversary is *T-divergent* if it is F-divergent with $F = \{(l, v, c) \mid l \in T \wedge v \in inv(l) \wedge c \geq 0\}$.

Definition 3.2.2 Assume given a $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$, target locations $T \subseteq L$, an operator $\sqsupseteq \in \{\geq, >\}$, a target probability $\lambda \in [0, 1]$, and cost bound κ. Let TPS denote the semantics of $WPTA$.
The *cost-bounded probabilistic reachability problem* $(T, \sqsupseteq, \lambda, \kappa)$ is the question:

"Is $Prob^A\{\omega \in Path^A_{\mathsf{full}} \mid \omega(0) = (l_0, \bar{0}, 0) \wedge (\exists i \in \mathbb{N}.(l, v, c) = \omega(i) \wedge l \in T \wedge c \leq \kappa)\} \sqsupseteq \lambda$ true, for some adversary $A \in Adv_{TPS}$?"

Using the definition of *ProbReach* the last expression equals $ProbReach^A(s_0, F) \sqsupseteq \lambda$, with $F = \{(l, v, c) \mid l \in T \wedge v \in \mathbb{R}^{\mathbb{X}}_+ \wedge c \leq \kappa\}$.
□

To solve the cost-bounded probabilistic reachability problem $(T, \sqsupseteq, \lambda, \kappa)$ for a $WPTA$, the problem of maximal probabilistic reachability (section 2.3.1) needs to be solved on the TPS semantics, where we are interested in reachability probability on states that have location in T and cost at most κ. Formally, if $MaxProbReach(s_0, F) \sqsupseteq \lambda$, with s_0 and F the same as above, the answer is "yes" and "no" otherwise.

The dual problem is that of *inevitability* of reaching T with a given probability λ or less. Clearly, such a property is true if it is not possible to reach the set of states with probability greater than λ.

3.3 Known Results for WPTA

Minimal cost reachability is decidable for WPTA, and thereby cost-bounded reachability is decidable. This can simply be verified by transforming the WPTA to a LPTA: every probabilistic edge can be replaced with a set of normal edges, where every normal edge has one possibility of reset and target location from the distribution of the probabilistic edge. Formally, for some $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$ let $WPTA_{LPTA} = (L, l_0, \mathbb{X}, inv, E, c, \dot{\$}, \$)$ be a linear priced timed automaton, where $E = E_{WPTA}$, and $\forall e \in E_{WPTA}.\$(e) = 0$ thus discrete cost changes are not used.

On the other hand, if all prices in a WPTA are ignored, it becomes a PTA, and maximal probabilistic reachability is computable. The notion of divergence on WPTA is equivalent to divergence on this PTA. Deciding non-Zenoness for WPTA can then be done by using the methods of [KNS03]. From minimal cost reachability on LPTA and maximal probabilistic reachability on PTA the

problem of cost-bounded probabilistic reachability on WPTA seems easy, but in the next chapter we will see that the combination of probabilities and prices makes the problem not so trivial.

3.4 Symbolic States for WPTA

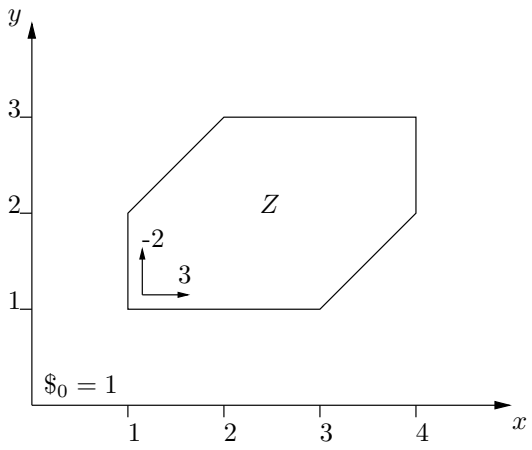
Symbolic states are needed to make an abstraction of the infinite state space. Here we present our symbolic states for WPTA, using the priced zones of [LBB+01]. Furthermore, a computable definition for the backward predecessor operation *pre* on the new symbolic states is given. Our symbolic states are useful in the algorithm of section 3.5, but section 3.6 will show some problems.

3.4.1 Priced zones

Definition 3.4.1 A **Priced Zone** is a tuple $Z = (N, \$_0, o)$ where N is a (normal) closed zone. The *offset cost* $\$_0 \in \mathbb{Z} \cup \{\infty\}$ describes the maximal cost associated with clock valuation $\bar{0}$, and $o : \mathbb{X} \to \mathbb{Z}$ assigns a cost-rate to every clock.
We identify a priced zone with (clock) valuation/cost-pairs, such that $(v, c) \in Z$, if and only if $v \in N$ and $0 \leq c \leq \mathit{maxcost}(v, Z)$. The function *maxcost* gives the maximal cost possible for a clock valuation:
$\mathit{maxcost}(v, Z) = \$_0 + \sum_{x \in \mathbb{X}} o(x) \cdot v(x)$ for $v \in N$. □

Figure 3.3 gives an example of a priced zone $Z = (N, \$_0, o)$ with two clocks, where $o(x) = 3$ and $o(y) = -2$. The cost-rates are written next to the arrows in the priced zone.

Figure 3.3: Example of a priced zone.



The priced zones we use are slightly different from those of [LBB+01]. Firstly, they identify priced zones with sets of clock valuations and a cost function that

defines the *minimum* cost for each clock valuation. Second, we define the offset cost on $\bar{0}$ instead of the offset clock valuation Δ_N of N, and allow it to be negative and unbounded, i. e. ∞.

For priced zones Z and Z' we define $Z' \subseteq Z$, if for any $(v, c) \in Z'$, then $(v, c) \in Z$. The inclusion relation can be decided by formulating a linear programming problem, see [LBB+01] for details. We have equality $Z = Z'$ between two priced zones if and only if $Z' \subseteq Z$ and $Z \subseteq Z'$. Likewise, we can define inequality.

Note that different priced zones can describe the same set of valuation/cost-pairs, but equality is decided on equality of the valuation/cost-pairs. For example: in the case of one clock x, let N be induced by clock constraint $(x \leq 1 \wedge x \geq 1)$, then priced zone $(N, 0, \{x \mapsto 1\})$ equals priced zone $(N, -1, \{x \mapsto 2\})$.

3.4.2 Predecessor with priced zones

With priced zones we are able to define a new sort of symbolic states. Here we show how a computable predecessor operation on these symbolic states can be defined. Before we present the predecessor operation in Definition 3.4.6, we need some definitions.

The *facets* of a closed zone are the sub-zones on the bounds of the zone. $\mathrm{LF}(N)$ denotes the set of lower facets of N. $\mathrm{UF}(N)$ denotes the set of upper facets of N. The formal definition is given below, it also gives the facets of priced zones, which are priced zones themselves and are called *priced facets*. Figure 3.4 gives an example zone, and all its facets. Lemma 3.4.3 states that facets can be used to decompose the $^{\downarrow}$ operation for normal zones.

Definition 3.4.2 (Facets) [LBB+01]. Let Z be a (priced) zone. For arbitrary integer $b \in \mathbb{N}$, we define the (priced) facets as follows:

$$\mathrm{LF}(Z) = \{F \mid \exists x \in \mathbb{X}.F = Z \wedge (x \leq b) \text{ and } (x \geq b) \text{ is a clock constraint in } Z\}$$
$$\mathrm{UF}(Z) = \{F \mid \exists x \in \mathbb{X}.F = Z \wedge (x \geq b) \text{ and } (x \leq b) \text{ is a clock constraint in } Z\}$$

□

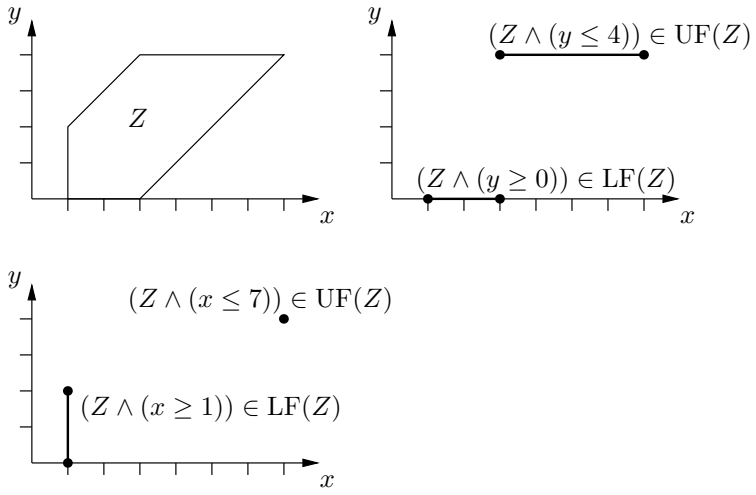
Lemma 3.4.3 [LBB+01]. Let N be a zone. Then:

- $N^{\downarrow} = N \cup \bigcup_{F \in \mathrm{LF}(N)} F^{\downarrow}$
- $N^{\downarrow} = \bigcup_{F \in \mathrm{UF}(N)} F^{\downarrow}$

□

The next definition is a priced version of the past-operator from Definition 2.1.14. It takes an arbitrary set of valuation/cost-pairs V. Now $V^{\downarrow q}$ defines all valuation/cost-pairs that can become a valuation/cost-pair in V by letting time elapse. Here q is the rate by which the cost increases when time elapses.

Figure 3.4: A zone with its facets.



Definition 3.4.4 (General Priced Past-Operator) Let $V = \{(v, c) \mid v \in \mathbb{R}_+^{\mathbb{X}} \wedge c \geq 0\}$ be a set of valuation/cost-pairs, and $q \in \mathbb{N}$.

$$V^{\downarrow q} = \{(v, c) \mid \exists d \geq 0.(v + d, c + d \cdot q) \in V\}$$

□

The next definition shows how the priced past-operator can be computed on priced facets. Figure 3.5 gives an example. We see priced facet F'. Which uses $\$'_0$ in its description. The values for y in F are bounded to a single value. Thus the cost-rate on y is changed to -1 in $F'^{\downarrow 2}$. The new cost in $\bar{0}$, i. e. $\$_0$ is computed from $maxcost(\Delta_N, F)$, which equals 4 in the figure. Note that in certain cases we need $\$_0$ to be negative.

Definition 3.4.5 (WPTA Priced Past-Operator) Let $F' = (N', \$'_0, o')$ be a priced facet, thus there is at least one clock x with clock constraint $(x \leq b \wedge x \geq b)$, choose x arbitrary. Let $q \in \mathbb{N}$ be a price. Then $F'^{\downarrow q} = (N, \$_0, o)$ where

- $N = N'^{\downarrow}$,
- $o(x) = q - \sum_{z \neq x} o'(z)$ and $o(y) = o'(y)$ for $y \neq x$,
- $\$_0 = maxcost(\Delta'_N, F') - \sum_{y \in \mathbb{X}} o(y) \cdot \Delta'_N(y)$.

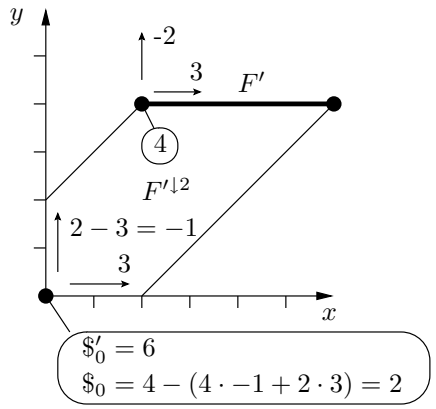
□

The idea of Definition 3.4.5 is that in the priced past we are interested in the maximal cost possible for clock valuations. Let Z be a priced upper or lower facet for some clock x, with clock constraint $x \leq b \wedge x \geq b$. In general to represent $Z^{\downarrow q}$ by a priced zone, we need $(N, \$_0, o)$, with $\sum_{z \in \mathbb{X}} o(z) = q$. Following Definition 3.4.4, for every valuation/cost-pair $(v, c) \in Z^{\downarrow q}$ there must exist d such that

$(v+d, c+dq) \in Z$. But as Z only permits a single value for clock x, d is unique for every valuation/cost-pair. We can see that clock valuations from N' must have the same maximum allowed cost in Z and $Z^{\downarrow q}$. Therefore $o(y) = o'(y)$ for $y \neq x$. We can choose $o(x)$ different because it plays no role of significance in Z, as x is bounded to a single value. Choose $o(x) = q - \sum_{z \neq x} o'(z)$, and in that way $\sum_{z \in \mathbb{X}} o(z) = q$. Note that by changing the cost-rate on x, the offset cost $\$'_0$ needs to be recomputed. We compute it from the maximum cost in offset valuation Δ'_N, which is also the maximum cost for valuation Δ'_N in $Z^{\downarrow q}$.

When more than one clock is bounded to a single value in Z, we can choose any of those clocks to play the role of x in the above. Depending on the choice of $x, Z^{\downarrow q}$ may be represented by a different priced zone.

Figure 3.5: An example priced facet F, and $F^{\downarrow 2}$. The values at the arrows give the cost-rates for the clocks. The values in the points are the maximal cost in those clock valuations.



Recall that a priced zone consists of a closed normal zone, and a price function. We are now ready to give computable predecessor operations that satisfy the general Definition 2.1.13. Note that *tpre* returns a set of predecessor symbolic states. The union of these give all predecessor states.

Definition 3.4.6 (predecessor in WPTA).
Let $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$. Let $l \in L$ with $q = \dot{\$}(l), Z = (N, \$_0, o)$ be a priced zone, and $e = (l', g, \cdot, r, l) \in E_{WPTA}$.

$$tpre(l,Z) = \begin{cases} \{(l,Z)\} \cup \{(l, H^{\downarrow q} \wedge inv(l)) \mid H \in \mathrm{LF}(Z)\} & \text{if } q > \sum_{x\in\mathbb{X}} o(x) \\ \{(l, H^{\downarrow q} \wedge inv(l)) \mid H \in \mathrm{UF}(Z)\} & \text{if } q < \sum_{x\in\mathbb{X}} o(x) \\ & \text{and } \mathrm{UF}(Z) \neq \emptyset \\ \{(l, ((N^{\downarrow}) \wedge inv(l), \infty, o))\} & \text{if } q < \sum_{x\in\mathbb{X}} o(x) \\ & \text{and } \mathrm{UF}(Z) = \emptyset \\ \{(l, ((N^{\downarrow}) \wedge inv(l), \$_0, o))\} & \text{if } q = \sum_{x\in\mathbb{X}} o(x) \end{cases}$$

$dpre_e(l,Z) = (l', Z')$, with $Z' = (([r := 0]N) \wedge g \wedge inv(l), \$_0, o')$

$$\text{and } o'(x) = \begin{cases} 0 & \text{if } x \in r \\ o(x) & \text{otherwise.} \end{cases}$$

$pre_e(\tau) = \{dpre_e(\sigma) \mid \sigma \in tpre(\tau)\}$

□

In Definition 3.4.6, the condition on the first case of *tpre* shows that letting time elapse is cheaper according to Z than according to q. Because we are interested in maximal cost on clock valuations by which some state in (l, Z) is reachable, the cheapest way of time elapse is of interest as in that case clock valuations with higher costs may still lead to (l, Z). Therefore we need to maintain Z in our description of the time predecessor. From Lemma 3.4.3 we have seen how the past-operator can be composed of applying the past-operator on facets. This is extended to our priced setting, where we take the priced past-operator on the lower priced facets of Z, thereby not interfering with the maximal cost assigned to clock valuations by Z. The left part of figure 3.6 shows an example in case of two clocks.

The second case of *tpre* is the counter-part of the first case. Now letting time elapse is cheaper according to q and the maximal costs described by Z should be abolished. We use the priced past-operator on the upper priced facets of Z. The condition makes sure these priced upper facets exist. The right part of figure 3.6 shows an example in case of two clocks.

The third case of *tpre* is the same as the second, but now no upper priced facets exist. Lack of upper priced facets means all clocks are unbounded, and an arbitrary amount of time may be spent for both (l, Z) and $tpre(l, Z)$. Because spending time is more expensive according to Z than according to q, we can always spend an amount of time according to q such that the cost becomes lower than the maximum defined by Z. Thus all costs are allowed in $tpre(l, Z)$, denoted by ∞. Note that o becomes unimportant.

The fourth case is a special case, where $tpre(l, Z)$ can be easily obtained by only extending the underlying normal zone, as spending time according to Z or q induces the same cost.

Figure 3.7 gives an example of the *dpre* operation on a priced zone, note that the zone has only one clock on the y-axis, and the cost is on the x-axis.

Figure 3.6: Example of the first two cases in *tpre* operation on priced zones.

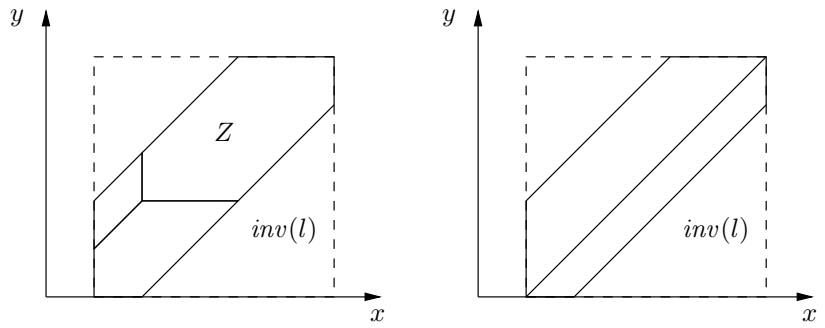
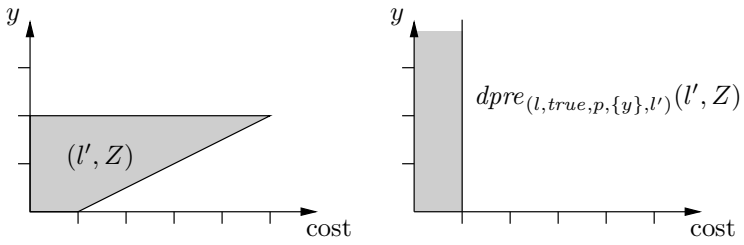


Figure 3.7: *dpre* operation on a priced zone with one clock y, and $inv(l) = true$. The cost is on the horizontal axis.



3.5 Backward Cost-Bounded Reachability

With the predecessor operations we are able to construct algorithm *BwReachability* on page 46, which decides cost-bounded reachability using backward analysis. The verification is performed on the underlying LPTA, i. e. $WPTA_{LPTA}$ as in section 3.3. Recall that this problem can be solved by forward analysis, but backward analysis is of interest, because the verification procedure of chapter 5 will also use backward analysis. The target is not a set of locations but a set of symbolic states ψ. This more general approach makes the computation nothing more difficult. If we want to compute reachability for a set of locations T, and cost bound κ, we initialize the target set as $\psi = \{(l_T, (inv(l_T), \kappa, o)) \mid l_T \in T \wedge \forall x \in \mathbb{X}.o(x) = 0\}$. *BwReachability* returns the set of symbolic states *Visited*. The time predecessors of symbolic states in *Visited* contain all states from which target ψ is reachable with a positive probability.

The fact that *BwReachability* terminates in a finite number of steps is important. Otherwise, it would be hopeless to search for an algorithm computing *cost-bounded maximal probabilistic reachability*, using predecessor operations. Theorem 3.5.1 states the correctness of the *BwReachability*.

We think the usage of priced zones will lead to a fast implementation of

BwReachability. An implementation can first generate the state space of the underlying unpriced zones that would be generated by using Definition 3.4.6. If the target is unreachable from the starting state already a negative verdict can be given. The priced zones use the normal zones of the generated set, and two priced zones can share the structure for their normal zone.

Finally, we think that *BwReachability* can be used on LPTA that include difference constraints (not diagonal-free).

Algorithm 1: *BwReachability*(ψ)

(1) **Input:** $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$,
(2) target set of symbolic states ψ
Output: symbolic state space *Visited*
$Visited := \emptyset$ //set of generated symbolic states
(3) $Waiting := \psi$ //set of symbolic states waiting to be explored
(4) **repeat**
(5) **get** and **remove** τ from *Waiting*
(6) $Visited := Visited \cup \{\tau\}$
(7) **foreach** $e = (l, g, p, r, l') \in E_{WPTA}$ with $l' = loc(\tau)$
(8) **foreach** $\sigma \in pre_e(\tau)$
(9) **if** $\forall \iota \in Visited.\sigma \not\subseteq \iota$
(10) $Waiting := Waiting \cup \{\sigma\}$
(11) **until** $Waiting = \emptyset$
(12) **return** *Visited*

The condition $\forall \iota \in Visited.\sigma \not\subseteq \iota$ on line 9 is sufficient to guarantee termination. *BwReachability* is in fact very generic. For example, if we use symbolic states with normal zones and the predecessor operation of Definition 2.1.14, *BwReachability* becomes a backward exploration algorithm for timed automata. The only thing that has to be taken into account, is that *pre* for timed automata does not generate a set of predecessors, but just one. When we have a representation for symbolic states and predecessor operations, even other systems like hybrid automata can be handled by *BwReachability*. The problem is that not for all kinds of systems termination is guaranteed.

Theorem 3.5.1 Let $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$, and ψ be the target set of symbolic states. Let $TPS = (S, Steps)$ denote the semantics of *WPTA*. Algorithm *BwReachability* terminates in a finite number of steps. The time predecessors of symbolic states in *Visited*, that is the set $\{tpre(\tau) \mid \tau \in Visited\}$, contains all states from which target ψ is reachable with a positive probability. Formally, for any $s \in S, MaxProbReach(s, \psi) > 0$ if and only if there exists $\sigma \in Visited$ such that $s \in tpre(\sigma)$. □

Proof By definition of the predecessor operation, if *BwReachability* terminates, $\{tpre(\tau) \mid \tau \in Visited\}$ contains all states from which ψ is reachable. The proof that *BwReachability* terminates is more involved, and is given in the next section. □

The following theorem is a simplified version of Theorem 3.5.1.

Theorem 3.5.2 Let $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$, and ψ be a target set of symbolic states with the following condition: for each symbolic state $(l, (N, \$_0, o)), \forall x \in \mathbb{X}.o(x) = 0$. Let $TPS = (S, Steps)$ denote the semantics of *WPTA*. Algorithm *BwReachability* is altered by replacing line 9 with:

$$\textbf{if } \forall \iota \in Visited.\sigma \neq \iota \wedge \sigma \neq \emptyset$$

The altered algorithm generates symbolic states such that all states able to reach the target are in the time predecessor of some symbolic state, just like the unaltered version. More important, the altered algorithm terminates in a finite number of steps. □

Proof Only termination needs to be proven. By using lemmas 3.5.6 and 3.5.7 all priced zones that are unbounded w. r. t. some clock x will have $o(x) = 0$. By using the alternative representation of priced zones of the proof of Theorem 3.5.1, we see that $[c_0, \ldots, c_n] < 0$ is for all priced zones a sufficient condition for concluding that the priced zone is empty. Thus the number of generated priced zones is finite and the theorem holds. □

The following lemma is a very straightforward result.

Lemma 3.5.3 When allowing priced zones with $\$_0 \in \mathbb{Q}$, all the above results on priced zones including Theorem 3.5.1 still hold. □

Proof Given a priced zone $Z = (N, \$_0, o)$ with $\$_0 \in \mathbb{Q}$. Let $f = \$_0 - \lfloor \$_0 \rfloor$ be the positive fraction. Construct original priced zone $Y = (N, \lfloor \$_0 \rfloor, o)$ from Z. For priced zones we have operations $Z^{\downarrow q}, [r := 0]Z$, and $Z \wedge g$ for some constraint g. All operations can be applied to Y. If $Y' = (M, \$'_0, o')$ is a resulting priced zone, then $(M, \$'_0 + f, o')$ is the result that would be given when the operation was directly applied to Z. □

3.5.1 Proof of termination of algorithm *BwReachability*

To prepare the proof, we need a few lemmas.

Any zone N with clocks $\mathbb{X} = \{x_1, \ldots, x_n\}$ describes a convex polyhedron in space $\mathbb{Z}^n$. Clocks are numbered, and valuations can be given as vectors $\vec{v} = (a_1, \ldots, a_n)^T$, such that $v(x_i) = a_i$. Let $vertices(N) \subset \mathbb{Z}^n$ be the set of clock valuations that give the vertices of the polyhedron. Let $V \subseteq \mathbb{Z}$ be a set of clock valuations. $convexhull(V)$ is the minimal convex polyhedron that includes all clock valuations in V. From mathematical geometry we know that $convexhull(vertices(N)) \subseteq N$ for any N. Note that the convex hull is possibly not representable as a zone.

Lemma 3.5.4 Let zone N be unbounded w. r. t. all clocks in U, then $v \in N$ implies there exists $v' \in convexhull(vertices(N))$ such that

$$(v - v')(x) \begin{cases} \geq 0 & \text{if } x \in U \\ = 0 & \text{otherwise} \end{cases}$$

□

Proof For $x \in U$, let $x_{\min}$ be s. t. $(x \geq x_{\min})$ is a constraint in N. Such a constraint always exists, because of the canonical form of N, and the fact that at least $x \geq 0$ should hold. Choose

$$v'(x) = \begin{cases} x_{\min} & \text{if } x \in U \\ v(x) & \text{otherwise} \end{cases}$$

□

Lemma 3.5.5 Given two priced zones $Z = (N, \$_0, o)$ and $Z' = (N, \$'_0, o')$, that have the same underlying normal zone. Let $U \subseteq 2^{\mathbb{X}}$ be the maximal set of clocks, such that Z and Z' are unbounded for all clocks in U. Now $Z' \subseteq Z$ if for all $v \in \mathit{vertices}(N)$ we have $\mathit{maxcost}(v, Z') \leq \mathit{maxcost}(v, Z)$, and for all $x \in U$ we have $o'(x) \leq o(x)$. □

Proof Let Z, Z' and U be as in the lemma. Assume for all $v \in \mathit{vertices}(N)$ we have $\mathit{maxcost}(v, Z') \leq \mathit{maxcost}(v, Z)$, and for all $x \in U$ we have $o'(x) \leq o(x)$. From mathematical geometry and linearity of priced zones, we conclude that for all $v' \in \mathit{convexhull}(\mathit{vertices}(N)), \mathit{maxcost}(v', Z') \leq \mathit{maxcost}(v', Z)$. Take arbitrary $u \in N$, by Lemma 3.5.4 there exists $u' \in \mathit{convexhull}(\mathit{vertices}(N))$ such that

$$(u - u')(x) \begin{cases} \geq 0 & \text{if } x \in U \\ = 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \mathit{maxcost}(u, Z') &= \mathit{maxcost}(u' + (u - u'), Z') \\ &= \mathit{maxcost}(u', Z') + \sum_{y \in \mathbb{X}} o'(y)(u - u')(y) \\ &= \mathit{maxcost}(u', Z') + \sum_{y \in U} o'(y)(u - u')(y) \\ &\leq \mathit{maxcost}(u', Z) + \sum_{y \in U} o(y)(u - u')(y) \\ &= \mathit{maxcost}(u' + (u - u'), Z) = \mathit{maxcost}(u, Z) \end{aligned}$$

□

From now we say a symbolic state (l, Z) is *(un)bounded* w. r. t. clock x if Z is (un)bounded w. r. t. x.

Lemma 3.5.6 If some symbolic state (l', Z') is bounded w. r. t. x, then for all $(N, \$_0, o) \in \mathit{pre}_e(l', Z')$ that are unbounded w. r. t. x, we have that $o(x) = 0$. □

Proof Given some symbolic state that is bounded w. r. t. x. All symbolic states generated from it by operation *tpre* are also bounded w. r. t. x. Choose arbitrary (l', Z) from the symbolic states generated by *tpre*, and let $e = (l, g, p, r, l') \in E_{WPTA}$. Now operation dpre_e can only generate a priced zone that is unbounded w. r. t. x if $x \in r$. Assume $x \in r$ and let $(l, (N, \$_0, o)) = \mathit{dpre}_e(l', Z)$, then $o(x) = 0$. □

Lemma 3.5.7 If some symbolic state $(l', Z') = (l', (N', \$'_0, o'))$ is unbounded w. r. t. x, then for all $(l, (N, \$_0, o)) \in \mathit{pre}_e(l', Z')$ that are unbounded w. r. t. x, we have that $o(x) = 0 \vee o(x) = o'(x)$. □

Proof Given some symbolic state (l, Z') unbounded w. r. t. x, where $Z' = (N', \$'_0, o')$. Take an arbitrary symbolic state $(l, (N, \$_0, o)) \in \mathit{tpre}(l, Z')$ which is unbounded w. r. t. x. Let $q = \dot{\$}(l)$. We have to consider the four cases of *tpre*, see Definition 3.4.6.

- if $q = \sum_{z \in \mathbb{X}} o(z)$ then $o(x) = o'(x)$.
- if $q < \sum_{z \in \mathbb{X}} o(z)$ and $\mathrm{UF}(Z') = \emptyset$ then $o(x) = o'(x)$.
- if $q > \sum_{z \in \mathbb{X}} o(z)$ then if $(N, \$_0, o) = Z'$ the lemma holds. Otherwise, $(N, \$_0, o) = H^{\downarrow q} \wedge \mathit{inv}(l)$ for some $H \in \mathrm{LF}(Z')$, and there are two possibilities:
  1. H is a lower priced facet obtained by strengthening Z with a constraint of the form $(y \leq n)$, where $y \neq x, n \in \mathbb{N}$. From Definition 3.4.5 we have that $o(x) = o'(x)$.
  2. H is a lower priced facet obtained by strengthening Z with a constraint of the form $(x \leq n)$, where $n \in \mathbb{N}$. From Definition 3.4.5 we see that the $H^{\downarrow q}$ becomes bounded w. r. t. x. From the proof of Lemma 3.5.6, if *dpre* generates a priced zone that is unbounded w. r. t. x, then $o(x) = 0$.
- if $q < \sum_{z \in \mathbb{X}} o(z)$ and $\mathrm{UF}(Z') \neq \emptyset$ then H cannot be obtained by strengthening Z with some constraint $(x \geq n)$, because there is no such upper facet due to unboundedness w. r. t. x. When H is obtained by strengthening Z with some constraint $(y \geq n)$, with $y \neq x$, the situation is the same as in 1.

What remains to be considered (except for case 2.) is operation *dpre*. From Definition 3.4.6 it is clear that *dpre* can only change the cost-rate on clock x to zero or leave it unchanged. By combining this observation with that on *tpre* the proof is complete. □

Completing the proof

Let $\$^{\max}_{\psi} \in \mathbb{N} \cup \{\infty\}$ denote the maximum on the cost of all states in ψ. If $\$^{\max}_{\psi} = \infty$, this means that the cost can get arbitrarily high. Let $\mathit{rates}_{\psi} \subseteq \mathbb{N}$ be the finite set of all cost-rates used in the priced zones of ψ. By combining Lemma 3.5.6 and 3.5.7, all priced zones generated by *BwReachability* that are unbounded w. r. t. some clock x, will have a cost-rate $o(x) \in \mathit{rates}_{\psi} \cup \{0\}$. For priced zone $Z = (N, \$_0, o)$, if a vertex $v \in N$ has $\mathit{maxcost}(v, Z) = \infty$ then all vertices $u \in N$ have $\mathit{maxcost}(u, Z) = \infty$.

From the general definition of *tpre* and the semantics of LPTA that say cost can only grow, we know that the maximum on the cost of all states in a symbolic state is always higher or equal to the maximum on the cost of all states in some predecessor symbolic state. By definition of the predecessor operation, when all vertices of a symbolic state have maximum cost in $\{\ldots, -1, 0, 1, \ldots, \$^{\max}_{\psi}\}$ then vertices of a predecessor symbolic state have maximal cost in $\{\ldots, -1, 0, 1, \ldots, \$^{\max}_{\psi}\}$. When all vertices of a symbolic state have maximum cost ∞, then all vertices of

the predecessor operation have maximum cost ∞. We conclude that maximum costs on vertices are elements from $\{\ldots, -1, 0, 1, \ldots, \$_\psi^{\max}, \infty\}$.

To complete the proof we choose another representation for priced zones. Priced zone $Z = (N, \$_0, o)$ can be represented as a tuple $(N, [v_1, \ldots, v_n], [c_0, \ldots, c_n], \theta)$, where $[v_1, \ldots, v_n]$ are the vertices of N in some order, $[c_0, \ldots, c_n]$ are the maximal costs of the vertices in the same order, and $\theta : \mathbb{X} \to \mathbb{Z}$ is a function such that

$$\theta(x) = \begin{cases} o(x) & \text{if } N \text{ is unbounded w. r. t. } x \\ 0 & \text{otherwise.} \end{cases}$$

For arbitrary symbolic state $(l, (N, [v_1, \ldots, v_n], [c_0, \ldots, c_n], \theta))$, there are finitely many possible values for all of the following:

- l, as the set of locations L is finite,
- N, as from the theory on timed automata [HNSY92], we know that the set of normal zones generated by *BwReachability* is finite,
- θ, as for all clocks $\theta(x) \in \mathit{rates}_\psi \cup \{0\}$,
- $[v_1, \ldots, v_n]$, with the same argument as for N.

The proof is completed by showing that for fixed $l, N, [v_1, \ldots, v_n], \theta$, it is impossible to generate an infinite sequence of symbolic states $[\sigma_i]_{0 \le i \le \infty}$ with $\sigma_i = (l, (N, [v_1, \ldots, v_n], [c_0, \ldots, c_n]_i, \theta))$, without having $\sigma_j \supseteq \sigma_k$ for some $j < k$.

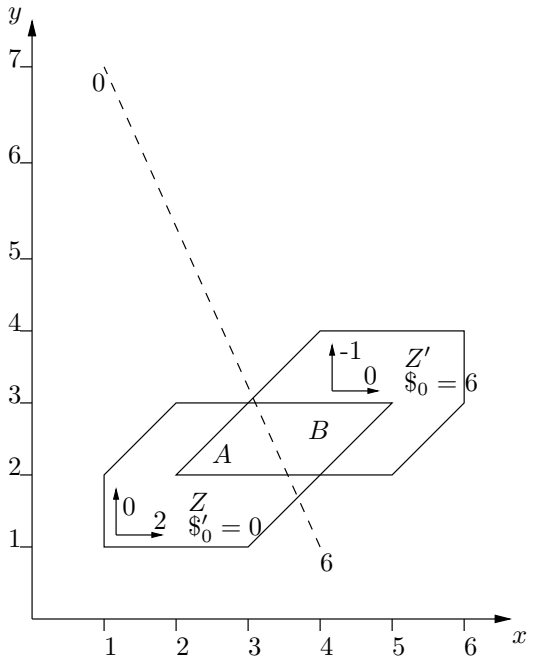
$\geq$ is a well-quasi-order (sometimes called well partial order) on $\{\ldots, -1, 0, 1, \ldots, \$_\psi^{\max}, \infty\}$. Application of Higman's Lemma [Hig52] gives that point-wise $\geq$ is a well-quasi-order on tuples of elements of $\{\ldots, -1, 0, 1, \ldots, \$_\psi^{\max}, \infty\}$. From Lemma 3.5.5 we see that $\sigma_j \supseteq \sigma_k$ if $[c_0, \ldots, c_n]_j \geq [c_0, \ldots, c_n]_k$ point-wise. Thus $\supseteq$ is a well-quasi-order on $[\sigma_i]_{0 \le i \le \infty}$ and by definition of well-quasi-order $\sigma_j \supseteq \sigma_k$ for some $j < k$. The use of Higman's Lemma is similar to that in [BFH+01].

3.6 Intersections of Symbolic States

We present a new abstraction called *multi-priced zones*. In chapter 5 we will see that the intersection between symbolic states is needed. An intersection of symbolic states gives all common states. In the computation of intersections, it is clear that the location of symbolic states plays no role of importance, because for the intersection to be non-empty locations must be the same.

Priced zones with n clocks can be seen as convex polyhedra in $\mathbb{Q}^{n+1}$. The intersection of priced zones, is by rules of geometry, also a convex polyhedron in $\mathbb{Q}^{n+1}$. But now constraints containing equations like for example $x \le \frac{1}{2}$ or $x - 2y \le 3$ might be needed, see figure 3.8 that shows two priced zones, and their intersection. Figure 3.9 shows the time predecessor of an intersection of priced zones. Rational numbers are needed on the cost rate function o to describe the time predecessor.

Figure 3.8: Intersection of two priced zones Z and Z', clock valuations on the dashed line have the same maximal cost on both Z and Z'. The clock valuations of the intersection are $A \cup B$. In A the price function of Z is needed. In B that of Z'.



From literature [AHH96, HPR94] it is known that convex polyhedra in $\mathbb{Q}^{n+1}$ are closed under time predecessor, discrete predecessor, and conjunction. Convex polyhedra can be described by a conjunction of linear formulas on clocks and cost. Formally, when $\mathbb{X} = \{x_1, \ldots, x_n\}$ and c is the cost variable, a *linear formula* ϕ on clocks and cost is of the form $\phi : ac + b_1x_1 + \cdots + b_nx_n \leq b_0$, with $a, b_0, \ldots, b_n \in \mathbb{N}$. A valuation/cost-pair $(\vec{v}, c)$ satisfies ϕ, written $(\vec{v}, c) \vdash \phi$ if the formula holds for the values of $\vec{v}$ and c. [AHH96] presents a method that uses quantifier elimination to compute predecessors on polyhedra that are described by a conjunction of linear formula. [HPR94] manipulates polyhedra by switching between two different representations of polyhedra. The first representation is equivalent to the conjunction of linear formula. The second representation uses spanning vertices and generator vectors. [AHH96] suggests that the method of [HPR94] will have better performance.

To be able to describe intersections we may use *multi-priced* zones. These are zones with multiple cost functions defined on them.

Definition 3.6.1 A **Multi-Priced Zone (mp-zone)** is a tuple $M = (N, \Phi)$ where N is a (normal) zone. Let c denote the cost variable. Φ is a finite set of linear functions such that for all $\phi \in \Phi$ we have that $\phi : ac \leq b_1 x_1 + \cdots + b_n x_n + b_0$, with $a, b_0, \ldots, b_n \in \mathbb{N}$ and $a > 0$. We require that Φ is minimal in the sense that $\forall \phi \in \Phi. \bigwedge(\Phi \backslash \phi) \not\Rightarrow \bigwedge \Phi$. Write $(v, c) \in M$ if and only if $v \in N$ and $(v, c) \vdash \bigwedge \Phi \wedge (c \geq 0)$. □

From the definition of mp-zones we see that the formulas in Φ are not equivalent to any constraint of N, because $a > 0$ implies that cost always is a nontrivial variable in the formula. Mp-zones are a class of polyhedra. The predecessor operations are redefined in the definition below. Operators ${}^{\downarrow q}, [r := 0]\cdot$, and conjunction can be implemented as operations on polyhedra, we make no assumptions on the exact implementations.

Definition 3.6.2 (Predecessor on Multi Priced Zones) Let $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$. Let $l \in L$ with $q = \dot{\$}(l), Z = (N, \$_0, o)$ be a multi-priced zone, and $e = (l', g, \cdot, r, l) \in E_{WPTA}$.

$$
\begin{aligned}
&tpre(l, Z) = (l, (Z^{\downarrow q}) \wedge inv(l)) \\
&dpre_e(l, Z) = (l', ([r := 0]Z) \wedge g \wedge inv(l)) \\
&pre_e = dpre_e \circ tpre
\end{aligned}
$$

□

Theorem 3.6.3 states that mp-zones are closed under conjunction and *pre*. Only one simple assumption is that the conjunction of linear formulas must be minimal in the sense of Definition 3.6.1. Algorithm *CBMaxReachAlg* on page 73 of chapter 5 needs to compute inclusion or inequality between symbolic states. Because we can use mp-zones these computations can typically be performed faster by first comparing the normal zones of the two mp-zones. Moreover, *CBMaxReachAlg* generates a symbolic state space. By using mp-zones we can first explore the state space of the underlying normal zones. This terminates from theory on timed automata in a finite number of steps. Then we can calculate the behaviour when the linear price functions are included. Whether more recent approaches than [HPR94], in combination with the fact that mp-zones are a subclass of general polyhedra, will lead to more improvements on algorithms remains to be investigated.

Theorem 3.6.3 Given *WPTA*, mp-zone M, location l and edge $e \in E_{WPTA}$.

- Given another mp-zone K, then $M \cap K$ can be described using a mp-zone.
- $S = pre_e(M)$ can be described using a mp-zone.

□

Proof of Theorem 3.6.3

Let ϕ be the linear formula on clocks and cost that describes M as a convex polyhedron. Let ψ be the same for K. Now $M \cap K$ can be described by the formula $\phi \wedge \psi$. This formula is of course representable as mp-zone. This proves

the first bullet of the theorem.

Operation *pre* is composed of *dpre* and *tpre*, which in turn are composed of: conjunction with a clock constraint, $^{\downarrow q}$, and $[r := 0]\cdot$. Conjunction with a clock constraint is nothing more than the first bullet of the theorem, using a priced zone that permits arbitrary cost.

Given mp-zone $M, [r := 0]M$ can be constructed in three steps. Figure 3.10 gives an example of this construction.

1. $M_1 = M \wedge \bigwedge\{(x = 0) \mid x \in r\}$, where $(x = 0)$ is a constraint. Clearly M_1 is a mp-zone.
2. Let $M_1 = (N, \Phi)$. Construct MP-zone $M_2 = (N, \Phi')$, where Φ' is obtained by reducing all formulas of Φ. For all $\phi \in \Phi : ac \leq b_1x_1 + \cdots + b_nx_n + b_0$. All terms b_ix_i, with $x_i \in r$, are removed.
3. M_3 is obtained by removing all bounds in M_2 of the form $x \leq b$, for $x \in r$.

Now $M_3 = [r := 0]M$. Clearly M_3 is a mp-zone, and mp-zones are closed under $[r := 0]\cdot$.

For operation $^{\downarrow q}$ let $H = M^{\downarrow q}$ for some mp-zone M. We know that H is a convex polyhedron in $\mathbb{Q}^{n+1}$, where n is the number of clocks. When only observing clock valuations, by Definition 2.1.13:

$$\forall(v, c) \in H.\exists d \geq 0.\exists c' \geq 0.(v + d, c') \in M \tag{3.1}$$

We see that all possible clock valuations of H can be described using a normal zone. From Definition 3.4.4 we have:

$$\forall(v, c) \in H.\exists d \geq 0.(v + d, c + dq) \in M$$

We see that inclusion of a valuation/cost-pair in H, relies in the first place on (3.1) and then only on its cost. As costs are bounded by the price functions, we see that multi-priced zones are sufficient to describe the set of values obtained after applying $^{\downarrow q}$.

Figure 3.9: The intersection of two priced zones Z and Z' in the first picture. Its time predecessor $(Z \cup Z')^{\downarrow 1}$ in the second picture. Clock valuations on the dashed line have the same maximal cost on both Z and Z'.

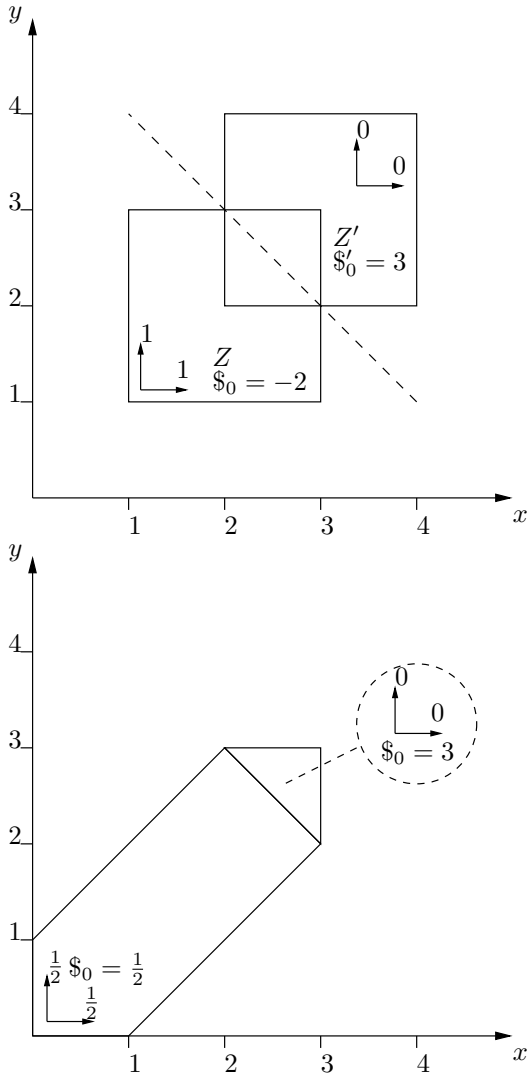
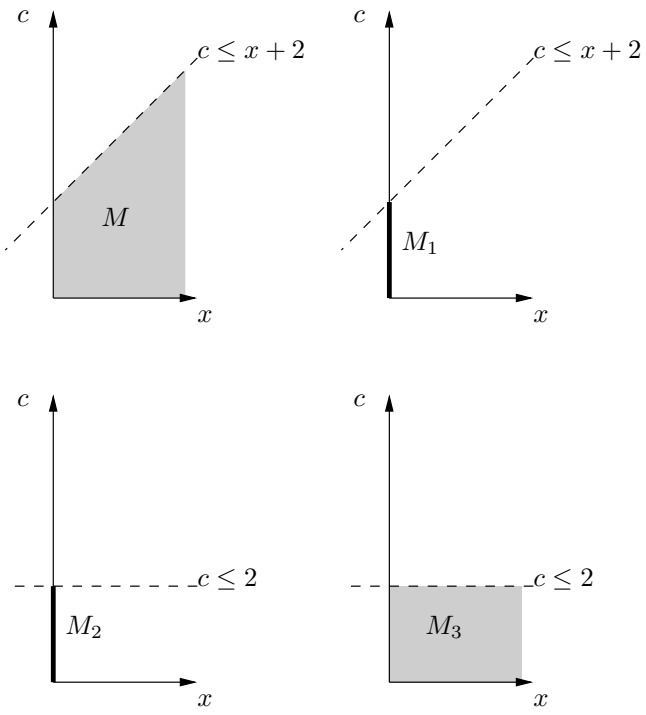


Figure 3.10: Example construction of $[r := 0]M$ for multi-priced zone M.



Chapter 4

A first naive algorithm

This chapter presents a first naive algorithm to calculate cost-bounded maximal probabilistic reachability. First, we introduce our idea and algorithm. Unfortunately, the algorithm is incorrect: we give a counter-example. Next the problem of the counter-example is analyzed, and we proof that the algorithm can be used to compute an upper bound. Finally some ideas of how to compute a lower bound are given.

4.1 Idea to Solve Cost-Bounded Probabilistic Reachability

The idea is to construct a probabilistic region graph as in section 2.3.3 with the priced regions of section 2.2.1, where the cost-bounded maximal probabilistic reachability computed on this graph is equal to the cost-bounded maximal probabilistic reachability of the WPTA. Note that in this chapter we will *not* use symbolic states based on priced zones or multi-priced zones.

The first issue is that priced regions are only defined for bounded regions. Therefore we need a similar construction as for LPTA (Theorem 2 in [BFH$^+$01]), to construct a bounded equivalent for some WPTA. A WPTA is bounded if the clocks can't get arbitrary high, thus all regions will be bounded. [BFH$^+$01] constructs the bounded equivalent for LPTA, by introducing constant updates on clocks in extension to clock resets: edges may now reset a clock to some natural number. However, they do not alter their symbolic semantics for LPTA to capture constant updates, but this should be no big problem.

We do not introduce constant updates because this is stronger than necessary, and interferes with our definition of WPTA. Instead, we introduce $WPTA^\delta$ that has the same probabilistic edges as $WPTA$, and in addition implicit probabilistic edges that have exactly the same semantics as the edges with constant updates of [BFH$^+$01]. We proceed by giving the semantics of $WPTA^\delta$.

Definition 4.1.1 Assume given $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$, with clock ceiling function k. We construct the special bounded WPTA $WPTA^\delta = (L, l_0, \mathbb{X}, inv', pE, c, \dot{\$})$, where $\forall l \in L.inv'(l) = inv(l) \bigwedge_{x \in \mathbb{X}} (x \leq k(x) + 2)$. The semantics for $WPTA^\delta$ are defined by $TPS = (S, Steps)$, and are the same as for normal WPTA, but with the following extra probabilistic transitions:

$((l, v, c), 0, \mathsf{disc}, \mu) \in Steps$ if

$$(l, v, c) \in S, \exists x \in \mathbb{X}.v(x) = k(x) + 2, \mu(l, v', c) = 1, \text{ and}$$

$$\forall y \in \mathbb{X}.v'(y) = \begin{cases} k(x) + 1 & \text{if } y = x \\ v(y) & \text{otherwise} \end{cases}$$

□

Theorem 4.1.2 Assume given $WPTA$ and its special bounded version $WPTA^\delta$ of Definition 4.1.1. For any location $l \in L, l$ is reachable with cost k and probability p in $WPTA$ if and only if l is reachable with cost k and probability p in $WPTA^\delta$.

Proof: by Theorem 2 in [BFH+01] and the fact that the new transitions in the semantics of $WPTA^\delta$ do not alter the probability on paths, as their probability is one. □

In our algorithm we use the symbolic semantics of the underlying LPTA of $WPTA^\delta$. We denote the underlying LPTA by $WPTA^\delta_{LPTA}$, and construct it as in section 3.3, with edges E_{WPTA}, but with the difference that the implicit edges will also have equivalent edges in $WPTA^\delta_{LPTA}$. To capture the new transitions in $WPTA^\delta_{LPTA}$ the symbolic semantics of Definition 2.2.10 needs to be extended with a new type of transitions. First we need the following new operation on priced regions.

Definition 4.1.3 (Priced Region Translate) Given a priced region $R = (h, [r_0 \dots r_n], [c_0 \dots c_n])$ and a function $f : \mathbb{X} \to \mathbb{Z}$ giving for every clock the value by which it should be translated, R translated by f is the priced region $translate(R, f) = (h', [r_0, \dots, r_n], [c_0, \dots, c_n])$, where $\forall x \in \mathbb{X}.h'(x) = h(x) + f(x)$. □

Now we are able to give the symbolic semantics for $WPTA^\delta_{LPTA}$ using priced regions, similar to Definition 2.2.10.

Definition 4.1.4 The **symbolic semantics of** $WPTA^{\delta}_{LPTA} = (L, l_0, \mathbb{X}, inv, E, c, \dot{\$}, \$)$ is defined as a labeled transition system where symbolic states are of the form (l, R), with l a location and R a priced region satisfying $inv(l)$. The starting symbolic state is $(l_0, (\bar{0}, [\mathbb{X}], [0]))$. The transition relation is as follows:

- $(l, R) \xrightarrow{\epsilon} (l, delay(R, \dot{\$}(l)))$ if $delay(R, \dot{\$}(l)) \in inv(l)$,
- $(l, R) \xrightarrow{(l,g,r,l')} (l', R')$ if there exists g, r such that $(l \xrightarrow{g,r} l') \in E, R \subseteq g, R' = self(inc(reset(r, R), \$(l, g, r, l')), \dot{\$}(l'))$,
- $(l, R) \xrightarrow{\delta} (l, R')$ if there exists $x \in \mathbb{X}$ such that $R \wedge (k(x) + 2) \neq \emptyset$ and $R' = translate(R, f)$, where $\forall y \in \mathbb{X}.f(y) = \begin{cases} -1 & \text{if } y = x \\ 0 & \text{otherwise} \end{cases}$.

□

The new discrete transitions of Definition 4.1.1 are correctly mimicked by the δ-transitions of Definition 4.1.4. We have probabilistic transition $((l, v, c), 0, \mathsf{disc}, \{(l, v', c) \mapsto 1\}) \in Steps$ as in Definition 4.1.1 if and only if there exist R, R' such that $v \in R, v' \in R'$, and $(l, R) \xrightarrow{\delta} (l, R')$. Now Lemma 2.2.11 will hold for $WPTA^{\delta}_{LPTA}$, using the symbolic semantics of 4.1.4.

4.2 The Algorithm

Algorithm *FwdBoundedReach* on page 61 generates a symbolic state space, for a WPTA, with cost bound κ. Symbolic states consist of a location and a priced region, and are generated by forward analysis using Definition 4.1.4. Recall that these symbolic states are *not* sets of states, but rather a location and a set of clock valuations with a minimal cost function. We construct a *priced* probabilistic region graph (PPRG) on the symbolic state space, similar to the construction of the probabilistic region graph in section 2.3.3. The idea was that the PPRG could be used to calculate cost-bounded maximal probabilistic reachabilities, but a counter-example will show the **incorrectness** of this approach. However, it can be proven that the returned probability is an *upper bound* on cost-bounded maximal probabilistic reachability. With an upper bound we may be able to say that the answer to some cost-bounded probabilistic reachability problem is "no", but we cannot give a positive verdict.

The symbolic semantics of Definition 4.1.4 is infinite. Since region equivalence is a finite time-abstracting bisimulation, the number of generated unpriced regions is finite, but costs are unbounded. The algorithm of [BFH+01] terminates by stating that in a finite number of steps only 'more expensive' symbolic states than the symbolic states already explored will be generated. These 'more expensive' symbolic states are not added to the symbolic state space, as only a single path reaching a target location with minimal cost is of interest.

For cost-bounded maximal probabilistic reachability we are interested in *all* paths, including non-minimal ones, under some adversary that lead to a target location, unless their cost is higher than the cost bound. All these paths are of

interest, as together they define the reachability probability.

FwdBoundedReach terminates when no new symbolic states are added. To guarantee termination, symbolic states with the same underlying normal region, that have for every vertex in the priced region the same costs or both costs above κ, are regarded cost-equivalent. There are only a finite number of cost-equivalent priced regions, thus guaranteeing termination. The next definition defines cost-equivalence.

Definition 4.2.1 Priced regions $(h, [r_0 \dots r_n], [c_0 \dots c_n])$ and $(h', [r'_0 \dots r'_n], [c'_0 \dots c'_n])$ are **cost-equivalent** w. r. t. cost bound $\kappa \in \mathbb{N}$ if and only if:

- their unpriced regions are the same: $(h, [r_0 \dots r_n]) = (h', [r'_0 \dots r'_n])$, and
- for all $0 \leq i \leq n$ we have $c_i = c'_i$ or $c_i > \kappa \wedge c'_i > \kappa$

□

The next definition gives a canonical form for priced regions.

Definition 4.2.2 (Cost Bound Normalization) Let $R = (h, [r_0 \dots r_n], [c_0 \dots c_n])$ be a priced region, and $\kappa \in \mathbb{N}$. $pnorm_\kappa(R) = (h, [r_0 \dots r_n], [c'_0 \dots c'_n])$ where $c'_i = \min(c_i, \kappa + 1)$ for $0 \leq i \leq n$. □

Lemma 4.2.3 Two priced regions R, R' are cost-equivalent w. r. t. $\kappa \in \mathbb{N}$ if and only if $pnorm_\kappa(R) = pnorm_\kappa(R')$. **Proof:** by definition of *pnorm* and cost-equivalence. □

From the operations on priced regions in definitions 2.2.6, 2.2.7, 2.2.9, and 4.1.3, the following lemma can be concluded. By this lemma the number of canonical cost-normalized priced zones will be finite.

Lemma 4.2.4 For any priced regions R, R' that are cost-equivalent w. r. t. κ:

- $pnorm_\kappa(delay(R, q)) = pnorm_\kappa(delay(R', q))$ for all $q \in \mathbb{N}$
- $pnorm_\kappa(reset(r, R)) = pnorm_\kappa(reset(r, R'))$ for all $r \subseteq \mathbb{X}$
- $pnorm_\kappa(inc(R, q)) = pnorm_\kappa(inc(R', q))$ for all $q \in \mathbb{N}$ [1]
- $pnorm_\kappa(self(R, q)) = pnorm_\kappa(self(R', q))$ for all $q \in \mathbb{N}$
- $pnorm_\kappa(translate(R, f)) = pnorm_\kappa(translate(R', f))$ for all $f : \mathbb{X} \to \mathbb{Z}$

□

Proof Assume given cost-equivalent priced regions $A = (h, [r_0 \dots r_n], [a_0 \dots a_n])$ and $B = (h, [r_0 \dots r_n], [b_0 \dots b_n])$. Let A', B' be the result of applying one of the operators $delay, reset, inc, self$ for some $q \in \mathbb{N}, r \subseteq \mathbb{X}$, or $f : \mathbb{X} \to \mathbb{Z}$, on both A, B respectively. Let $A' = (h', [r'_0 \dots r'_m], [a'_0 \dots a'_m])$ and $B' = (h', [r'_0 \dots r'_m], [b'_0 \dots b'_m])$. From the definitions of the operators, one of following properties must hold for a'_j and b'_j

[1] For completeness operation *inc* is included, although it is not needed in the symbolic semantics, as WPTA do not allow discrete cost changes.

- $0 \leq j \leq m, a'_j = a_i$, and $b'_j = b_i$, for some $0 \leq i \leq n$, or
- $j = m, a'_m = a_0 + q$, and $b'_m = b_0 + q$, or
- $0 < j \leq m, a'_j = \min(a_{n-j}, a_{n-j+1})$, and $b'_j = \min(b_{n-j}, b_{n-j+1})$, or
- $0 \leq j \leq m, a'_j = a_j + q$, and $b'_j = b_j + q$, or
- $j = m, a'_m = \min(a_n, a_0 + q)$, and $b'_m = \min(b_n, b_0 + q)$.

We know that for all $0 \leq k \leq n$ we have that $a_k = b_k$ or $a_k > \kappa \wedge b_k > \kappa$. Now inspection of all the cases mentioned above shows us that for all $0 \leq j \leq m$ we have $a'_j = b'_j$ or $a'_j > \kappa \wedge b'_j > \kappa$, and thus the lemma holds. □

Cost-equivalence and $pnorm_\kappa$ are lifted to symbolic states by requiring the same location and cost-equivalence on the priced regions. Lemma 4.2.4 ensures that the semantics of Definition 4.1.4 on two cost-equivalent symbolic states maintains cost-equivalence. If some symbolic state σ has a state that reaches the target location within the cost bound, all symbolic states that are cost-equivalent to σ also have such a state (not necessarily the same state).

Algorithm 2: *FwdBoundedReach*$(WPTA, \kappa)$

(1) **Input:** $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$, cost bound $\kappa \in \mathbb{N}_0$
Output: the LTS: $(Visited, D)$
$Visited := \emptyset$ //set of generated symbolic states
(2) $Waiting := \{(l_0, (\bar{0}, [\mathbb{X}], [0]))\}$ //set of symbolic states waiting to be explored
(3) $D := \emptyset$ //set of edges
(4) **repeat**
(5) **get** and **remove** σ from $Waiting$
(6) $Visited := Visited \cup \{\sigma\}$
(7) **foreach** $\sigma \xrightarrow{a} \tau$, with $a = \epsilon, a = (l, g, r, l')$ or $a = \delta$ //as in Definition 4.1.4
(8) **if** $pnorm_\kappa(\tau) \notin Visited$
(9) $Waiting := Waiting \cup \{pnorm_\kappa(\tau)\}$
(10) $D := D \cup \{\sigma \xrightarrow{a} pnorm_\kappa(\tau)\}$
(11) **until** $Waiting = \emptyset$
(12) **return** $(Visited, D)$

After algorithm *FwdBoundedReach* has finished, the next step consists of constructing the priced probabilistic region graph (PPRG) from the LTS: $(Visited, D)$. PPRG is a probabilistic system $(Visited, Steps)$, where $Steps$ contains all pairs $(\sigma, \mu) \in Visited \times \text{Dist}(Visited)$ such that one of the following holds:

- there is a transition $(\sigma \xrightarrow{\epsilon} \tau) \in D$ such that $\mu = \{\tau \mapsto 1\}$, or
- there is a transition $(\sigma \xrightarrow{\delta} \tau) \in D$ such that $\mu = \{\tau \mapsto 1\}$, or
- for all $\tau \in Visited$

$$\mu(\tau) = \sum_{(\sigma \xrightarrow{l,g,p,r,l'} \tau) \in D} p(r, l')$$

For target set of locations $T \subseteq L$, the cost-bounded maximal probabilistic reachability can be computed from the PPRG. It is expressed by $\mathit{MaxProbReach}(\sigma_0, \psi)$, where $\sigma_0 = (l_0, (\overline{0}, [\mathbb{X}], [0]))$ the starting symbolic state, and $\psi = \{\sigma \in \mathit{Visited} \mid \mathit{loc}(\sigma) \in T \wedge \mathit{mincost}(\text{region of } \sigma) \leq \kappa\}$

4.3 Counter-example

Consider the WPTA and its PPRG of figure 4.1, with the cost-bounded maximal probabilistic reachability problem $(T, \sqsupseteq, \lambda, \kappa) = (\{l_4\}, \geq, 0.8, 2)$. Only paths leading to l_4 within the cost bound are included. The maximal probabilistic reachability of the PPRG is 1, and the answer to the problem is "yes". Clearly this is not correct. The error comes from the fact that in PPRG for the left path in location l_0, arbitrarily close to 1 units of time are spent, while for the right path in l_0, arbitrarily close to 0 units of time are spent. But an adversary must choose in advance how much time exactly is spent in location l_0. The problem is that we abstract away from the exact amount of time that is spent. The *self* operator in ϵ-transitions uses this assumption to let time progress in the cheapest way.

4.4 Problem Analysis

In the previous section we showed a counter-example of our approach, and we gave an analysis of the problem. Here we formalize that analysis, and we show that *FwdBoundedReach* computes an upper bound on the cost-bounded maximal probabilistic reachability. First, some definitions and methods for probability on finite paths are presented. Then the use of time-abstracting bisimulations on the verification of probabilistic timed automata (PTA) is discussed. Finally the exact problem is given, and we prove that *FwdBoundedReach* computes an upper bound on the cost-bounded maximal reachability.

4.4.1 Reachability probability on paths of finite length

Definition 4.4.1 Let $PS = (S, \mathit{Steps})$ be a probabilistic system and $F \subseteq S$ a set of states. For any adversary $A \in \mathit{Adv}_{PS}$ and finite path $\omega \in \mathit{Path}^A_{\mathsf{fin}}$, let:

$$P^A_0(\omega \rightsquigarrow F) = \begin{cases} 1 & \text{if } \mathrm{last}(\omega) \in F \\ 0 & \text{otherwise} \end{cases}$$

and for any $n \in \mathbb{N}$

$$P^A_{n+1}(\omega \rightsquigarrow F) = \begin{cases} 1 & \text{if } \mathrm{last}(\omega) \in F \\ \sum_{s \in S} \mu(s) \cdot P^A_n(\omega \xrightarrow{\mu} s \rightsquigarrow F) & \text{otherwise} \end{cases}$$

where $\mu = A(\omega)$. □

Define:

$$P_n^{\max} \stackrel{\text{def}}{=} \sup_{A \in Adv_{PS}} P_n^A$$

Computation of $P_n^{\max}(s \rightsquigarrow F)$ can be formulated as a dynamic programming problem [dA99], which can be solved by value iteration (see [Tij03]). Assume given $PS = (S, Steps)$, with $k = |S|$. $\lambda = [\lambda_s]_{s \in S \setminus F} \in \mathbb{R}^k$ denote a vector of real numbers indexed by the states of PS without the target states. Define the Bellman operator $L : \mathbb{R}^k \mapsto \mathbb{R}^k$ on the space of λ by

$$[L(\lambda)]_s = \min_{(s,\mu) \in Steps} \left[\sum_{t \in S \setminus F} \mu(t)\lambda_t + \sum_{t \in F} \mu(t) \cdot -1 \right]$$

where $[L(\lambda)]_s$ denotes the s-component of vector $L(\lambda)$. Given initial vector λ^0 that has value zero for all elements, the value iteration method computes the sequence of vectors $\lambda^0, \lambda^1, \ldots, \lambda^n$ by $\lambda^i = L(\lambda^{i-1})$, for $1 \leq i \leq n$. The solution is $P_n^{\max}(s \rightsquigarrow F) = -\lambda_s^n$.

The following lemma shows us how maximal probabilistic reachability can be computed.

Lemma 4.4.2 [KNS03, KNS01] For any probabilistic system $PS = (S, Steps)$, adversary $A \in Adv_{PS}$, state $s \in S$ and target $F \subseteq S$: $[P_n^A(s \rightsquigarrow F)]_{n \in \mathbb{N}}$ is an ascending (or non-decreasing) sequence in $[0, 1]$ which converges to $ProbReach^A(s, F)$. □

4.4.2 Time-Abstracting Bisimulations

The following results are useful, when working with time-abstracting bisimulations (TaBs).

Definition 4.4.3 (Time transitions traversing classes) [TY01]. Now we prove an important property of TaBs related to the passage of time. Consider TTS and a TaB $\simeq$ on TTS. Given a time transition of $TTS, s \xrightarrow{d} t$, and m different classes $\sigma_1, \ldots, \sigma_m$, we say that the transition traverses $\sigma_1, \ldots, \sigma_m$ if:

1. $s \in \sigma_1$ and $t \in \sigma_m$.
2. For all $0 < d' < d$, there exists $1 \leq i \leq m$ such that $s + d' \in \sigma_i$.

□

Lemma 4.4.4 [TY01].

1. Any time transition traverses a unique (finite) set of classes.
2. If $s \simeq s'$ then for any time transition $s \xrightarrow{d} t$, there exists a time transition $s' \xrightarrow{d'} t'$ such that $t \simeq t'$ and the two transitions traverse the same classes.

□

For a path ω in $TTS = (S, Act, \rightarrow)$, let word($\omega$) be the sequence of labels of discrete transitions from Act in ω, in the same order. word is defined on symbolic paths of $TTS/_{\simeq}$ in the same way.

Lemma 4.4.5 (Symbolic Correspondence) Every path ω in TTS has a unique path denoted $[\omega]$ in $TTS/_{\simeq}$ s. t. word(ω) = word($[\omega]$). Inversely, if $\Omega = \sigma_1 \rightarrow/_{\simeq} \sigma_2 \rightarrow/_{\simeq} \cdots$ is a path in $TTS/_{\simeq}$ then for all $s_1 \in \sigma_1$ there exists a path ω starting from s_1 with $[\omega] = \Omega$. □

Proof (see also proof of proposition 25 in [KNSS02]). Given finite or infinite path $\omega = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n \rightarrow s_{n+1} \rightarrow$ in TTS. For all transitions $s_i \rightarrow s_{i+1}$, with $i \geq 0$, we can find a symbolic path Ω_i such that $s_i \in \Omega_i(0)$ and $s_{i+1} \in \text{last}(\Omega)$ in one of the following ways:

- if $s_i \rightarrow s_{i+1}$ is a time transition, then by Lemma 4.4.4 we can find unique symbolic path Ω_i, possibly of length zero,
- if $s_i \rightarrow s_{i+1}$ is a discrete transition, then by time-abstracting bisimulations we can find unique symbolic path $\Omega_i = \sigma_i \rightarrow \sigma_{i+1}$,

We can write concatenation $[\omega] = \Omega_0 \cdot \Omega_1 \cdots \Omega_n \cdots$. We know that $\forall i \geq 0. \text{last}(\Omega_i) = \Omega_{i+1}(0)$, and by leaving out symbolic paths of length zero, we get the combined symbolic path.

Inversely, by definition of TaB, for every symbolic path $\sigma \rightarrow \tau$ and every $s \in \sigma$, we can find $t \in \tau$ such that $s \rightarrow t$ is a transition in TTS. Constructing path ω for some symbolic path Ω in this way, it is easy to see that $[\omega] = \Omega$. □

4.4.3 Probabilistic Region Graph

With a probabilistic region graph it is possible to construct a quotient system that preserves probabilistic reachability in a PTA.

Definition 4.4.6 (Probabilistic region graph) [KNSS02]. Given PTA, let TTS give the semantics of PTA_{TA} (section 2.3.3), and region equivalence $\simeq_k$, where $k : \mathbb{X} \rightarrow \mathbb{N}$ maps each clock to its ceiling. For brevity we omit k. The region graph of PTA_{TA} is $TTS/_{\simeq} = (S/_{\simeq}, Act \cup \{\epsilon\}, \rightarrow/_{\simeq})$, with $Act = E_{PTA}$. The probabilistic region graph is a probabilistic system (Σ, Steps), where $\Sigma = S/_{\simeq}$ and Steps contains all pairs $(\sigma, \mu) \in \mathit{Visited} \times \text{Dist}(\mathit{Visited})$ such that one of the following holds:

- There is a transition $(\sigma \xrightarrow{\epsilon}/_{\simeq} \tau)$ such that $\mu = \{\tau \mapsto 1\}$, or
- $\mu = \{\tau \mapsto 1\}$ if $\sigma \xrightarrow{\epsilon}/_{\simeq} \tau$ exists,
- for all $\tau \in S/_{\simeq}$

$$\mu(\tau) = \sum_{\sigma \xrightarrow{l,g,p,r,l'}/_{\simeq} \tau} p(r, l').$$

□

The main Theorem of [KNSS02] states that model checking the logic PTCTL on some PTA can be reduced to model checking the probabilistic region graph, which is finite. Next to that we have the following lemma on probabilistic region graphs.

Lemma 4.4.7 Given *PTA* with semantics $TPS = (S, Steps)$. Let $Q = (S/_{\simeq}, Steps)$ be the probabilistic region graph. For every $A \in Adv_{TPS}$, and $G \subseteq \{\omega \mid Path^A_{\mathsf{full}}\}$ measurable, there exists unique $[A] \in Adv_Q$ such that:

$$Prob^{[A]}\{[\omega] \mid \omega \in G\} = Prob^A(G)$$

Proof: using part 2. of the proof of proposition 25 in [KNSS02] □

4.4.4 The problem & an upper bound

By using [Spr00], the TPS semantics of a WPTA can be formulated as a *concurrent probabilistic process*. Cost-bounded maximal probabilistic reachability can be expressed in the logic PBTL, together with an atomic proposition in the concurrent probabilistic process, that distinguishes target states, i. e. states having the correct target location and cost at most the cost bound. Now [Spr00] states (after Theorem 1), that if the concurrent probabilistic process has a finite *probabilistic bisimulation*, then model checking of all PBTL properties (including ours) is decidable. A probabilistic bisimulation respects the partitioning by atomic propositions. If we can find a finite time-abstracting bisimulation that respects the partitioning, then this TaB is a special case of finite probabilistic bisimulations, and cost-bounded maximal probabilistic reachability becomes decidable.

The problem is that priced regions are not the equivalence classes of some finite TaB. In the first place, because priced regions do not model a set of states, but rather a set of clock valuations with a price function. Even if we would regard a priced region as an equivalence class, with the clock valuation of a state in the unpriced region, and the cost equal to the outcome of the price function, then two states in this definition are not bisimilar. This can be seen from the WPTA of figure 4.2 on page 70, where state $(l_0, \frac{1}{3}, \frac{2}{3})$ and $(l_0, \frac{1}{2}, 1)$ are both in the same priced region of the PPRG, but only the latter will reach l_4 by the left path (for more information on the figure, read the next section). In chapter 5 we will see that in general, a WPTA may not have any finite TaB that respects target states.

However, if finite bisimulations do not exist, we cannot conclude that exact maximal reachability is not computable [BBR04]. Therefore it remains an open question whether there is a terminating algorithm that returns the exact maximal probabilistic reachability.

Although priced regions are not equivalence classes of some TaB, we may take the following approach. Assume given *WPTA* and its semantics $TPS = (S, Steps)$. Let T be the target set of locations, and κ the cost bound. Define $F = \{(l, v, c) \mid l \in T \wedge v \geq \mathbb{R}_+ \wedge 0 \leq c \leq \kappa\}$, the set of target states. Let $\sigma_0 = (l_0, (\bar{0}, [\mathbb{X}], [0]))$ is the symbolic state representing the starting state. Let $\psi = \{(l, R) \mid l \in T \wedge mincost(R) \leq \kappa\}$ is the set of symbolic states that reach a target location within cost bound κ. We would like the following two properties to hold:

1. For every $B \in Adv_{PPRG}$ there exists $A \in Adv_{TPS}$ such that $ProbReach^A(s_0, F) \geq ProbReach^B(\sigma_0, \psi)$.

2. For every $A \in Adv_{TPS}$ there exists $B \in Adv_{PPRG}$ such that $ProbReach^B(\sigma_0, \psi) \geq ProbReach^A(s_0, F)$.

Unfortunately, property (1) is **incorrect** as shown by figure 4.1; see section 4.3 for details. This means that the PPRG is not a sound representation of the TPS. Still, we will prove property (2) below, so we can use the PPRG to compute an upper bound of the reach probability.

Proof of property (2)

For some state (l, v, c) in a WPTA or LPTA, let $unprice((l, v, c)) = (l, v)$ denote its unpriced version. Also for a symbolic state with a priced region $\sigma = (l, (h, [r_0, \ldots, r_n], [c_0, \ldots, c_n]))$ let $unprice(\sigma) = (l, (h, [r_0, \ldots, r_n]))$ denote its unpriced version. *unprice* is extended to (symbolic) paths in the following way: given $\omega = s_0 \rightarrow s_1 \rightarrow \cdots$ then $unprice(\omega) = unprice(s_0) \rightarrow unprice(s_1) \rightarrow \cdots$.

We need the following lemma which is stronger than Lemma 2.2.11, because a correspondence between paths is required.

Lemma 4.4.8 For every finite path ω in a LPTA, with $\text{last}(\omega) = (l, v, c)$, there is a unique finite symbolic path Ω using the symbolic semantics of Definition 4.1.4, with $unprice(\Omega) = [unprice(\omega)]$ (as defined by Lemma 4.4.5) and $\text{last}(\Omega) = (l, R)$, with $v \in R$ and $cost(v, R) \leq c$ (2.2.5).
Proof: by observing the proof of Lemma 1 in [BFH+01]. □

Assume given some $A \in Adv_{TPS}$. Let

$$Reach = \{\omega \in Path^A_{\mathsf{full}} \wedge \omega(0) = s_0 \wedge (\exists i \in \mathbb{N}.\omega(i) = (l, \cdot, c) \wedge l \in T \wedge c \leq \kappa)\}$$

denote the set of all infinite paths generated by A that respect the problem conditions. By ignoring cost, with region equivalence we can construct for *WPTA* a time-abstracting probabilistic quotient Q. By Lemma 4.4.7, it follows that a corresponding adversary $B' \in Adv_Q$ can be constructed such that

$$Prob^A(Reach) = Prob^{B'}\{[unprice(\omega)] \mid \omega \in Reach\} \tag{4.1}$$

From B' an adversary $B \in Adv_{PPRG}$ can be constructed in the following way: for all paths Ω in *PPRG*,

$$B(\Omega) = \mu \text{ iff } \forall \sigma \in \text{support}(\mu).\mu(\sigma) = \mu'(unprice(\sigma)) \text{, with } \mu' = B'(unprice(\Omega)).$$

For B the following holds:

$$Prob^B\{\Omega \mid unprice(\Omega) = [unprice(\omega)] \wedge \omega \in Reach\} = Prob^{B'}\{[unprice(\omega)] \mid \omega \in Reach\} \tag{4.2}$$

By Lemma 4.4.8:

$$\{\Omega \in Path^B_{\mathsf{full}} \mid unprice(\Omega) = [unprice(\omega)] \wedge \omega \in Reach\} \subseteq$$
$$\{\Omega \in Path^B_{\mathsf{full}} \mid \Omega(0) = \sigma_0 \wedge (\exists i \in \mathbb{N}.(l, R) = \Omega(i) \wedge mincost(R) \leq \kappa \wedge l \in T)\} \tag{4.3}$$

With σ_0, ψ as in the property, combining everything:

$$\begin{aligned}
&ProbReach^A(s_0, F)\\
&= Prob^A(Reach)\\
&= Prob^{B'}\{[unprice(\omega)] \mid \omega \in Reach\} \text{ by (4.1)}\\
&= Prob^B\{\Omega \mid unprice(\Omega) = [unprice(\omega)] \wedge \omega \in Reach\} \text{ by (4.2)}\\
&\leq Prob^B\{\Omega \in Path^B_{\mathsf{full}} \mid \Omega(0) = \sigma_0 \wedge (\exists i \in \mathbb{N}.(l, R) = \Omega(i) \wedge\\
&\quad mincost(R) \leq \kappa \wedge l \in T)\} \text{ by (4.3)}\\
&= ProbReach(\sigma_0, \psi)
\end{aligned}$$

4.5 A Lower bound

We claim that it is possible to calculate a lower bound on the maximal probabilistic reachability using a priced probabilistic region graph (PPRG). In this section, we describe an idea to find a lower bound and what needs to be changed in the algorithm. The difference with calculating the upper bound is that the symbolic semantics of Definition 4.1.4 is changed such that the *self* operator is only used in a special symbolic transition. The symbolic transitions are now:

- $(l, R) \xrightarrow{\epsilon} (l, delay(R, \dot{\$}(l)))$ if $delay(R, \dot{\$}(l)) \in inv(l)$,
- $(l, R) \xrightarrow{(l,g,r,l')} (l', R')$ if there exists g, r such that $(l \xrightarrow{g,r} l') \in E, R \subseteq g, R' = inc(reset(r, R), \$(l, g, r, l'))$.
- $(l, R) \xrightarrow{\delta} (l, R')$ if there exists $x \in \mathbb{X}$ such that $R \wedge (k(x) + 2) \neq \emptyset$ and $R' = translate(R, f)$, where $\forall y \in \mathbb{X}.f(y) = \begin{cases} -1 & \text{if } y = x \\ 0 & \text{otherwise} \end{cases}$.
- $(l, R) \xrightarrow{self} (l, self(R, \dot{\$}(l)))$

Further, in a path of priced regions that are all self-delayable, i. e. $r_0 = \emptyset$ in the definition of priced region, one and only one self-transition must occur. In that way the algorithm has to choose where to spend time. If it has chosen to spend it in some symbolic state, all successor symbolic states, even in different branches, can't spend it any more. In this way the problem of figure 4.1 is solved, but figure 4.2 shows a WPTA with its PPRG constructed in the new

way. The maximal probabilistic reachability is 0.7. But the optimal adversary should spend $\frac{1}{2}$ time units in location l_0. Thus only a lower bound can be computed.

Figure 4.1: Counter example: WPTA and its priced region graph.

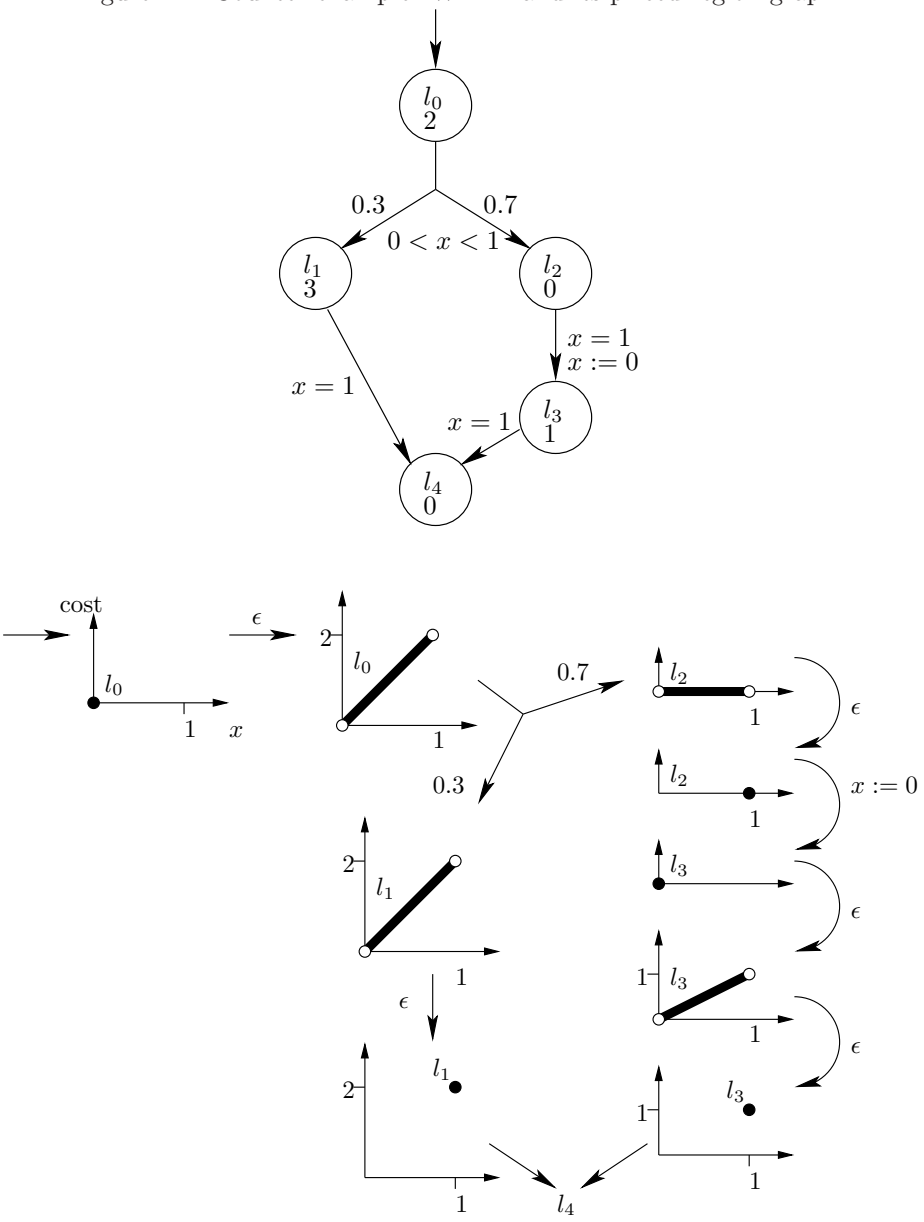
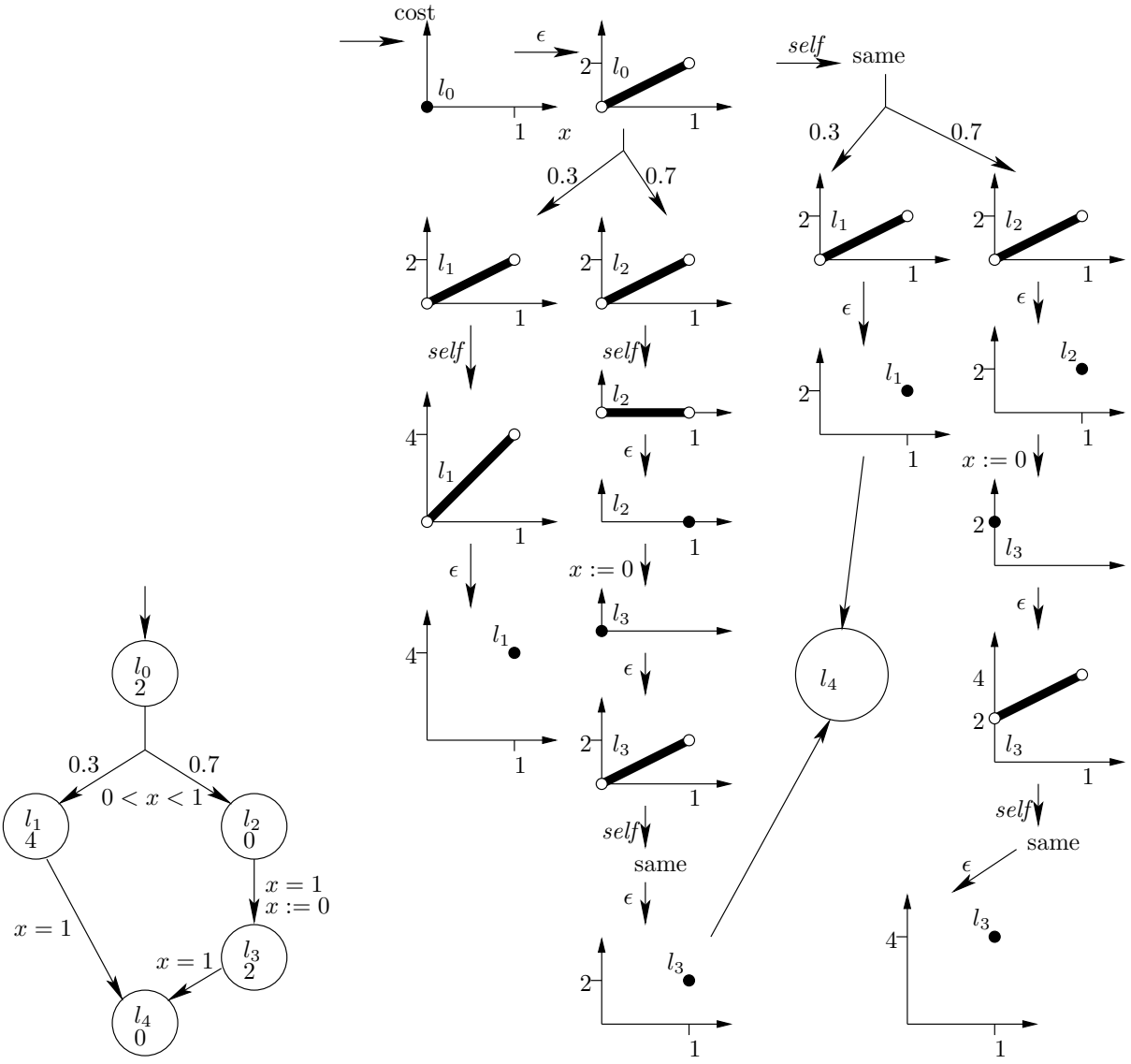


Figure 4.2: Example on how to calculate a lower bound. A transition to 'same' means nothing changes.



Chapter 5

Cost-Bounded Maximal Probabilistic Reachability

In section 5.1, an algorithm is given that does not always terminate, and will output an ascending (or non-decreasing) sequence of values that converges to the solution of cost-bounded maximal probabilistic reachability in closed WPTA. In the rest of this chapter we assume every WPTA to be closed. Also a number of properties of WPTA are discussed for which the algorithm *will* terminate. Section 5.2 gives some ideas on algorithms that generate a sequence of values that is descending (or non-increasing) and converges to the solution. This opens the way for a solution to the cost-bounded maximal probabilistic reachability problem that works in many cases.

5.1 Ascending Converging Value

The algorithm presented here is based on the algorithm in [KNS03] that computes maximal probabilistic reachability for probabilistic timed automata (PTA). We incorporate prices by using symbolic states that consist of a location and a multi-priced zone. Moreover, the algorithm differs in that it computes a sequence of values, where the n-th value is the maximal probabilistic reachability using at most n steps. Theorem 5.1.2 states the correctness of the computed probabilities. The proof is based on the proof in [KNS03]; it follows the same lines. The main difference is in one of the intermediate properties. [KNS03] states that for every abstract adversary that uses n steps, there is a normal adversary that has at least the same probability and uses $2n$ steps. We require the stronger statement that there is a normal adversary with also n steps.

The work of [KNS03] is based on that of [KNS01], where an algorithm is presented that calculates maximal reachability for *symbolic probabilistic systems*. Symbolic probabilistic systems are a very general framework, and it is possible to formulate a WPTA as a symbolic probabilistic system. With this different approach no new algorithm needs to be devised, although an implementation for symbolic states and its operations needs to be given. Now we can base our

proof on the one in [KNS01]. This is easier because here the number of steps in the abstract adversary and the corresponding normal adversary are the same. The problem is that the predecessor operation is split up in timed predecessor (*tpre*) and discrete predecessor (*dpre*). This generates unnecessary symbolic states, namely those between *tpre* and *dpre*.

Corollary 5.1.3 shows the usefulness of the algorithm, by stating that the sequence of probability values generated by the algorithm, converges to the actual maximal probability, although this value may never be quite reached.

Definition 5.1.1 [KNSW04]. A **sub-probabilistic system** is a more general version of the probabilistic system of Definition 2.3.3. The distributions can now be sub-distributions. A SPS takes the form (S, Steps), where S is a set of states and $\mathit{Steps} \subseteq S \times \mathrm{SubDist}(S)$ is a probabilistic transition relation. □

Every SPS can easily be converted to a PS by adding a state that has no outgoing edges, where all the 'missing' probabilities of the sub-distributions lead to.

5.1.1 The algorithm

Algorithm *CBMaxReachAlg* on page 73 is the algorithm that gives an ascending (or non-decreasing) sequence of values that converge to the solution of cost-bounded maximal probabilistic reachability, for some WPTA. Parameter s is the state for which the reachability probability is computed. ϕ is the set of target states. ϕ must be representable as a finite set of symbolic states, and must be closed under time predecessor, i. e. $\mathit{tpre}(\phi) = \phi$. From now on we assume that the symbolic states representing ϕ have the same maximum cost for all clock valuations. Formally, for all $\sigma \subseteq \phi, \exists \kappa \in \mathbb{N}. \forall (l, v, c) \in \sigma. c \leq \kappa$. The symbolic states consist of a location and a multi-priced zone. Execution of *CBMaxReachAlg* grows a sequence $[R_n]_{n=1..\infty}$ of output values that are ascending and converge to the cost-bounded maximal probabilistic reachability. Theorem 5.1.2 states the correctness of *CBMaxReachAlg*, and Corollary 5.1.3 its converging behaviour.

CBMaxReachAlg is derived from [KNS03, KNS01]. The key observation is that to preserve the probabilistic branching, one must consider the intersections of symbolic states generated by edges from the same distribution. The intuition is that by computing intersections, we get representations for states that have multiple edges from a probabilistic edge leading to the target, thereby enlarging the reach probability.

CBMaxReachAlg performs a breadth-first backward exploration of the symbolic state space, where in each iteration of the algorithm, the depth of exploration increases. The algorithm starts with the set ϕ. From this set successively predecessors are computed in a breadth-first manner. During the exploration a graph is constructed. The edges of the graph are contained in the sets $E_{(l,g,p)}$. The edges exist between symbolic states, where an edge going to some symbolic state, starts in its predecessor, or a subset of its predecessor. Formally, $(\sigma, r, l', \tau) \in E_{(l,g,p)}$, if $\sigma \subseteq \mathit{pre}_{(l,g,p,r,l')}(\tau)$. The sets $E_{(l,g,p)}$ are initialized as empty sets. The set *Visited* contains the generated symbolic states, it is a shorthand notation, because it can be completely derived from the sets $E_{(l,g,p)}$ (line

Algorithm 3: *CBMaxReachAlg*(s, ϕ) derived from [KNS03, KNS01]

(1) **Input:** $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$,
(2) starting state s,
(3) target set of symbolic states ϕ closed under $tpre$
Output: R_n in iteration n
foreach $(l, g, p) \in pE$ //Initialize all edge sets $E_{(l,g,p)}$
(4) $E_{(l,g,p)} := \emptyset$
(5) $Waiting_1 := \phi$ // $Waiting_n$ is the set of symbolic states waiting to be explored in iteration n.
(6) **short-hand** $Visited = \{\tau \mid \exists (l, g, p) \in pE.(\tau, \cdot, \cdot) \in E_{(l,g,p)}\} \cup \{\phi\}$
(7) **if** $\exists \sigma \in Visited.s \in tpre(\sigma)$ **then** $R_0 = 1$
(8) **else** $R_0 = 0$
(9) **for** $n = 1$ **to** ∞
(10) $Waiting_{n+1} = \emptyset$
(11) **foreach** $\tau \in Waiting_n$, $e = (l, g, p, r, l') \in E_{WPTA}$, with $l' = loc(\tau)$
(12) $\sigma = pre_e(\tau)$
(13) **if** $\sigma \neq \emptyset$ //if at least one state with positive cost
(14) **if** $\sigma \notin Visited$ //if not already visited
(15) $Waiting_{n+1} := Waiting_{n+1} \cup \{\sigma\}$
(16) $E_{(l,g,p)} := E_{(l,g,p)} \cup \{(\sigma, r, l', \tau)\}$
(17) **foreach** $(\sigma', \bar{r}, \bar{l}', \tau') \in E_{(l,g,p)}$
(18) **if** $\sigma \cap \sigma' \neq \emptyset \wedge (r, l') \neq (\bar{r}, \bar{l}')$
(19) $Waiting_{n+1} := Waiting_{n+1} \cup \{\sigma \cap \sigma'\}$
(20) $E_{(l,g,p)} := E_{(l,g,p)} \cup \{(\sigma \cap \sigma', r, l', \tau), (\sigma \cap \sigma', \bar{r}, \bar{l}', \tau')\}$
(21) $SPS_n := (Visited, Steps)$, where $(\sigma, \pi) \in Steps$ **if and only if there exists** $E_\pi \subseteq E_{(l,g,p)}$ **for some** $(l, g, p) \in pE$ **such that**
- $\forall (\sigma', \cdot, \cdot) \in E_\pi.\sigma' = \sigma$
- $\forall (\sigma, r, l', \tau), (\sigma, \bar{r}, \bar{l}', \tau') \in E_\pi.\tau \neq \tau' \Rightarrow (r, l') \neq (\bar{r}, \bar{l}')$
- $\forall \tau \in Visited.\pi(\tau) = \sum\{p(r, l') \mid (\cdot, r, l', \tau) \in E_\pi\}$

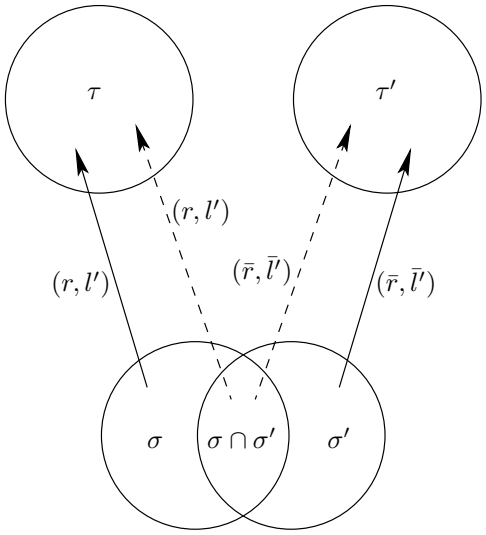
(22) $R_n = \max_{\substack{\sigma \in Visited \\ s \in tpre(\sigma)}} P_n^{\max}(\sigma \underset{SPS_n}{\rightsquigarrow} \phi)$
(23) **if** $Waiting_{n+1} = \emptyset$ **then stop**

6). Line 7 gives the result in case of iteration zero.

On line 16 an edge is added between a symbolic state and its predecessor. Lines 17–20 add intersections between the predecessor and previously generated symbolic states. These intersections are only generated between symbolic states that can reach the target using the same probabilistic edge (l, g, p), but another edge from the distribution (condition $(\bar{r}, \bar{l}') \neq (r, l')$ on line 18). Only these intersections are of interest for probabilistic branching. On line 20 the intersection symbolic state gets the two outgoing edges, from which it was originated. Figure 5.1 gives a graphical representation, where the normal arrows are the ones from which the intersection is originated. The dashed arrows are the new ones for the intersection symbolic state.

The intersection between for example three symbolic states is computed by first intersecting two symbolic states and then intersecting the result with the third one. More generally the intersection of a set of symbolic states is computed by

Figure 5.1: Graphical representation of execution of lines 17–20 of algorithm *CBMaxReachAlg*.



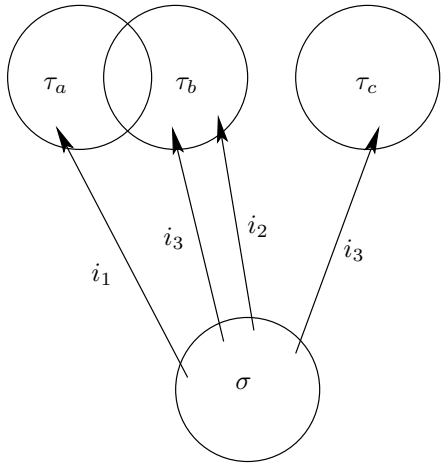
applying pairwise intersection until only one symbolic state remains, which is the intersection of all symbolic states of the original set. The order in which symbolic states are intersected does not make a difference as can be seen from lines 19 and 20.

On line 21, a sub-probabilistic system SPS_n is constructed from the edge sets $E_{(l,g,p)}$. The state space of SPS_n is *Visited*. For each symbolic state in *Visited*, probabilistic edges are constructed, by taking together as much edges from one edge set $E_{(l,g,p)}$ as possible. This set of edges is denoted as E_π. The first two bullets after line 21 are the conditions on edges in E_π. Condition 1 is obvious. Condition 2 ensures that different edges are used in the construction of the probabilistic edge. Note that these conditions imply that for all $(\sigma, r, l', \tau), (\sigma, (\bar{r}, \bar{l}'), \tau') \in E_\pi$ we have $(r, l') \neq (\bar{r}, \bar{l}')$. From E_π a probabilistic edge with distribution π is constructed at the third bullet. Note that each E_π that can be constructed in this way results in a probabilistic edge in SPS_n. Figure 5.2 gives an example of the construction of two probabilistic edges (π_1 and π_2).

To compute $R_n = \max_{\substack{\sigma \in Visited \\ s \in tpre(\sigma)}} P_n^{\max}(\sigma \underset{SPS_n}{\rightsquigarrow} \phi)$ on line 22 we can use value iteration, see section 4.4.1. With value iteration a maximum reachability probability for all symbolic states of SPS_n can be computed. Note that *SPS* is a *sub*-probabilistic system. Adding an absorbing state to convert it to a PS is not needed because there are no outgoing arrows from this new state; it plays no positive role in the reachability probability. Now the maximum probability of the symbolic states that have s in their time predecessor gives the result.

Value iteration computes probability values on states. The set of symbolic states is extended in each iteration of the algorithm. When new states are given

Figure 5.2: Example of the construction of the SPS on line 21 of algorithm *CBMaxReachAlg*.



$E_{(l,g,p)} \supseteq E_{\pi_1} = \{(\sigma, i_1, \tau_a), (\sigma, i_2, \tau_b), (\sigma, i_3, \tau_c)\}$
$\pi_1(\tau_a) = p(i_1)$
$\pi_1(\tau_b) = p(i_2)$
$\pi_1(\tau_c) = p(i_3)$

$E_{(l,g,p)} \supseteq E_{\pi_2} = \{(\sigma, i_1, \tau_a), (\sigma, i_2, \tau_b), (\sigma, i_3, \tau_b)\}$
$\pi_2(\tau_a) = p(i_1)$
$\pi_2(\tau_b) = p(i_2) + p(i_3)$
$\pi_2(\tau_c) = 0$

a maximal probability zero, value iteration can be continued, without having to start over again.

Possibly *CBMaxReachAlg* terminates on line 23 when no new symbolic states are explored. Value iteration can be continued, but another possibility is to solve the problem by reducing it to a linear programming problem [dA99]. When known in advance that the algorithm terminates the value iteration steps can be skipped. Linear programming can give an exact answer, while value iteration may only converge to an answer. Convergence is appropriate when the algorithm does not terminate in the first place. Which approach is best when no exact answer is needed, will probably depend on the concrete WPTA, and is subject for further research.

Examples

Figures 5.3 and 5.4 contain two examples of WPTA and a graphical representation of the symbolic computation of *CBMaxReachAlg*. Figure 5.3 uses a closed version of the WPTA of figure 4.1. Operation *pre* is split up in *tpre* and *dpre*, but intersections can only be added when *pre* is fully completed. The plain arrows are applications of *tpre* or *dpre*. The dashed arrows denote the intersection that is made. The last *tpre* is made to see which states are in the timed predecessor, as required. The target symbolic state has cost bound 2. The starting state has all clocks and cost zero. Only the left path can reach the target, so the reach probability is 0.3. Figure 5.4 is the counter-part. It uses a closed version of the WPTA of figure 4.2. Now the target symbolic state has cost bound 3, and the reach probability is 1. The generated SPS is in both cases clear from the figure.

In section 5.1.3 we will see a more complicated example.

5.1.2 Correctness of *CBMaxReachAlg*

Length-dependent adversaries are a special class of adversaries that only take the last state and the length of a finite path into account. Let A be a length-dependent adversary, then for any two paths ω and ω' in probabilistic system $PS = (S, Steps)$, with $|\omega| = |\omega'|$ and $\text{last}(\omega) = \text{last}(\omega')$, we have that $A(\omega) = A(\omega')$. The length-dependent adversary A can also be written as a function $A : \mathbb{N} \times S \to \text{Dist}(S)$. The natural number represents the path length. From now on Adv_{PS} denotes the set of length-dependent adversaries on probabilistic system PS.

Theorem 5.1.2 Given $WPTA = (L, l_0, \mathbb{X}, inv, pE, c, \dot{\$})$ and its semantics as $TPS = (S, Steps)$. Let ϕ be the set of target states, that is closed under time predecessor, i. e. $tpre(\phi) = \phi$, and representable as a finite set of symbolic states. $[R_n]_{n=1..\infty}$ is the sequence of values generated by algorithm *CBMaxReachAlg*. Define in the usual way: $\max \emptyset = 0$. For any state $s \in S$ and $n \in \mathbb{N}$ then

$$P_n^{\max}(s \underset{TPS}{\rightsquigarrow} \phi) = R_n$$

using length-dependent adversaries in TPS, and in calculating R_n. □

The maximal probabilistic reachability for some probabilistic system TPS, starting state s, and target set of states F equals $\lim_{n\to\infty} P_n^{\max}(s \underset{TPS}{\rightsquigarrow} F)$. The following corollary states that the sequence $[P_n^{\max}(s \underset{TPS}{\rightsquigarrow} F)]_{n\in\mathbb{N}}$ is an ascending sequence. In each iteration of *CBMaxReachAlg* we get a higher probability that is a better approximation of maximal probabilistic reachability.

Corollary 5.1.3 Given $TPS = (S, Steps)$. For any $s \in S, n \in \mathbb{N}$ and $F \subseteq S$ we have that

$$P_{n+1}^{\max}(s \underset{TPS}{\rightsquigarrow} F) \geq P_n^{\max}(s \underset{TPS}{\rightsquigarrow} F)$$

□

Proof Let

$$P_n^{\max}(s \underset{TPS}{\rightsquigarrow} F) = P_n^{A}(s \underset{TPS}{\rightsquigarrow} F)$$

for some adversary $A \in Adv_{TPS}$. Then

$$P_n^{A}(s \underset{TPS}{\rightsquigarrow} F) \leq P_{n+1}^{A}(s \underset{TPS}{\rightsquigarrow} F) \leq P_{n+1}^{\max}(s \underset{TPS}{\rightsquigarrow} F)$$

□

Proof of Theorem 5.1.2

The proof is very similar to the proof of proposition 14 in [KNS03]. Note that we only use adversaries that are length-dependent. $SPS_n = (\Sigma_n, Steps_n)$ denotes the subprobabilistic system generated by *CBMaxReachAlg* in iteration n. The proof is split up in proving properties (a), (b), (c) and (d).

The proofs of (c) and (d) use induction on the number of iterations n. By line 7, $\Sigma_0 = \phi$, and we can assume that $Steps_0$ has a looping probabilistic transition for every symbolic state in ϕ. Formally, $\forall \sigma \in \phi . \exists (\sigma, \{\sigma \mapsto 1\}) \in Steps_0$.

(a) If $(\sigma, r, l', \tau) \in E_{(l,g,p)}$ of the algorithm, and $(l, v, c) \in \sigma$, then $v \in inv(l), v \in g$, and $(l', v[r := 0], c) \in tpre(\tau)$. **Proof:** by definition of *dpre* and *tpre*.

(b) For any $s \in S, n \in \mathbb{N}, P_n^{\max}(s \underset{TPS}{\rightsquigarrow} \phi) > 0$ if and only if for some symbolic state $\sigma \in \Sigma_n, s \in tpre(\sigma)$. **Proof:** by definition of *tpre* and *pre*. In fact this holds for all symbolic transition systems generated by taking predecessor operations.

(c) For all $n \in \mathbb{N}, k \in \mathbb{N}, B \in Adv_{SPS_k}, \sigma \in \Sigma_k$ and $s \in tpre(\sigma)$, there exists $A \in Adv_{TPS}$ such that: $P_n^A(s \underset{TPS}{\rightsquigarrow} \phi) \geq P_n^B(\sigma \underset{SPS_k}{\rightsquigarrow} \phi)$.

(d) For all $n \in \mathbb{N}, A \in Adv_{TPS}$ and $s \in S$, if $P_n^{\max}(s \underset{TPS}{\rightsquigarrow} \phi) > 0$, then there exist $\sigma \in \Sigma_n$ and $B \in Adv_{SPS_n}$ such that $s \in tpre(\sigma)$ and $P_n^B(\sigma \underset{SPS_n}{\rightsquigarrow} \phi) \geq P_n^A(s \underset{TPS}{\rightsquigarrow} \phi)$.

By using (c) for all $n \in \mathbb{N}, k \in \mathbb{N}$ and

$$b \in \{P_n^B(\sigma \underset{SPS_k}{\rightsquigarrow} \phi) \mid B \in Adv_{SPS_k} \wedge \sigma \in \Sigma_k \wedge s \in tpre(\sigma)\}$$

the following holds:

$$\sup\{P_n^A(s \underset{TPS}{\rightsquigarrow} \phi) \mid A \in Adv_{TPS}\} \geq b$$

By using (d) for all $n \in \mathbb{N}, s \in S$ and

$$a \in \{P_n^A(s \underset{TPS}{\rightsquigarrow} \phi) \mid A \in Adv_{TPS}\}$$

if $P_n^{\max}(s \underset{TPS}{\rightsquigarrow} \phi) > 0$ the following holds:

$$\sup\{P_n^B(\sigma \underset{SPS_n}{\rightsquigarrow} \phi) \mid B \in Adv_{SPS_n} \wedge \sigma \in \Sigma_n \wedge s \in tpre(\sigma)\} \geq a$$

By definition of supremum and maximum we have the following equation.

$$\begin{aligned} &\sup\{P_n^B(\sigma \underset{SPS_k}{\rightsquigarrow} \phi) \mid B \in Adv_{SPS_k} \wedge \sigma \in \Sigma_k \wedge s \in tpre(\sigma)\} \\ &= \max_{\substack{\sigma \in \Sigma_k \\ s \in tpre(\sigma)}} \left(\sup \left\{ P_n^B(\sigma \underset{SPS_k}{\rightsquigarrow} \phi) \mid B \in Adv_{SPS_k} \right\} \right) \end{aligned}$$

Combining the results, with $k = n$, (b), and the common definition that $\max \emptyset = 0$, the proof is completed.

Proof of property (c)

Consider any $k \in \mathbb{N}, B \in Adv_{SPS_k}, \sigma \in \Sigma_k$ and $s \in tpre(\sigma)$. We prove (c) by induction on n. For $n = 0$, by Definition 4.4.1, two cases have to be considered.

- If $P_0^B(\sigma \underset{SPS_k}{\rightsquigarrow} \phi) = 1$, then $\sigma \subseteq \phi$. By the fact that ϕ is closed under $tpre$, $tpre(\sigma) \subseteq \phi$. Now $s \in \phi$ and $P_0^A(s \underset{TPS}{\rightsquigarrow} \phi) = 1$ for any adversary $A \in Adv_{TPS}$.
- If $P_0^B(\sigma \underset{SPS_k}{\rightsquigarrow} \phi) = 0$, then for any adversary $A \in Adv_{TPS}$ the equation holds.

Next, suppose that (c) holds for some $n \in \mathbb{N}$. If $\sigma \subseteq \phi$ the result follows as in the case for $n = 0$. We are therefore left to consider the case when $\sigma \nsubseteq \phi$.

By construction of $SPS_k, B(0, \sigma) = \pi$ for some $(\sigma, \pi) \in Steps_k$, and from line 21 of *CBMaxReachAlg*, there exist $(l, g, p) \in pE$ and a set of edges $E_\pi \subseteq E_{(l,g,p)}$ such that for any $\tau \in \Sigma_k$.

$$\pi(\tau) = \sum_{(\sigma, r, l', \tau) \in E_\pi} p(r, l')$$

If B' is the adversary such that for any $\tau \in \Sigma_k$: $P_n^{B'}(\tau \rightsquigarrow \phi) = P_n^B(\sigma \xrightarrow{\pi} \tau \rightsquigarrow \phi)$, then from Definition 4.4.1 and the construction of π we have:

$$\begin{aligned} P_{n+1}^B(\sigma \rightsquigarrow \phi) &= \sum_{\tau \in \Sigma} \pi(\tau) \cdot P_n^B(\sigma \xrightarrow{\pi} \tau \rightsquigarrow \phi) \\ &= \sum_{\tau \in \Sigma} \pi(\tau) \cdot P_n^{B'}(\tau \rightsquigarrow \phi) \\ &= \sum_{(\sigma, r, l', \tau) \in E_\pi} p(r, l') \cdot P_n^{B'}(\tau \rightsquigarrow \phi) \end{aligned} \tag{5.1}$$

We let $s = (l, v, c)$. Since $(l, v, c) \in tpre(\sigma)$, it follows that there exists $d \in \mathbb{R}_+$ such that $(l, v + d, c + \dot{\$}(l)d) \in \sigma$ and $((l, v, c) \xrightarrow[\mathsf{time}]{d, \{\cdot \mapsto 1\}} (l, v + d, c + \dot{\$}(l)d)) \in Steps$. Now, for any $(\sigma, r, l', \tau) \in E_\pi$ using (a) we have that $(l', (v + d)[r := 0], c + \dot{\$}(l)d) \in tpre(\tau)$. Therefore, by induction, for any $(\sigma, r, l', \tau) \in E_\pi$ there exists $A^{(\sigma,r,l',\tau)} \in Adv_{TPS}$ such that:

$$P_n^{A^{(\sigma,r,l',\tau)}}(l', v + d[r := 0], c + \dot{\$}(l)d) \rightsquigarrow \phi) \geq P_n^{B'}(\tau \rightsquigarrow \phi) \tag{5.2}$$

Let $A \in Adv_{TPS}$ be the adversary such that

- $A(0, (l, v, c)) = (d, \mu_p)$, it chooses the full probabilistic transition that combines the time transition and discrete transitions. The full probabilistic transition exists by Definition 2.3.4. Now by Definition 3.1.2, for any $(l', v', c + \dot{\$}(l)d) \in S$:

$$\mu_p(l', v', c + \dot{\$}(l)d) = \sum_{\substack{r \subseteq \mathbb{X} \\ v' = (v+d)[r:=0]}} p(r, l')$$

- $A(1,(l',(v+d)[r:=0],c+\dot{\$}(l)d)) = A^{(\sigma,r,l',\tau)}(0,(l',(v+d)[r:=0],c+\dot{\$}(l)d))$.

Now we are able to complete the proof of (c).

$$
\begin{aligned}
&P^A_{n+1}((l,v,c) \rightsquigarrow \phi)\\
&= \sum_{(l',v',c+\dot{\$}(l)d)\in S} \mu_p(l',v',c+\dot{\$}(l)d)\cdot P^A_n((l,v,c) \xrightarrow[\mathsf{full}]{d,\mu_p} (l',v',c+\dot{\$}(l)d) \rightsquigarrow \phi)\\
&\qquad\qquad \text{by Definition 4.4.1}\\
&= \sum_{(l',v',c+\dot{\$}(l)d)\in S} \left(\sum_{\substack{r\subseteq\mathbb{X}\\ v'=(v+d)[r:=0]}} p(r,l') \right) \cdot P^A_n((l,v,c) \xrightarrow[\mathsf{full}]{d,\mu_p} (l',v',c+\dot{\$}(l)d) \rightsquigarrow \phi)\\
&\qquad\qquad \text{by Definition 3.1.2}\\
&= \sum_{(r,l')\in \mathrm{support}(p)} p(r,l')\cdot P^A_n((l,v,c) \xrightarrow[\mathsf{full}]{d,\mu_p} (l',(v+d)[r:=0],c+\dot{\$}(l)d) \rightsquigarrow \phi)\\
&\qquad\qquad \text{by rearranging}\\
&\geq \sum_{(\sigma,r,l',\tau)\in E_\pi} p(r,l')\cdot P^A_n((l,v,c) \xrightarrow[\mathsf{full}]{d,\mu_p} (l',(v+d)[r:=0],c+\dot{\$}(l)d) \rightsquigarrow \phi)\\
&\qquad\qquad \text{by Definition of } E_\pi\\
&= \sum_{(\sigma,r,l',\tau)\in E_\pi} p(r,l')\cdot P^{A^{(\sigma,r,l',\tau)}}_n((l',(v+d)[r:=0],c+\dot{\$}(l)d) \rightsquigarrow \phi)\\
&\qquad\qquad \text{by construction of } A\\
&\geq \sum_{(\sigma,r,l',\tau)\in E_\pi} p(r,l')\cdot P^{B'}_n(\tau \rightsquigarrow \phi) \qquad \text{by (5.2)}\\
&= P^B_{n+1}(\sigma \rightsquigarrow \phi) \qquad \text{by (5.1)}
\end{aligned}
$$

Since k, σ and B are arbitrary, (c) holds by induction.

Proof of property (d)

Consider any $A \in Adv_{TPS}$ and $s \in S$ such that $P^{\max}_n(s \underset{TPS}{\rightsquigarrow} \phi) > 0$. We prove (d) by induction on n. For $n = 0$, by Definition 4.4.1, two cases have to be considered.

- If $P^A_0(s \underset{TPS}{\rightsquigarrow} \phi) = 1$, then $s \in \phi$. Now there exists $\sigma \subseteq \phi$, such that $s \in \sigma$, which implies $s \in tpre(\sigma)$. For arbitrary $B \in Adv_{SPS_n}$ the following holds: $P^B_0(\sigma \rightsquigarrow \phi) = 1$.
- If $P^A_0(s \underset{TPS}{\rightsquigarrow} \phi) = 0$, then the premissa of (d) holds, so (d) holds vacuously.

Now suppose that (d) holds for some $n \in \mathbb{N}$. If $P^A_{n+1}(s \underset{TPS}{\rightsquigarrow} \phi) = 0$, then the result follows as in the case when $n = 0$. It therefore remains to consider the

case when $P^A_{n+1}(s \underset{TPS}{\rightsquigarrow} \phi) > 0$, and from Definition 3.1.2 the following three cases have to be considered.

(Case 1) If $A(0,s) = (d, \{t \mapsto 1\})$ is a time transition, there are two possibilities: if $t \in \phi$ then $s \in \phi$, as ϕ is closed under *tpre*, the result follows similarly to when $n = 0$. If $t \notin \phi$, then

$$P^A_{n+1}(s \rightsquigarrow \phi) = P^A_n(s \xrightarrow[\mathsf{time}]{d,\{t\mapsto 1\}} t \rightsquigarrow \phi) > 0 \tag{5.3}$$

We have $P^A_n(t \rightsquigarrow \phi) > 0$, applying induction and the definition of *tpre* there exists an adversary $B \in Adv_{SPS_n}$ such that:

$$P^A_n(s \xrightarrow[\mathsf{time}]{d,\{t\mapsto 1\}} t \rightsquigarrow \phi) = P^B_n(\sigma \underset{SPS_n}{\rightsquigarrow} \phi)$$

, with $s \in tpre(\sigma)$.

Recall that an SPS is easily converted to an equivalent PS, and we can apply Lemma 4.4.2, completing the proof of this case.

$$P^B_n(\sigma \underset{SPS_n}{\rightsquigarrow} \phi) \leq P^B_{n+1}(\sigma \underset{SPS_{n+1}}{\rightsquigarrow} \phi)$$

(Case 2) We let $s = (l,v,c)$. If $A(0,(l,v,c)) = (0,\mu)$ is a discrete probabilistic transition, then by Definition 4.4.1 and the fact that the cost doesn't change, we have:

$$P^A_{n+1}((l,v,c) \rightsquigarrow \phi) = \sum_{(l',v',c)\in S} \mu(l',v',c) \cdot P^A_n((l,v,c) \xrightarrow[\mathsf{disc}]{0,\mu} (l',v',c) \rightsquigarrow \phi)$$

Now from Definition 3.1.2, there exists $(l,g,p) \in pE$ such that $v \in g$ and for any $(l',v',c) \in S$:

$$\mu(l',v',c) = \sum_{r\subseteq \mathbb{X} \wedge v' = v[r:=0]} p(r,l')$$

Letting $A^{r,l'}$ be the adversary such that $A^{r,l'}(0,(l',v[r:=0],c)) = A(1,(l',v[r:=0],c))$, it follows from the above that there exists $(l,g,p) \in pE$ such that $v \in g$ and:

$$\begin{aligned} &P^A_{n+1}((l,v,c) \rightsquigarrow \phi) \\ &\quad = \sum_{(l',v',c)\in S} \left(\sum_{r\subseteq \mathbb{X} \wedge v' = v[r:=0]} p(r,l') \cdot P^A_n((l,v,c) \to (l',v',c) \rightsquigarrow \phi) \right) \\ &\qquad = \sum_{(r,l')\in \mathrm{support}(p)} p(r,l') \cdot P^{A^{r,l'}}_n((l',v[r:=0],c) \rightsquigarrow \phi) \end{aligned} \tag{5.4}$$

Now consider any $(r, l') \in \text{support}(p)$ such that $P_n^{A^{r,l'}}((l', v[r := 0], c) \rightsquigarrow \phi) > 0$. We have $P_n^{\max}((l', v[r := 0], c) \rightsquigarrow \phi) > 0$. By definition $(l, g, p, r, l') \in E_{WPTA}$ and by induction there exists symbolic state $\tau_{r,l'} \in \Sigma_n$ and adversary $B^{r,l'}$ such that

$$P_n^{B^{r,l'}}(\tau_{r,l'} \rightsquigarrow \phi) \geq P_n^{A^{r,l'}}((l', v[r := 0], c) \rightsquigarrow \phi) \tag{5.5}$$

,with $(l', v[r := 0], c) \in \mathit{tpre}(\tau_{r,l'})$. Letting $\sigma_{r,l'} = \mathit{pre}_{(l,g,p,r,l')}(\tau_{r,l'})$, then $(\sigma_{r,l'}, r, l', \tau_{r,l'}) \in E_{(l,g,p)}, \sigma_{r,l'} \in \Sigma_{n+1}$ and $(l, v, c) \in \sigma_{r,l'}$. Therefore, from construction of SPS_{n+1} the following choice for σ is an existing symbolic state in Σ_{n+1}:

$$\sigma = \bigcap \{\sigma_{r,l'} \mid (r, l') \in \text{support}(p) \text{ and } P_n^{\max}((l', v[r := 0], c), \phi) > 0\} \tag{5.6}$$

We have that σ contains state (l, v, c), thus $(l, v, c) \in \sigma$. Furthermore, by construction of SPS_{n+1} there exists $(\sigma, \pi) \in \mathit{Steps}_{n+1}$ such that for any $\tau \in \Sigma_{n+1}$:

$$\pi(\tau) = \sum_{(\sigma, r, l', \tau) \in E_\pi} p(r, l') \geq \sum_{\substack{(r,l') \in \text{support}(p) \\ \tau = \tau_{r,l'} \\ P_n^{A^{r,l'}}((l', v[r:=0], c) \rightsquigarrow \phi) > 0}} p(r, l') \tag{5.7}$$

Now, set B to be the adversary of SPS_{n+1} such that $B(0, \sigma) = \pi$ and $\forall m \geq 0. B(m + 1, s) = B^{r,l'}(m, s)$. Choose σ as in 5.6. We are able to complete the proof of (d).

$$\begin{aligned}
&P_{n+1}^B(\sigma \rightsquigarrow \phi) = \sum_{\tau \in \Sigma_n} \pi(\tau) \cdot P_n^B(\sigma \xrightarrow{\pi} \tau \rightsquigarrow \phi) && \text{by Definition 4.4.1} \\
&\geq \sum_{\substack{(r,l') \in \text{support}(p) \\ P_n^{A^{r,l'}}((l', v[r:=0], c) \rightsquigarrow \phi) > 0}} p(r, l') \cdot P_n^B(\sigma \xrightarrow{\pi} \tau_{r,l'} \rightsquigarrow \phi) && \text{by (5.7)} \\
&= \sum_{\substack{(r,l') \in \text{support}(p) \\ P_n^{A^{r,l'}}((l', v[r:=0], c) \rightsquigarrow \phi) > 0}} p(r, l') \cdot P_n^{B^{r,l'}}(\tau_{r,l'} \rightsquigarrow \phi) && \text{by construction of } B^{r,l'} \\
&\geq \sum_{\substack{(r,l') \in \text{support}(p) \\ P_n^{A^{r,l'}}((l', v[r:=0], c) \rightsquigarrow \phi) > 0}} p(r, l') \cdot P_n^{A^{r,l'}}((l', v[r := 0], c) \rightsquigarrow \phi) && \text{by (5.5)} \\
&= \sum_{(r,l') \in \text{support}(p)} p(r, l') \cdot P_n^{A^{r,l'}}((l', v[r := 0], c) \rightsquigarrow \phi) && \text{drop sum condition} \\
&= P_{n+1}^A((l, v, c) \rightsquigarrow \phi) && \text{by (5.4)}
\end{aligned}$$

(Case 3) If $A(0, s) = (d, \mu)$ is a full probabilistic transition, then we can simply construct adversary A' that replaces this by a timed transition directly followed by a discrete probabilistic transition.

$$
\begin{aligned}
&P^{A}_{n+1}(s \rightsquigarrow \phi) = P^{A'}_{n+2}(s \rightsquigarrow \phi) && \text{by construction of } A' \\
&= P^{A'}_{n+1}(s \xrightarrow[\mathsf{time}]{d,\{t \mapsto 1\}} t \rightsquigarrow \phi) && \text{by (5.3)} \\
&= P^{A'}_{n+1}(t \rightsquigarrow \phi) \\
&\leq P^{B}_{n+1}(\sigma \rightsquigarrow \phi)
\end{aligned}
$$

The last step holds by analogy to the case for a discrete probabilistic transition and the fact that the full transition can't be eliminated twice, for some $\sigma \in \Sigma_n$ and $B \in Adv_{SPS_n}$.

5.1.3 Properties of algorithm *CBMaxReachAlg*

Infinite symbolic semantics

The SPS corresponding to a given TPS may have a infinite symbolic state space, and thus *CBMaxReachAlg* is not guaranteed to terminate. When no intersections are added, i. e. line 19 is skipped, *CBMaxReachAlg* will terminate. First note that the target symbolic states in ϕ can be described using the extended priced zones of Lemma 3.5.3. Using our assumption that target symbolic states have the same maximal cost for every clock valuation, we can use algorithm *BwReachability* in the form of Theorem 3.5.2 to decide backward reachability, and this terminates. Unions of these extended priced zones exactly describe the mp-zones generated by *BwReachability* when no intersections are added. Therefore without intersections, *CBMaxReachAlg* will terminate.

Intersections can have predecessors that again create new intersections, and in this way infinitely many intersections are added. Figure 5.5 gives an example WPTA, and the part of the symbolic state space that is generated with $n = 5$ iterations. p_1 and p_2 denote the probability distributions on the probabilistic edges of the WPTA. Note that cost is on the x-axis, and *pre* is not split up. The plain arrows are the edges from the WPTA that are used in pre_e. The dashed arrow are the edges in the sets $E_{(l,g,p)}$. A symbolic state is inscribed with '$\cap$' to denote it is an intersection. Figure 5.6 shows the generated sub-probabilistic system SPS. We clearly see converging probability values. From figure 5.5 it is clear that an infinite symbolic state space would be generated if *CBMaxReachAlg* is not aborted at some point.

From figure 5.5 and literature [BBR04] we conclude that in general WPTA do **not have finite bisimulations** that respect all possible integer cost bounds. Recall from chapter 4 that cost-bounded probabilistic reachability can be formulated in the logic PBTL, but that by absence of a finite bisimulation we can't conclude there is a decidable model checking procedure [Spr00].

Application to broader classes

Clearly decidability of non-probabilistic reachability is a necessary condition for *CBMaxReachAlg* to work correctly. Therefore a similar approach for the

broader class of probabilistic linear hybrid automata, will not work, as the reachability problem for these kind of systems is undecidable [HKPV98, BBR04]. Subclasses of probabilistic linear hybrid automata are known that have finite bisimulations [Spr00]. A well know subclass are of course probabilistic timed automata. By having a finite bisimulation, maximal probabilistic reachability is decidable [Spr00].

5.2 Descending Converging Value

In the previous section we have presented algorithm *CBMaxReachAlg* that computes cost-bounded maximal probabilistic reachability as a (possibly infinite) sequence of values converging to the exact value. Assume given the cost-bounded maximal probabilistic reachability problem $\Xi = (T, \sqsupseteq, \lambda, \kappa)$ (see 3.2). Let p denote the exact value that is approximated. If λ is such that $\lambda \geq p$ we cannot use *CBMaxReachAlg* to give a verdict on Ξ, although the verdict should be "no". This is unsatisfying, we have investigated methods to give a negative verdict if $\lambda > p$. Our solution is to compute p with a descending (or non-increasing) converging sequence.

First we would like to point out that possibly by investigating properties of convergence of *CBMaxReachAlg* it is possible to decide the problem. This is a topic for further research. The approaches presented here try to find another way to compute maxprob that creates also an infinite converging sequence of values, but now with values that are descending. Note that still, if we would choose $\lambda = p$, the problem is not decidable by our algorithms.

We give non-formal descriptions of two possible algorithms to compute p with a descending converging sequence. The first algorithm (section 5.2.1) is incorrect for some (finite) paths, but it remains to be investigated if it would be correct for paths of length greater than some integer. We think the second algorithm (section 5.2.2) is correct. Both algorithms rely on computing the probability of reaching a set of states, denoted *fail*, that are certainly wrong. Let ψ denote the target states representable as a set of symbolic states. *fail* are those states from which there is no path to a state in ψ. Clearly paths from *fail* can only reach states in *fail*. Note that typically $\psi \cup \mathit{fail} \neq S$, otherwise we can give the verdict right away.

The first step in constructing *fail* consists of using *BwReachability* on page 46 to get the set of states that can reach ψ. *BwReachability* returns the set of symbolic states *Visited*. Now the set of symbolic states $G = \{\mathit{tpre}(\sigma) \mid \sigma \in \mathit{Visited}\}$ represents all states that can reach ψ. By interpreting G semantically as a set of states, we have $\mathit{fail} = S \backslash G$. Both S and G can be described as a finite union of symbolic states that use convex polyhedra (see section 3.6). Then, by theory on geometry, *fail* can also be described as a finite union of symbolic states that use convex polyhedra. We know that the predecessor operator is computable on these symbolic states. In contrast to G, *fail* cannot use multi-priced zones, as mp-zones are not closed under the difference operator ($\backslash$).

5.2.1 Fail reachability

The algorithm described here is in fact not an algorithm on its own, but merely an extension of algorithm *CBMaxReachAlg*. Algorithm *CBMaxReachAlg* computes in each iteration a probability maximizing adversary on the constructed SPS, using some existing technique. Note that possibly more than one adversary in the SPS will give the maximal reachability probability. According to literature [Tij03, dA99], it should be possible to compute the whole set of maximal adversaries Γ. By incorporating *fail* as a special state in the SPS, we can compute the reachability probability on *fail* for all adversaries in Γ. The idea was that in each iteration of the new algorithm, the minimum of the reach probabilities on *fail* for all adversaries in Γ, would give us a sequence of values that converges from above to the actual solution of cost-bounded maximal probabilistic reachability. We will see however the incorrectness of this approach.

We skip the details of how *fail* can be incorporated as a special state in the SPS generated by the algorithm, and proceed with the example that shows the incorrectness. Figure 5.7 depicts a WPTA. The part in the dashed box is equal to the WPTA of figure 5.7. l_2 is again the target location. From the starting location, there is a choice between α and β. Clearly $fail = (l_3, \{(v, c) \mid v \in \mathbb{R}^{\mathbb{X}}_+ \wedge c \geq 0\})$, which is representable by a symbolic state. The arrow to ? means that after this transition, it is unclear with what probability it is possible to reach the target. From previous results on the WPTA in the dashed box we know that in iteration $n = 4$, the probability from l_0 reaching the target is 0.372. Thus for iterations 0–3, the maximizing adversaries Γ (in this case only one) will choose β. And from iterations 4 and higher, they will choose α. But, the minimal probability of reaching *fail*, when choosing β is 0.61, whereas the minimal probability of reaching *fail* in iteration 4, when choosing α is 0.62. Thus, in fact the minimal reach probability on *fail* increases, showing the incorrectness of this approach.

What remains to be investigated is if there exists a number of iterations from where the probability on reaching *fail* can only descend. For example, in figure 5.7, assuming that from ? the target cannot be reached, this number would be iteration 4.

5.2.2 Cost-bounded minimal probabilistic reachability

The algorithm described here gives an ascending sequence $[F_n]_{n=0\ldots\infty}$ that converges to the the minimal reachability probability on some target set of symbolic states. Now we can take *fail* as target set. If p denotes the exact maximal reachability probability for the target set, and q denotes the exact minimal reachability probability on *fail*, then clearly $p = 1 - q$. Therefore with this algorithm it is possible to give a sequence $[1 - F_n]_{n=0\ldots\infty}$, that is descending and converges to p.

[KNS03] presents a method to compute minimal probabilistic reachability, but this method is not suited for our purpose, as in fact it uses the maximal probabilistic reachability result p and computes the minimal reachability probability q as $q = 1 - p$.

The idea of our new algorithm is to compute in each iteration n, a weaker version of the coarsest time-abstracting bisimulation, i. e. two states are bisimilar if they both have the same path leading to the target in n or less steps. We think the equivalence classes are computable with an algorithm similar to algorithm *CBMaxReachAlg*, but now in addition to taking intersections between symbolic states, also the differences between them are computed, like the algorithms in [TY01] do. On this state space we can construct a sub-probabilistic system (SPS) similar to *CBMaxReachAlg*. The next step is to compute minimal probabilistic reachability on this SPS using techniques of [dA99].

In each iteration the state space grows, and represents equivalence classes which respect longer paths to the target. We are convinced this will result in an ascending sequence of minimal reachability probabilities that converges to the exact minimal probabilistic reachability value. From [KNS03, KNSS02] we conclude that T-divergent adversaries (see section 3.2) should be considered. If the state space grows, longer paths to the target are possible. Therefore a T-divergent adversary in iteration n will give a higher probability on reaching the target in iteration $n+1$. Thus, the minimal adversary in $n+1$ will have a higher probability on reaching the target. The correspondence between symbolic system and semantical system is clear. The difference between symbolic states is needed, because we are interested in symbolic states were as less as possible edges of the same probabilistic edge may lead to the target. These will give minimal probabilistic reachability.

Figure 5.3: A WPTA and its symbolic state graph.

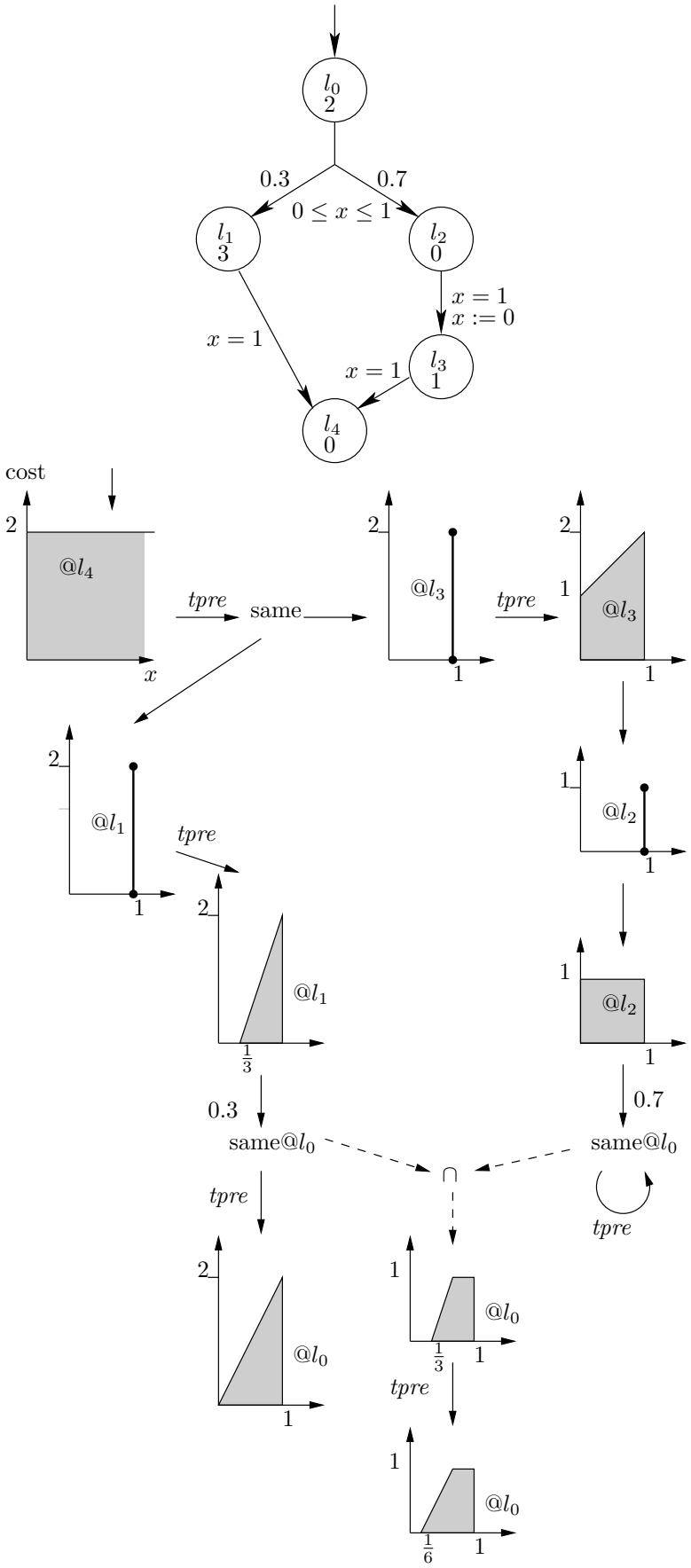


Figure 5.4: A WPTA and its symbolic state graph.

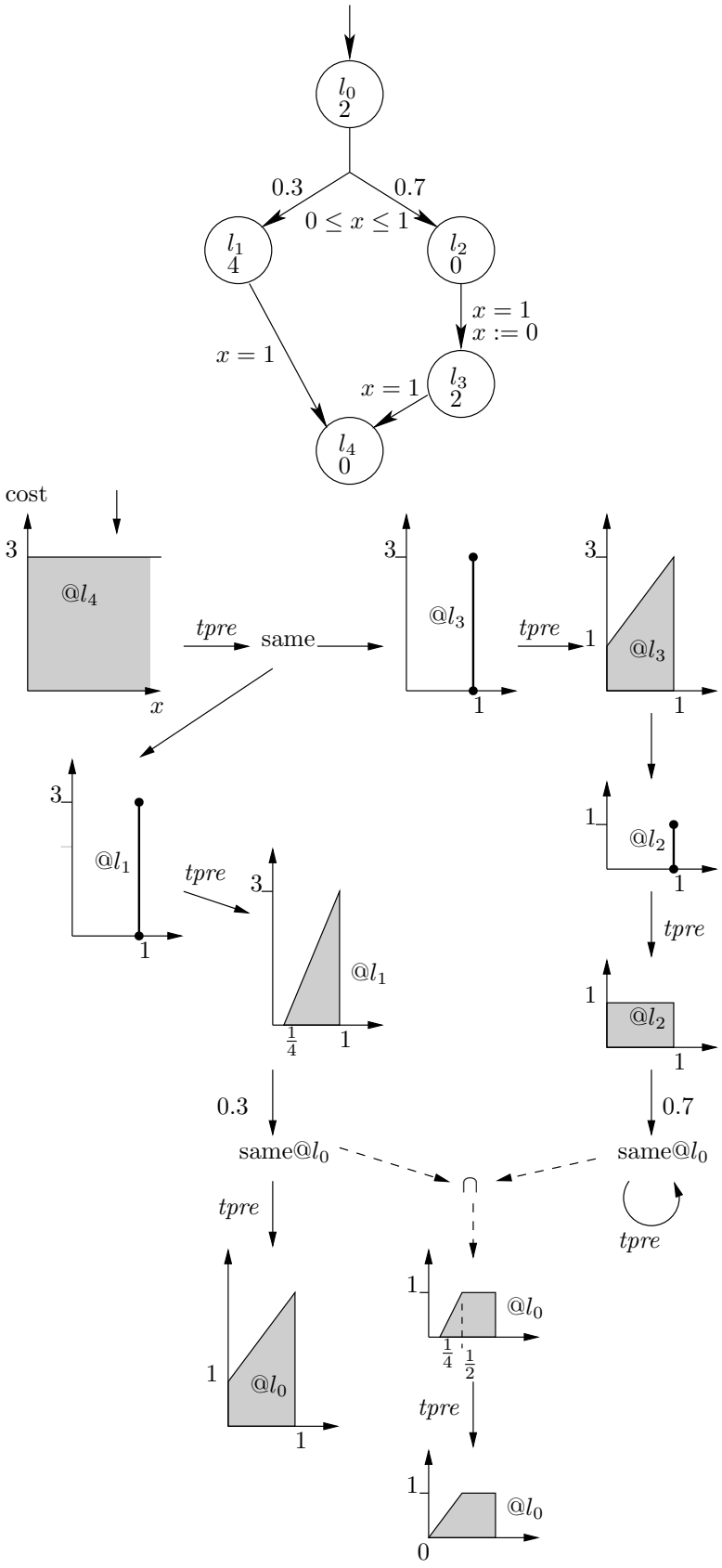


Figure 5.5: Example of an infinite symbolic state space [BBR04].

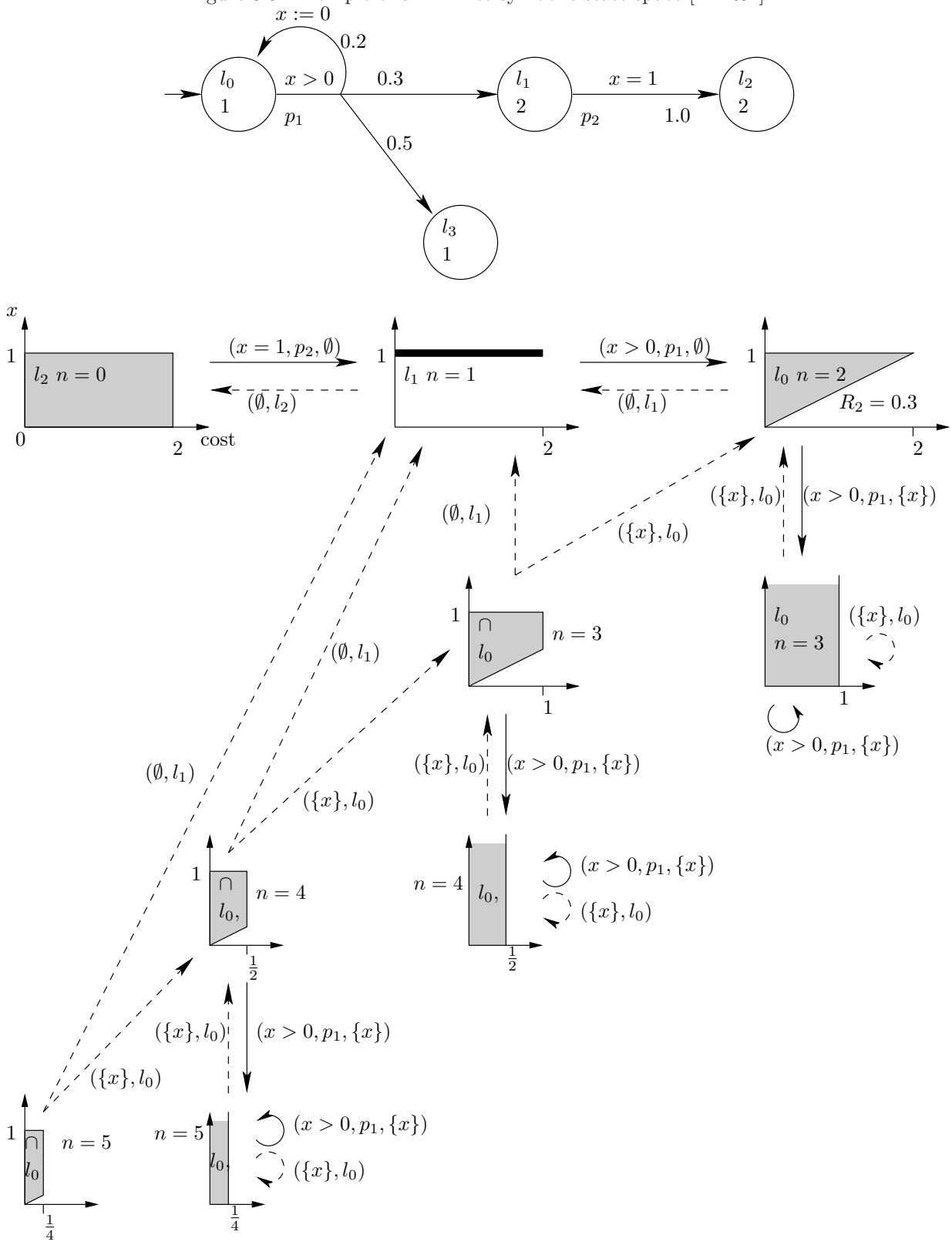


Figure 5.6: SPS generated from the state space graph of figure 5.5. Every symbolic state is placed in somewhat the same place. The numbers on the arrows give the probabilities from the sub-probabilistic distributions. The numbers in the symbolic states give the maximum reachability probability of the target from that symbolic state.

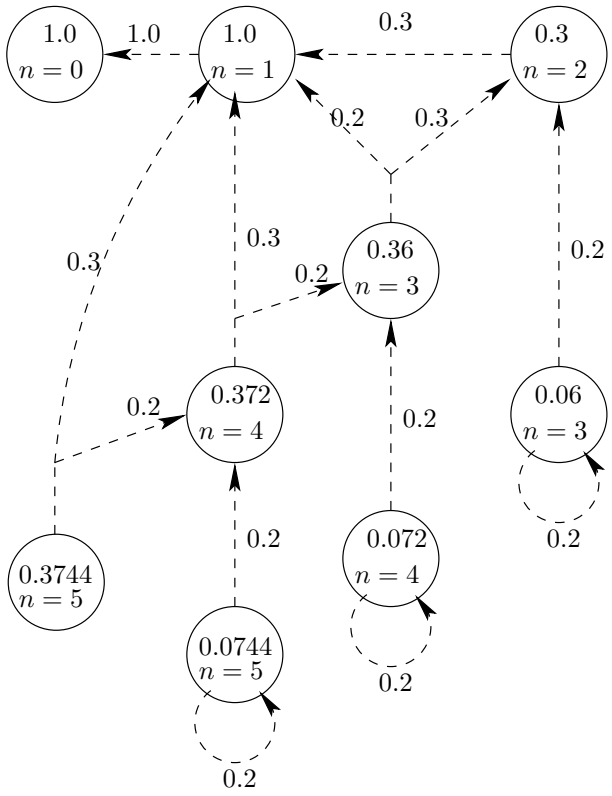
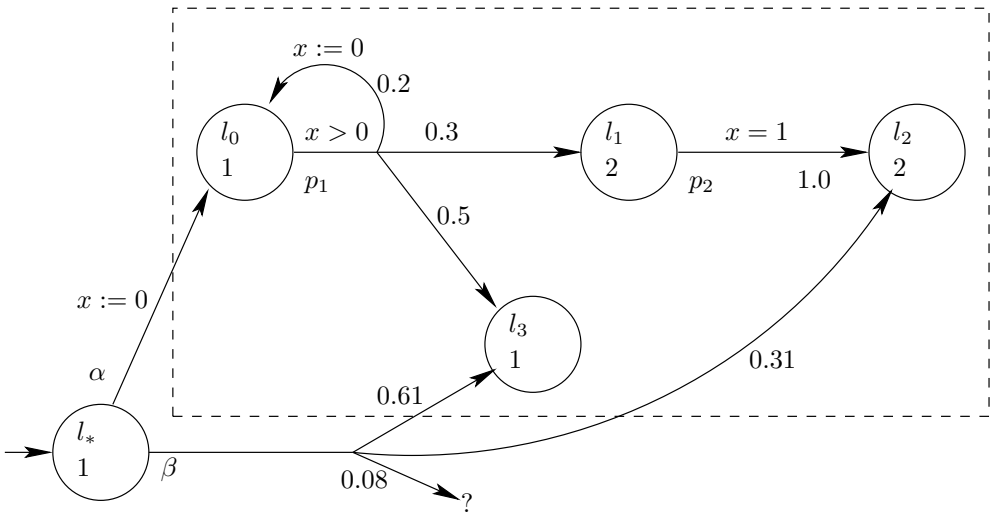


Figure 5.7: Counter-example of method using fail reachability.



Conclusion

We introduced a new model called Weighted Probabilistic Timed Automata. WPTA are timed automata with prices in locations and probabilistic edges. I investigated model checking of *cost-bounded probabilistic reachability* properties on WPTA. These properties state that a location is reachable with probability λ and cost at most κ. To check these properties we need to compute *cost-bounded maximal probabilistic reachability* on WPTA.

Although we know from literature that the problems of minimal cost reachability [BFH+01, ATP01] and maximal probabilistic reachability [KNSS02] (both on locations in WPTA) are decidable, the problem of cost-bounded maximal probabilistic reachability is not trivial. The reason for this is that WPTA do not have finite time abstracting bisimulations respecting cost bounds. Our first naive solution resulted in a forward algorithm, that uses priced regions. The algorithm is capable of computing an upper bound. Now we can only conclude to a negative verdict on some cost-bounded probabilistic reachability property, but cannot give a positive verdict for certain.

We have investigated backward algorithms and constructed an efficient backward algorithm that computes cost-bounded reachability, where probabilities are ignored. This provides no fundamental new results, but shows that backward reachability using priced zones is possible. The algorithm is efficient in the sense that predecessor symbolic states are easily computable on the facets of priced zones. Next, we introduced a new abstraction called multi-priced zones, which are a subclass of rational polyhedra. Multi-priced zones are closed under the priced versions of predecessor operations and conjunction. Predecessor operations are realizable as operations on polyhedra [HPR94, AHH96], but we do not make assumptions of the used method.

With symbolic states consisting of a location and multi-priced zone we can use an altered version of the algorithm of [KNS03] that computes maximal probabilistic reachability for probabilistic timed automata. The difference in our algorithm is that we do breadth-first backward exploration of the symbolic state space, where in each iteration of the algorithm, the depth of exploration increases. The algorithm starts with a set containing all symbolic states that represent the target location and cost bound. From this set successively predecessors are computed in a breadth-first manner. Like [KNS03] only intersections between generated symbolic states need to be computed. The algorithm gives a better approximation of maximal probabilistic reachability in each iteration, but may not terminate. In this way an ascending converging sequence of values

is obtained. When some cost-bounded probabilistic reachability property uses a probability bound that is strictly lower than the exact solution of maximal probabilistic reachability, we are able to conclude to a positive verdict.

Directions for Further Research

Algorithm *CBMaxReachAlg* only gives a partial solution of the problem. Therefore a natural direction for further research is the full solution. Our own steps in this direction concerned finding algorithms that compute the minimal probabilistic reachability on states that are certainly not in the target (see section 5.2). Another important direction is to study the approximating behavior of *CBMaxReachAlg*. For certain WPTA *CBMaxReachAlg* will terminate, returning a system on which the exact probability can be computed. Of interest are the subclasses of WPTA for which this is the case. Finally, one could look for some other kind of abstraction that is finite, and can be used for probability computation.

The presented algorithms are implementable in their present form, but for efficient implementations optimizations are useful. In *CBMaxReachAlg* the calculation of maximal probabilistic reachability is performed on the generated SPS. It would be interesting to see if these computations could be integrated into the algorithm itself. We think that symbolic states completely included in another symbolic state, and with a lower probability of reaching the target can be discarded. Operations on price functions, and intersections are expensive, therefore methods that already discard part of the state space are of interest. Finally, we think symbolic states can share their unpriced part if it is the same, thus reducing memory usage.

The model of WPTA can be extended in numerous ways. All extensions according to probabilistic linear hybrid automata are of interest. The most obvious is by adding discrete cost increments like LPTA. We have excluded discrete cost increments from our model to completely focus on the problems with prices in locations. We are convinced that the algorithms are easily adapted for discrete cost increments by small adjustments to the successor/predecessor operations.

On WPTA, other properties can be formulated. Two interesting properties are: expected cost reachability, and average cost reachability. Also a logic like PTCTL could be formulated.

Bibliography

[ACD93] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking in dense real-time. *Inf. Comput.*, 104(1):2–34, 1993.

[ACH+95] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.*, 138(1):3–34, 1995.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[AHH96] Rajeev Alur, Thomas A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. Software Eng.*, 22(3):181–201, 1996.

[ATP01] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In Maria Domenica Di Benedetto and Alberto L. Sangiovanni-Vincentelli, editors, *HSCC*, volume 2034 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2001.

[BBR04] Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. Model-checking for weighted timed automata. In Lakhnech and Yovine [LY04], pages 277–292.

[BDFP00] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Are timed automata updatable? In E. Allen Emerson and A. Prasad Sistla, editors, *CAV*, volume 1855 of *Lecture Notes in Computer Science*, pages 464–479. Springer, 2000.

[BFH+01] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-Cost Reachability for Priced Timed Automata. In Maria Domenica Di Benedetto and Alberto Sangiovanni-Vincentelli, editors, *Proceedings of the 4th International Workshop on Hybris Systems: Computation and Control*, number 2034 in Lecture Notes in Computer Sciences, pages 147–161. Springer–Verlag, 2001.

[BY03] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In Jörg Desel, Wolfgang Reisig, and Grzegorz

Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 87–124. Springer, 2003.

[dA99] Luca de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In Jos C. M. Baeten and Sjouke Mauw, editors, *CONCUR*, volume 1664 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 1999.

[Dil89] David L. Dill. Timing assumptions and verification of finite-state concurrent systems. In Joseph Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1989.

[Hen95] Thomas A. Henzinger. Hybrid automata with finite bisimulatioins. In Zoltán Fülöp and Ferenc Gécseg, editors, *ICALP*, volume 944 of *Lecture Notes in Computer Science*, pages 324–335. Springer, 1995.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proc. of the London Math. Soc.*, 2:326–336, 1952.

[HKPV98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *J. Comput. Syst. Sci.*, 57(1):94–124, 1998.

[HNSY92] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. In *7th. Symposium of Logics in Computer Science*, pages 394–406, Santa-Cruz, California, 1992. IEEE Computer Scienty Press.

[HPR94] Nicolas Halbwachs, Yann-Eric Proy, and Pascal Raymond. Verification of linear hybrid systems by means of convex approximations. In *SAS*, pages 223–237, 1994.

[KNS01] Marta Z. Kwiatkowska, Gethin Norman, and Jeremy Sproston. Symbolic computation of maximal probabilistic reachability. In Kim Guldstrand Larsen and Mogens Nielsen, editors, *CONCUR*, volume 2154 of *Lecture Notes in Computer Science*, pages 169–183. Springer, 2001.

[KNS03] M. Kwiatkowska, G. Norman, and J. Sproston. Symbolic model checking for probabilistic timed automata. Technical Report CSR-03-10, School of Computer Science, University of Birmingham, 2003.

[KNSS02] Marta Z. Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theor. Comput. Sci.*, 282(1):101–150, 2002.

[KNSW04] Marta Z. Kwiatkowska, Gethin Norman, Jeremy Sproston, and Fuzhi Wang. Symbolic model checking for probabilistic timed automata. In Lakhnech and Yovine [LY04], pages 293–308.

[KSK76] J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains*. Springer, 2nd edition, 1976.

[LBB+01] Kim Guldstrand Larsen, Gerd Behrmann, Ed Brinksma, Ansgar Fehnker, Thomas Hune, Paul Pettersson, and Judi Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *CAV*, volume 2102 of *Lecture Notes in Computer Science*, pages 493–505. Springer, 2001.

[LY04] Yassine Lakhnech and Sergio Yovine, editors. *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences FORMATS and FTRTFT Proceedings*, volume 3253 of *Lecture Notes in Computer Science*. Springer, 2004.

[Spr00] Jeremy Sproston. Decidable model checking of probabilistic hybrid automata. In Mathai Joseph, editor, *FTRTFT*, volume 1926 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 2000.

[Sto02] Mariëlle Stoelinga. An introduction to probabilistic automata. *Bulletin of the EATCS*, 78:176–198, 2002.

[Tij03] Henk C. Tijms. *A First Course in Stochastic Models*. John Wiley & Sons, 2003.

[TY01] Stavros Tripakis and Sergio Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, 2001.