Title: **Discounted Markov Automata**

Author: Yuliya Butkova

Report Number: **2018-01**

ERC Project: **Power to the People. Verified.**

ERC Project ID: **695614**

Funded Under: **H2020-EU.1.1. – EXCELLENT SCIENCE**

Host Institution: **Universität des Saarlandes, Dependable Systems and Software Saarland Informatics Campus**
Discounted Markov Automata*

Yuliya Butkova
Saarland University, Saarbrücken, Germany
butkova@cs.uni-saarland.de

Abstract. Markov automata (MA) are a rich modelling formalism for complex systems combining compositionality with probabilistic choices and continuous stochastic timing. Model checking algorithms for different classes of properties involving probabilities and rewards have been devised for MA, opening up a spectrum of applications in dependability engineering and artificial intelligence, reaching out into economy and finance. In the latter more general contexts, several quantities of considerable importance are based on the idea of discounting reward expectations, so that the near future is more important than the far future. This paper introduces the expected discounted reward value for MA and develops effective iterative algorithms to quantify it, based on value- as well as policy-iteration. To arrive there, we reduce the problem to the computation of expected discounted rewards and expected total rewards in Markov decision processes. This allows us to adapt well-known algorithms to the MA setting. Experimental results clearly show that our algorithms are efficient and scale to MA with hundred thousands of states.

1 Introduction

The design and analysis of complex systems operating in uncertain environments requires a powerful modelling language. It is desirable to support compositionality for constructing large models from individual components; nondeterminism for abstraction and representing unknown behaviour of the environment; continuous stochastic timing and probabilistic choices. Markov automata (MA) combine all these aspects in one formalism. They have been extended to express costs and rewards, yielding Markov reward automata. Efficient algorithms for the automatic analysis of Markov (reward) automata are available for a broad range of properties like time- and cost-bounded reachability probabilities, expected rewards, long-run averages and properties expressed in the temporal logic CSL. Tool support is available: Both IMCA and Storm support Markov automata model checking. This makes Markov automata well suited not only as a modelling formalism by itself, but also as the semantical foundation of higher-level formalisms like dynamic fault trees and generalized stochastic Petri nets.

* This work was partly supported by the ERC Advanced Grant POWVER (695614) and by the Sino-German project CAP (GZ 1023).
All the measures considered so far for Markov reward automata do not make a difference between a unit of reward being accumulated early or late along the evolution of the system. In economics, in artificial intelligence, and in the theory of optimal control [3], it is a well-understood practice to discount the future, that is, to give more weight to the near future than to the far away future. This view is natural in model checking quantitative temporal logic properties of systems [2], including continuous-time Markov decision processes (CTMDPs) [10]. It appears equally natural for Markov automata, but as yet, there is neither a theory nor an algorithmic approach for discounting in Markov reward automata.

The present paper investigates discounting on MRA. We first settle the foundational basis of what discounting actually means for Markov automata. Due to the continuous nature of time in MRA we define discounting analogously to the way it is defined for CTMDPs.

Our findings are rooted in the observation that we can view any MRA as a representation encoding a possibly exponentially larger CTMDP, preserving discounted reward values. This enables one to quantify the discounted reward in MRA by computing the respective value on its value-preserving CTMDP, however at the price of possibly exponential time and space requirements.

Overall our approach has similarities in spirit to the one introduced to quantify long-run average rewards on MRA [5], but the constructions needed have to be entirely different due to the dependency of the discounted reward on time. Instead of the naïve approach, we show that the exponential blow-up can be avoided by recognising that the value requiring exponentially many computational steps as the expected total reward in a specific linear-sized discrete-time MDP. Using classic dynamic programming for the latter then turns the exponential naïve approach into an effective polynomial characterisation. In this way we derive the Bellman equation characterising the expected total reward in the presence of discounting in MRA. The Bellman equation in turn is the basis for value- and policy-iteration algorithms quantifying the discounted reward on MRA. The efficiency of the approach is demonstrated with examples of MRAs with hundreds of thousands of states.

### 2 Foundations

Given a finite set $S$, a probability distribution over $S$ is a function $\mu : S \rightarrow [0,1]$ with $\sum_{s \in S} \mu(s) = 1$. We denote the set of all probability distributions over $S$ by $\text{Dist}(S)$. $\xi_s$ is the Dirac distribution on $s$, i.e. $\xi_s(s) = 1$ and $\xi_s(s') = 0$ for $s' \neq s$.

**Definition 1.** A Markov reward automaton (MRA) $\mathcal{M}$ is a tuple $\mathcal{M} = (S, s_{\text{init}}, \text{Act}, \rightarrow, \vartrianglerighteq, r, \rho)$ s.t. $S$ is a finite set of states; $s_{\text{init}} \in S$ is the initial state; $\text{Act}$ is a finite set of actions; $\rightarrow \subseteq S \times \text{Act} \times \text{Dist}(S)$ is a finite probabilistic transition relation; $\vartrianglerighteq \subseteq S \times \mathbb{R}_{\geq 0} \times S$ is a finite Markovian transition relation; $r : \rightarrow \rightarrow \mathbb{R}_{\geq 0}$ is a transition reward function; and $\rho : S \rightarrow \mathbb{R}_{\geq 0}$ is a state reward function.
We abbreviate \((s, \alpha, \mu) \in \leadsto\) by \(s \overset{\alpha}{\rightarrow} \mu\) and write \(s \overset{\lambda}{\rightarrow} s'\) instead of \((s, \lambda, s') \in \leadsto\). \(\text{Act}(s) = \{ \alpha \in \text{Act} \mid \exists \mu \in \text{Dist}(S) : s \overset{\alpha}{\rightarrow} \mu \}\) denotes the set of actions that are enabled in state \(s \in S\). A state \(s\) is probabilistic (Markovian), if it has at least one probabilistic (Markovian) transition \(s \overset{\alpha}{\rightarrow} \mu\) (\(s \overset{\lambda}{\rightarrow} s'\), resp.). States can be both probabilistic and Markovian. We denote the set of probabilistic states by \(PS_M\) and the Markovian states by \(MS_M\). We assume w.l.o.g. that actions of probabilistic transitions of a state are pairwise different\(^1\). Therefore we will write \(r(s, \alpha)\) instead of \(r(s, \alpha, \mu)\). The successors of a state \(s \in S\) are given by \(\text{succ}(s) = \{ s' \in S \mid \exists \alpha \in \text{Act} \exists \mu \in \text{Dist}(S) : s \overset{\alpha}{\rightarrow} \mu \land \mu(s') > 0 \lor \exists \lambda \in \mathbb{R}_{>0} : s \overset{\lambda}{\rightarrow} s' \}\) and its predecessors by \(\text{pred}(s) = \{ s' \in S \mid s \in \text{succ}(s') \}\).

For a Markovian state \(s \in MS_M\), the value \(R(s, s') := \sum_{(s, \lambda, s') \in \leadsto} \lambda\) is called the transition rate from \(s\) to \(s'\). The exit rate of a Markovian state \(s\) is \(E(s) := \sum_{s' \in S} R(s, s')\). We require \(E(s) < \infty\) for all \(s \in MS_M\).

For \(s \in PS_M\) with \(s \overset{\alpha}{\rightarrow} \mu\) for some \(\alpha\), we set \(\mathbb{P}[s, \alpha, s'] := \mu(s')\). For \(s \in MS_M\) with \(E(s) > 0\), the branching probability distribution when leaving the state through a Markovian transition is denoted by \(\mathbb{P}[s, \cdot] \in \text{Dist}(S)\) and defined by \(\mathbb{P}[s, s'] := R(s, s')/E(s)\).

The evolution of an MRA starts in its initial state. Whenever the system encounters a Markovian state \(s \in MS_M\), its sojourn time in \(s\) is governed by an exponential distribution, i.e. the probability of leaving \(s\) within \(t \geq 0\) time units is given by \(1 - e^{-E(s) \cdot t}\), after which the next state is chosen according to \(\mathbb{P}[s, \cdot]\).

The behaviour of the system in probabilistic states is different. In this paper we consider closed MRA, which are not subject to further composition operations that could delay the execution of probabilistic transitions. Therefore we can make the usual urgency assumption: Probabilistic transitions happen instantaneously. The residence time in probabilistic states is therefore always 0. Whenever the system is in state \(s\) with \(\text{Act}(s) \neq \emptyset\) and an action \(\alpha \in \text{Act}(s)\) is chosen, the successor \(s'\) is selected according to the distribution \(\mathbb{P}[s, \alpha, \cdot]\) and the system moves instantaneously from \(s\) to \(s'\). As the execution of a probabilistic transition is instantaneous and because the probability that a Markovian transition is triggered immediately is 0, we can assume that the probabilistic transitions take precedence over Markovian transitions. We therefore assume \(PS_M \cap MS_M = \emptyset\). The way Markov automata choose actions will be covered shortly.

During its evolution a Markov reward automaton collects rewards. The transition reward \(r(s, \alpha)\) is granted immediately for taking the (probabilistic)

---

\(^1\) This can be achieved by renaming the actions and does not affect compositionality properties of MRA due to the fact that only closed MRA are considered in this work.
Whenever a task is lost, no further reward for it is paid. Processing high reliability with probability 0.01, while low reliability tasks are lost ten times more often.

whether to handle it with high or low reliability. In the former case, the system needs the notion of a scheduler. A

\[ \text{Paths} \]

\[ \text{Scheduler} \]

\[ \alpha 
\]

\[ \geq \]

\[ k \]

\[ \tau \]

\[ t \]

\[ \text{exists a transition} \]

\[ s \]

\[ \rightarrow \]

\[ s \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]

\[ \rightarrow \]
transitions \((\pi_1, \alpha, \mu) \in \rightarrow\), for some \(\alpha, \mu\). The set of all measurable schedulers is denoted by \(GM_M\). A (deterministic) stationary scheduler is a function \(\sigma : PS_M \rightarrow \rightarrow\), s.t. \(\sigma(s)\) chooses only from transitions \((s, \alpha, \mu) \in \rightarrow\), for some \(\alpha, \mu\).

For the definition of the probability measure on MRA we refer to [14 Sect. 3.2].

**Markov Decision Processes.**

**Definition 2.** A continuous-time Markov decision process \(\text{(CTMDP)}\) is a tuple \(\mathcal{C} = (S, s_{\text{init}}, \text{Act}, R)\), where \(S\) is a finite set of states, \(s_{\text{init}} \in S\) is an initial state, \(\text{Act}\) is a finite set of actions, and \(R : S \times \text{Act} \times S \rightarrow \mathbb{R}_{\geq 0}\) is a rate function.

The set \(\text{Act}(s) = \{\alpha \in \text{Act} \mid \exists s' \in S : R(s, \alpha, s') > 0\}\) is the set of enabled actions in state \(s\). A path in a CTMDP is a finite or infinite sequence \(\pi = s_0 \xrightarrow{\alpha_0, t_0} s_1 \xrightarrow{\alpha_1, t_1} \cdots \xrightarrow{\alpha_{k-1}, t_{k-1}} s_k \cdots\), where \(s_0 = s_{\text{init}}\), \(\alpha_i \in \text{Act}(s_i)\) and \(t_i\) denotes the residence time of the system in state \(s_i\). \(E(s, \alpha) := \sum_{s' \in S} R(s, \alpha, s')\) and \(P_C[s, \alpha, s'] := \frac{R(s, \alpha, s')}{E(s, \alpha)}\). The notions of \(\text{Paths}^s_C, \text{Paths}_C, \pi_i^-\) and schedulers are defined analogously to corresponding definitions for an MRA. A reward structure on a CTMDP \(\mathcal{C}\) is a tuple \((\rho_c, r_c)\), where \(\rho_c : S \rightarrow \mathbb{R}_{\geq 0}\) and \(r_c : S \times \text{Act} \rightarrow \mathbb{R}_{\geq 0}\).

The counterpart of CTMDPs and MRA in discrete time are (discrete-time) Markov decision processes:

**Definition 3.** A Markov decision process \(\text{(MDP)}\) is a tuple \(\mathcal{D} = (S, s_{\text{init}}, \text{Act}_D, P_D)\) where \(S\) is a finite set of states, \(s_{\text{init}}\) is the initial state, \(\text{Act}_D\) is a finite set of actions and \(P_D : S \times \text{Act}_D \rightarrow \text{Dist}(S)\) is a probabilistic transition function.

The definitions of paths, schedulers, etc. are discrete analogues of those definitions for CTMDPs. A reward structure on an MDP is a function \(\gamma_D : S \times \text{Act}_D \rightarrow \mathbb{R}_{\geq 0}\).

In the following a special subclass of MDPs – acyclic MDPs – will be of particular importance. A state of an MDP is called terminal if all its outgoing transitions are self-loops with probability 1 and reward 0. We call an MDP acyclic if the self-loops of terminal states are the only loops appearing in the MDP.

## 3 Discounted Reward for Markov Automata

In this section, we define the discounted reward value for Markov reward automata and consider its relation to discounted rewards on CTMDPs.

### 3.1 Continuous Discounting

Markov automata are a continuous-time model and we therefore define discounting in a classical way via the continuous exponential decay over time. Yet special care has to be taken when dealing with probabilistic states due to the fact that time in those states does not pass. Essentially the definition is lifted to the MRA setting from CTMDPs [15].

Let \(\mathcal{M} = (S, s_{\text{init}}, \text{Act}, \rightarrow, \rho, \beta)\) be a Markov reward automaton, \(\beta > 0\), and \(\pi = s_0 \xrightarrow{\alpha_0, t_0} s_1 \xrightarrow{\alpha_1, t_1} \cdots \xrightarrow{\alpha_{k-1}, t_{k-1}} s_k +1 \xrightarrow{\alpha_{k+1}, t_{k+1}} \cdots\) an infinite path in \(\mathcal{M}\).
Then rew\(_{M,\beta}(\pi)\) is the discounted reward with rate \(\beta\) of the path \(\pi\) within \(N \in \mathbb{N}\) steps, where rew\(^0_{M,\beta}(\pi) := 0\) and

\[
\text{rew}\_N^{N}_{M,\beta}(\pi) := \sum_{k=0}^{N-1} \left[ e^{-\beta \tau[k]} \cdot r(s_k, \alpha_k) + \int_{\tau[k]}^{\tau[k]+t_k} e^{-\beta \tau} \cdot \rho(s_k) \, d\tau \right].
\]

**Example 2.** Consider the MRA from Fig. 1 and its path \(\pi = (nt) \xrightarrow{t_1} (ht) \xrightarrow{t_2} (\text{high}) \xrightarrow{\ldots} \), where \((nt)\) stands for \((\text{no tasks})\) and \((ht)\) for \((\text{has a task})\). Then the discounted reward collected over this path for \(N = 4\) is:

\[
\text{rew}\_4^{N}_{M,\beta}(\pi) = 
\int_0^{t_1} \rho(nt) \cdot e^{-\beta \tau} \, d\tau + e^{-\beta t_1} (r(ht, \text{high}) + r(\text{lost}, \alpha)) + 
\int_{t_1}^{t_1+t_2} \rho(nt) \cdot e^{-\beta \tau} \, d\tau.
\]

The optimal expected cumulative discounted reward (or just discounted reward) in \(M\) with discount rate \(\beta\) is:

\[
dR^{\text{opt}}_{M,\beta} := \text{opt} \left\{ \lim_{N \to \infty} \mathbb{E}_{\sigma}[\text{rew}_N^{M,\beta}] \right\} = \text{opt} \left\{ \lim_{N \to \infty} \int_{\text{Paths}_M} \text{rew}_N^{M,\beta}(\pi) \cdot \Pr_{M,\sigma}[^{\text{d} \pi}] \right\},
\]

where \(\text{opt} \in \{\sup, \inf\}\). A scheduler \(\sigma\) is called optimal for \(dR^{\text{opt}}_{M,\beta}\), if it attains optimum in this equation. In the following, \(dR^{\text{opt}}_{M,\beta}(s)\) denotes the discounted reward collected in \(M\) assuming that the initial state is \(s\). Whenever it is clear from the context, we omit the subscripts and use notation \(dR^{\text{opt}}\).

**Lemma 1.** The value \(dR^{\text{opt}}_{M,\beta}\) exists.

### 3.2 Relation to Discounted Rewards on CTMDP

We now show that for each MRA there exists a (possibly exponentially larger) CTMDP that preserves the discounted reward property. We first need to introduce uniform and normalised MRA:

**Definition 4.** An MRA \(M\) is uniform if \(\exists \eta \in \mathbb{R}_{>0}, s.t. \forall s \in MS_M : E(s) = \eta\).

**Definition 5.** An MRA \(M\) is called normalised if

1. the initial state of \(M\) is probabilistic;
2. every Markovian state \(s\) has only probabilistic predecessors: \(\text{pred}(s) \subseteq PS_M\);
3. probabilistic states of \(M\) have either only probabilistic or only Markovian predecessors: \(\forall s \in PS_M : \text{pred}(s) \subseteq PS_M \lor \text{pred}(s) \subseteq MS_M\).

**Lemma 2.** For any MRA \(M, \eta \geq \max_{s \in S} E(s)\) there exists a uniform normalised MRA \(\overline{M}_\eta\), s.t. \(dR^{\text{opt}}_{\overline{M}_\eta,\beta} = dR^{\text{opt}}_{M,\beta}\) and its size is linear in the size of \(M\).
Informally, \( \overline{\mathcal{M}}_\eta \) is obtained by first uniformising the Markovian states. This is performed via the well-known approach from [17] by adding self-loop transitions to them. Then this uniform MRA is normalised by introducing probabilistic states of zero reward (i) in between each pair of states that violate Properties 2 or 3 of the definition above, or (ii) as a new initial state detailed in Appendix A.3.

In the following, we assume that the MRA at hand is uniform and normalised and show how to construct a value-preserving CTMDP for it. Before proceeding we need to introduce some notation:

- \( \Pi(s, s') \) is the set of all un timed paths \( \pi = s \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots s_k \xrightarrow{a_k} s' \) (paths of \( \mathcal{M} \) with abstracted timing information), such that \( \forall i \in \{1..k \}, \ s_i \notin B \);
- \( PS\setminus B(s) \) is the set of states containing \( s \) and all states \( s' \in PS \setminus B \) that are related to \( s \) via the transitive closure of relation \( \Rightarrow \);
- \( P[\pi] := \prod_{i=1}^{\left|\pi\right|-1} P[\pi[i], a[i], \pi[i+1]], \ r(\pi) := \sum_{i=0}^{\left|\pi\right|-1} r(\pi[i], a[i]), \ \rho(\pi) := \sum_{i=0}^{\left|\pi\right|-1} \rho(\pi[i]). \)

### Value-Preserving CTMDP.

Let \( \mathcal{M} = (S, s_{init}, Act, \xrightarrow{\cdot}, \xrightarrow{\cdot}, r, \rho) \) be a uniform normalised MRA with exit rate \( \eta \). We define the CTMDP \( C(\mathcal{M}) := (S_C, s_{init}, Act_C, R_C) \) and reward structure \( (\rho_C, R_C) \) as follows:

- **SC**: The state space of \( C(\mathcal{M}) \) is the set \( S_C \subseteq PS \) that contains the initial state \( s_{init} \) and all probabilistic states of \( \mathcal{M} \) that are successors of a Markovian state in \( \mathcal{M} \): \( S_C = \{ s \in PS | s = s_{init} \text{ or } \exists s' \in MS : s' \xrightarrow{\lambda} s \} \). We define the set of marked states as \( S_{\text{mark}} := S_C \).

- **Actc**: An action of a state \( s \) in this CTMDP is a mapping \( A : PS\setminus S_{\text{mark}}(s) \to Act \), such that \( A(s') \in Act(s') \). Then the set of all enabled actions \( Act_C(s) \) is the set of all possible functions \( A \), and \( Act_C = \bigcup_{s \in S_{\text{mark}}} Act_C(s) \).

- **Rc**: Let \( s, s' \in S_{\text{mark}} \) and \( \Pi(s, A, s') \subseteq \Pi(s, s') \) be the set of all paths from \( \Pi(s, s') \) that select actions according to \( A \), i.e., for each path \( \pi : \pi[i] \in PS \Rightarrow a[i] = A(\pi[i]) \). Then \( R_C(s, A, s') := \eta \cdot \sum_{\pi \in \Pi(s, s')} P[\pi] \).

- **Rc**: The state reward of a state \( s \) in \( C(\mathcal{M}) \) is the expected state reward gathered in \( \mathcal{M} \) on paths between \( s \) and any other \( s' \in S_{\text{mark}} \) that is a successor of \( s \) in \( C(\mathcal{M}) \): \( \rho_C(s) := \sum_{s' \in S_{\text{mark}}} \sum_{\pi \in \Pi(s, s')} \rho(\pi) \cdot P[\pi] \).

- **Rc**: The transition reward of a state \( s \) and action \( A \) in \( C(\mathcal{M}) \) is the expected transition reward gathered in \( \mathcal{M} \) on paths between \( s \) and any other \( s' \in S_{\text{mark}} \) that is a successor of \( s \) in \( C(\mathcal{M}) \): \( r_C(s, A) := \sum_{s' \in S_{\text{mark}}} \sum_{\pi \in \Pi(s, A, s')} r(\pi) \cdot P[\pi] \).

The main idea of this construction is to lump together states that are all entered at the same time point. For example, in a sequence of probabilistic states followed by a Markovian state all of the states of the sequence will be entered at the same time due to the fact that probabilistic states are left instantaneously upon entry. The construction is similar in spirit to the construction of a value-preserving CTMDP for the long-run average reward problem from [5]. It differs, however, in the treatment of Markovian and probabilistic states due to the fact that timing information effects collected rewards and thus has to be preserved.
Fig. 2: An example of $C(M)$ (on the right) for an MRA $M$ (on the left). We omitted the probabilities of the probabilistic transitions of $M$. Here $A_0 = [p_0 \rightarrow \alpha_0, p_2 \rightarrow \gamma_2, p_3 \rightarrow \gamma_3]$ and other actions of $C(M)$ are constructed analogously. If all the probabilistic distributions are uniform, then $R_C(p_0, A_0, p_4) = \eta \cdot [0.5 \cdot 0.5 \cdot 1 + 0.5 \cdot 0.5 \cdot 1] = 0.5 \cdot \eta$.

An example of this transformation is depicted in Fig. 2. One can easily see that even in small examples the amount of transitions of $C(M)$ can grow extremely fast. Let $s \in S_{mrk}$ and $|PS \setminus S_{mrk}(s)| = n$. If every probabilistic state of $M$ has two enabled actions, then the set of enabled actions of $s$ in $C(M)$ is $2^n$. This growth is therefore exponential in the worst case.

**Theorem 1.** For any uniform normalised MRA $M$ we have $d_{R_{M, \beta}}^{opt} = d_{R_{C(M), \beta}}^{opt}$ and there is an optimal scheduler for $M$ that is stationary.

## 4 Bellman Equation

In this section, we introduce the Bellman equation for the discounted reward problem on MRA.

First of all, due to the results obtained in Sect. 3.2, one could obtain the Bellman equation for an MRA by constructing the value-preserving CTMDP and using the Bellman equation for this CTMDP. However, since the construction of $C(M)$ is exponential in the size of $M$, using the thus obtained Bellman equation for quantifying $d_{R_{C(M), \beta}}^{opt}$ is not efficient for general MRA.

First we informally discuss the reasons why using this method would be inefficient for $M$. First of all, in order to use this approach one would need to construct $C(M)$, which may require exponentially many computations. Additionally, having constructed $C(M)$, the solution of its Bellman equation requires computing an extremum of an operator $F_{C(M)}$ over all enabled actions: $V^* := \text{opt}_{A \in Act_C} F_{C(M)}(A)$. The definition of $F_{C(M)}$ is irrelevant for the current discussion. Since the number of enabled actions in $C(M)$ is in the worst case exponential in the size of $M$, this operation is essentially a brute-force check over exponentially many options. However, we can show that this optimisation problem on $C(M)$ when mirrored back to $M$ itself reduces to the computation of the expected total reward $tR_{D(M)}^{opt}$.

---

2 Here $d_{R_{C,G}}^{opt}$ denotes discounted reward on a CTMDP $C$.

3 For details we refer to Appendix A.4.
on a discrete-time Markov decision process $\mathcal{D}(\mathcal{M})$, whose size is linear in the size of $\mathcal{M}$. Computing $tR_{\mathcal{D}(\mathcal{M})}^{opt}$ is a well-studied problem on MDPs that admits an efficient solution via dynamic programming. Thus instead of naively brute-forcing $\sup_{A \in Act_{\mathcal{C}}} F_{\mathcal{C}(\mathcal{M})}(A)$, the value $V^*$ can be efficiently computed by well-known dynamic programming techniques for $tR_{\mathcal{D}(\mathcal{M})}^{opt}$ [3,18]. To formalise this result we need to introduce the expected total reward $tR_{\mathcal{D}(\mathcal{M})}^{opt}$ on MDPs, as defined in [18].

**Expected total reward.** Let $\mathcal{D}$ be a (discrete-time) MDP and $X^*_i, Y^*_i$ be random variables denoting the state occupied by $\mathcal{D}$ and the action chosen at step $i$ starting from state $s$. Then the value

$$tR_{\mathcal{D},r}(s) := \lim_{N \to \infty} \mathbb{E}_{s,s^\prime} \left[ \sum_{i=0}^{N-1} r_{\mathcal{D}}(X^*_i, Y^*_i) \right],$$

where $\mathbb{E} \in \{\sup, \inf\}$, denotes the optimal expected total reward on $\mathcal{D}$ with reward structure $r_{\mathcal{D}}$, starting from state $s$.

**Terminal MDP.** We now construct the discrete MDP and the reward structure on it that enables us to substitute the naive brute-force approach with the efficient computation of the expected total reward.

Let $\mathcal{M}$ be a uniform normalised MRA. Informally, we keep the structure of $\mathcal{M}$, but for each marked state $s \in S_{\text{mrk}}$, we introduce a copy state $s_{\text{cp}}$ and redirect all the transitions leading to $s$ to the new copy state $s_{\text{cp}}$. These copy states have only transitions with probability 1 to a new terminal state $t$. Formally, the terminal MDP of $\mathcal{M}$ is $\mathcal{D}(\mathcal{M}) := (S_{\mathcal{D}}, s_{\text{init}}, Act_{\mathcal{D}}, P_{\mathcal{D}})$, where $S_{\text{cp}} = \{s_{\text{cp}} \mid s \in S_{\text{mrk}}\}$, $S_{\mathcal{D}} = S \cup S_{\text{cp}} \cup \{t\}$ and

$$P_{\mathcal{D}}[s, \alpha, s'] = \begin{cases} P[s, \alpha, s'] & \text{for } s \in PS, s' \notin S_{\text{cp}}, \\ P[s, \alpha] & \text{for } s \in MS, s' = p_{\text{cp}} \in S_{\text{cp}}, \alpha = \bot, \\ 1 & \text{for } (s \in S_{\text{cp}} \text{ or } s = t), s' = t, \alpha = \bot, \\ 0 & \text{otherwise.} \end{cases}$$

Figure 3(b) depicts the terminal MDP for the MRA from Fig. 1.

We can now present an efficient characterisation of the discounted reward on MRA. Let $\mathcal{D}(\mathcal{M}) = (S_{\mathcal{D}}, s_{\text{init}}, Act_{\mathcal{D}}, P_{\mathcal{D}})$ be the terminal MDP of $\mathcal{M}$ and $h : S_{\text{mrk}} \to \mathbb{R}$. We define a reward structure $r_{\mathcal{D}(\mathcal{M}),h}$ for $\mathcal{D}(\mathcal{M})$ as follows:

$$r_{\mathcal{D}(\mathcal{M}),h}(s, \alpha) := \begin{cases} r(s, \alpha) & \text{for } s \in PS, \alpha \in Act_{\mathcal{D}}(s), \\ \rho(s) \frac{\beta}{\beta+\eta} & \text{for } s \in MS, \alpha = \bot, \\ \frac{\eta}{\beta+\eta} h(s) & \text{for } s \in S_{\text{cp}}, \alpha = \bot, \\ 0 & \text{for } s = t, \alpha = \bot. \end{cases}$$
Fig. 3: Figure (a) depicts the uniform normalised MRA from Fig. 1 and (b) shows the corresponding terminal MDP. All the added/updated transitions and states are highlighted in green color. The figure omits Dirac distributions and rewards.

**Theorem 2.** Let $\mathcal{M}$ be a uniform normalised MRA with exit rate $\eta$ and $\mathcal{D}(\mathcal{M})$ the corresponding terminal MDP. Then the vector $\mathbf{d}R^\text{opt}_{\mathcal{M},\beta} := (\mathbf{d}R^\text{opt}_{\mathcal{M},\beta}(s)), \forall s \in S_{\text{mark}}$, is the unique solution to the Bellman equation:

$$\forall s \in S_{\text{mark}} : \, v(s) = tR^\text{opt}_{\mathcal{D}(\mathcal{M}), \text{rew}_\mathcal{D}(\mathcal{M}), v}(s).$$

The reason for this characterisation being efficient is the right-hand side of Equation 1. Quantification of the expected total reward on MDPs is a well-established problem that admits such algorithms as policy-iteration and linear programming [18]. Moreover, for a subclass of models it can be solved in time linear in the size of the MRA. Those are models that have no cycles consisting of only probabilistic states. Such cycles (even non-Zeno ones) almost never happen in real-world applications and are usually considered a modelling mistake. In fact, we are not aware of any practical example where that case occurs.

5 Numerical Solution

Having a Bellman equation at hand, one can normally derive three types of algorithms: based on value-iteration, policy-iteration, and linear programming. Due to scalability issues of the latter we do not consider it and present here only value- and policy-iteration algorithms.

In the following let $\text{sp}(v) := \left[ \max_{s \in S_{\text{mark}}} v(s) - \min_{s \in S_{\text{mark}}} v(s) \right]$. Additionally, we denote with $\text{ExpectedTotalReward}(\mathcal{D}, \text{rew}, \sigma)$ the function that computes the expected total reward on an MDP $\mathcal{D}$ for reward structure $\text{rew}$. The last parameter $\sigma$ can be either a stationary deterministic scheduler or one of $\{\text{sup}, \text{inf}\}$. In the latter case, the optimal expected total reward is computed and in the former the expected total reward for scheduler $\sigma$.

Algorithms [1] and [2] are value- and modified policy-iteration algorithms that compute the value $\mathbf{d}R^\text{opt}_{\mathcal{M},\beta}$ for an arbitrary MRA $\mathcal{M}$ and discounting rate $\beta > 0$. 
Algorithm 1: ValueIteration

```
input : MRA \( \mathcal{M} = (S, s_{init}, Act, \rightarrow, \leftarrow, r, \rho) \), opt \( \in \{\text{sup}, \text{inf}\} \), \( \beta > 0 \), approximation error \( \varepsilon > 0 \)
output : \( v \) such that \( \|v - d_{R_{\mathcal{M}, \beta}}\| < \varepsilon \), and the \( \varepsilon \)-optimal scheduler \( \sigma \)
1 \( \mathcal{M}_\eta := \text{normalise}(\text{uniformise}(\mathcal{M}, \text{rate } \eta := \max_{s \in S} E(s))) \);
2 \( \mathcal{D}(\mathcal{M}_\eta) := \text{terminal MDP for } \mathcal{M}_\eta \);
3 \( v_0 := 0 \);
4 for \( (n := 0; sp(v_{n+1} - v_n) < \varepsilon \beta ; n++) \) do
5 \( (v_{n+1}, \sigma) := \text{ExpectedTotalReward}(\mathcal{D}(\mathcal{M}_\eta), \text{rew}_{\mathcal{D}(\mathcal{M}_\eta)}, v_n, \text{opt}) \);
6 return \( v_{n+1}(s_{init}), \sigma \);
```

The standard policy-iteration algorithm in which the policy evaluation step is performed exactly is also possible in the setting of MRA. However, this requires the exact solution of a linear equation system with one variable per state of \( \mathcal{M} \). Since this is an expensive operation for hundreds of thousands of states, we choose the modified policy-iteration instead. The latter bypasses this issue by performing the policy evaluation step numerically. The algorithm depends on a sequence of natural numbers called order sequence \( (m_n)_{n \in \mathbb{N}_0} \); it converges however for an arbitrary sequence.

**Theorem 3.** Algorithms 1 and 2 are sound and complete.

Algorithms 1 and 2 are essentially the respective algorithms on CTMDPs [18], in which the extremum value over enabled actions is searched through the solution of the expected total reward problem \( tR_{\mathcal{D}(\mathcal{M})}^{\text{opt}} \) on the terminal MDP \( \mathcal{D}(\mathcal{M}) \). Therefore in both algorithms the complexity of an iteration equals the complexity of computing the value \( tR_{\mathcal{D}(\mathcal{M})}^{\text{opt}} \) (which is polynomial), and the convergence rate is the same as the convergence rate of the corresponding CTMDP algorithms.

**Computation of the expected total reward.** Notice that the presented algorithms are guaranteed to converge whenever the expected total reward of the terminal MDP is computed precisely. Exact quantification of this value can be achieved with policy-iteration or linear programming algorithms [18]. Moreover, for models that have no cycles and consist of only probabilistic states, the expected total reward can be solved efficiently in time \( O(|\rightarrow| + |\leftarrow|) \).

### 6 Experiments

Here we present the empirical evaluation of the discussed algorithms. Both algorithms were implemented as part of the IMCA/MAMA toolset [9]. All experiments were run on a single core of Intel Core i7-4790 with 8 GB of RAM.
Algorithm 2: ModifiedPolicyIteration

input: a model \( M \) of a Markov decision process (MDP) with states \( S \), initial state \( s_{\text{init}} \), actions \( A \), transition \( \rightarrow \), reward \( \rightarrow \), \( \beta > 0 \), \( \varepsilon > 0 \), and a sequence \( (m_n)_{n \geq 0} \) in \( \mathbb{N} \); a parameter \( \text{opt} \) \( \in \{\sup, \inf\} \); and a discount factor \( \beta > 0 \).

output: a value function \( v \) such that \( \|v - dR_{\text{opt}}^\beta\| < \varepsilon \) and the \( \varepsilon \)-optimal scheduler \( \sigma \).

1. Compute the uniformised MDP \( \overline{M}_\eta := \text{normalise}(\text{uniformise}(M, \text{rate } \eta \leftarrow \max_{s \in S} E(s))) \).
2. Define the terminal MDP \( D(\overline{M}_\eta) := \text{terminal MDP for } \overline{M}_\eta \).
3. Set \( v_0 := 0 \).
4. Set \( \text{stop} := \text{false} \); \( n := 0 \).
5. while (\( \neg \text{stop} \)) do
6.   // Policy improvement
   \( (u_n^0, \sigma_{n+1}) := \text{ExpectedTotalReward}(D(\overline{M}_\eta), \text{rew}_{D(\overline{M}_\eta)}, v_n, \text{opt}) \).
7.   // Partial policy evaluation
8.   if \( sp(u_n^0 - v_n) < \frac{\varepsilon \beta}{\eta} \) then
9.      \( \text{stop} := \text{true}; \text{break}; \)
10.     for \( k := 0; k < m_n; k++ \) do
11.        \( u_{n+k} := \text{ExpectedTotalReward}(D(\overline{M}_\eta), \text{rew}_{D(\overline{M}_\eta)}, v_n, \sigma_{n+1}) \).
12.     \( v_{n+1} := u_{m_n} \).
13.     \( n := n + 1 \).
14. return \( u_n^0(s_{\text{init}}), \sigma_{n+1} \).

Table 1: Parameters of some of the benchmarks.

<table>
<thead>
<tr>
<th>Model</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(</td>
<td>(</td>
<td>(</td>
<td>(</td>
<td>(</td>
<td>(</td>
</tr>
<tr>
<td>FTWC-resp-50-40</td>
<td>92,819</td>
<td>20,806</td>
<td>72,013</td>
<td>72,007</td>
<td>305,613</td>
<td>6.35</td>
</tr>
<tr>
<td>PS-256-3-4</td>
<td>131,529</td>
<td>87,605</td>
<td>43,924</td>
<td>189,129</td>
<td>72,965</td>
<td>3</td>
</tr>
<tr>
<td>QS-256-256</td>
<td>465,177</td>
<td>398,096</td>
<td>67,081</td>
<td>530,966</td>
<td>200,206</td>
<td>2</td>
</tr>
</tbody>
</table>

Benchmarks. We have evaluated our approach on a collection of published benchmark models: the Polling System [9, 20], Queuing System [12], and the Fault Tolerant Workstation Cluster [15]. Discounting for the selected benchmarks naturally models the decrease of the value of costs over time. In order to address a case study with a specific set of parameters we use the same notation as in [5]. We used the tool SCOOP [19] to generate those models and for this reason the degree of variation of some parameters is restricted by its runtime-space requirements.

Table 1 shows the parameters of some of the used models. We use the symbols \( \lambda^M \) to denote the maximal number of enabled actions in probabilistic states of \( M \), and \( E(M) \) shows the maximal exit rate of Markovian states of \( M \).

Empirical Evaluation. The space complexity of both algorithms is polynomial. Therefore, we have evaluated the effect of varying model size, precision, and the discounting rate on their runtime only. In plots, whenever the experiment covers several benchmarks, we use the symbol “X” to denote respective part of...
the model name, e.g. PS-2-X. In this section we will refer to the value-iteration algorithm [1] as VI and to the modified policy-iteration algorithm [2] as MPI.

As we have mentioned in Sect. [5] the modified policy-iteration algorithm depends on a parameter called order sequence. The optimal choice of the order sequence is an open question [18]. In this section, we present the best results we could achieve with different order sequences. Let us notice that if every element of the order sequence is 0 then MPI is the same as VI. When the values grow infinitely large, the algorithm turns into standard policy-iteration. Almost always in our experiments we could find an order sequence that led to lower running times than those of VI. Moreover, relatively small order sequences achieved the best value, e.g. \( m_n = 100 \), \( \forall n > 0 \) was sufficient for most of the models. Selecting a sequence with larger values, however, quite often led to running times significantly worse than those of VI.
Fig. 6: Observed dependency on discounting rate $\beta$ in log-log scale. Here $\varepsilon = 10^{-8}$.

Model size. Figure 4 shows the dependency of the running time of both algorithms on the size of the state space. Both algorithms exhibit polynomial dependency (linear in log-log scale) on the depicted size range. This agrees with theoretical expectations since the computation of expected total rewards is polynomial in the size of the state space and the convergence rate does not depend on the model size.

Precision. Figure 5 shows the dependency of the computation time on the precision parameter $\varepsilon$. The theoretical convergence rate of both algorithms resembles that of respective algorithms on CTMDPs. The expected complexity of VI is logarithmic in $\varepsilon$, which is supported by the observed results. Regarding MPI, we observed that the function of the running time repeats that of VI, possibly due to the relatively small values used for the order sequence.

Discounting rate. Figure 6 depicts the dependency of the running time of the algorithms on the discounting rate $\beta$. The observed dependency of VI follows the theoretical bound of $O\left(\frac{1}{1-\beta}\right)$. Similarly to the previous case, the function of the running time of policy-iteration repeats that of value iteration.

7 Conclusion

While discounting is a standard concept on Markov chains and Markov decision processes, this is the first paper to consider discounting for the more general model of Markov reward automata. We have discussed that computing discounted rewards on MRA can be reduced to the same task on a possibly exponentially larger CTMDP. Constructing and optimizing over this large CTMDP can be avoided by recognizing the essential computation as determining the expected total reward in a linear-sized discrete-time MDP. This in turn is a well-understood problem enabling an efficient solution. Experiments clearly demonstrate the efficiency of our approach, being able to handle MRAs with hundred thousands of states.
References

A Appendix

A.1 Proof of Lemma 1

Let $\mathcal{M} = (S, s_{\text{init}}, Act, \rightarrow, \sim, r, \rho)$ be a uniform normalised MRA. In the following we will use the following abbreviations:

- $\mathbb{I}_S(s) := 1$ if $s \in S$ and 0 otherwise.
- For a finite path $\pi = s_0 \xrightarrow{\alpha_1, t_0} s_1 \xrightarrow{\alpha_2, t_1} \cdots s_n$, $t[\pi, i] := t_i$ ($\pi$ is omitted when it is clear from the context) and $T(\pi) := \sum_{i=0}^{[\pi]-1} t[i]$.

Lemma 1. The value $dR^\text{opt}_{\mathcal{M}, \beta}$ exists.

Proof. We first show that for every scheduler $\sigma \in GM$ the value $dR^\sigma_{\mathcal{M}, \beta}$ exists. By definition:

$$dR^\sigma_{\mathcal{M}, \beta} = \lim_{N \to \infty} E_\sigma[\text{rew}_N^{\mathcal{M}, \beta}(\pi)]$$

$$= \lim_{N \to \infty} E_\sigma \left[ \sum_{k=0}^{N-1} \left[ e^{-\beta \cdot \tau[k]} \cdot r(\pi[k], \alpha[k]) + \int_{\tau[k]}^{\tau[k]+t[k]} e^{-\beta \cdot t} \cdot \rho(\pi[k]) \, dt \right] \right].$$

Let $\pi' \in \text{Paths}_\mathcal{M}$, such that $\pi' \downarrow \in \text{PS}_\mathcal{M}$. We denote with $E_\sigma, MS_{\mathcal{M}}[\text{rew}_N^{\mathcal{M}, \beta} | \pi']$ the expected discounted reward collected starting from state $\pi'$ until encountering a Markovian state $s$ and following the scheduler $\sigma$ that assumes that path $\pi'$ has been taken until reaching $s$. Then

$$E_\sigma, MS_{\mathcal{M}}[\text{rew}_N^{\mathcal{M}, \beta} | \pi'] = E_\sigma, MS_{\mathcal{M}} \left[ \sum_{k=0}^{\infty} e^{-\beta \cdot \tau[k]} \cdot r(\pi[k], \alpha[k]) \mid \pi' \right]$$

$$= E_\sigma, MS_{\mathcal{M}} \left[ \sum_{k=0}^{\infty} e^{-\beta \cdot \tau[\pi'[k]]} \cdot r(\pi[k], \alpha[k]) \mid \pi' \right]$$

$$= e^{-\beta \cdot \tau[\pi'[\downarrow]]} \cdot E_\sigma, MS_{\mathcal{M}} \left[ \sum_{k=0}^{\infty} r(\pi[k], \alpha[k]) \mid \pi' \right].$$

The value $E_\sigma, MS_{\mathcal{M}} \left[ \sum_{k=0}^{\infty} r(\pi[k], \alpha[k]) \mid \pi' \right]$ is the total expected reward gathered in the MDP $D_s$ formed by probabilistic states reachable from $s \sim \rightarrow$. Where the reward value of states of this MDP $r_D(s, \alpha) := r(s, \alpha)$.

Since $\mathcal{M}$ is non-Zeno, all the probabilistic states are transient. This is a sufficient condition [18] for the value $E_\sigma, MS_{\mathcal{M}} \left[ \sum_{k=0}^{\infty} r(\pi[k], \alpha[k]) \mid \pi' \right]$ to exist and be finite. Therefore there exists a value $M$, such that

$$E_\sigma, MS_{\mathcal{M}}[\text{rew}_N^{\mathcal{M}, \beta} | \pi'] \leq e^{-\beta \cdot \tau[\pi'[\downarrow]]} \cdot M.$$
Therefore

\[ dR_{M,\beta}^\sigma = \lim_{N \to \infty} E_{\sigma} \left[ \sum_{k=0}^{N-1} e^{-\beta \cdot \tau[k]} \cdot r(\pi[k], \alpha[k]) + \int_{\tau[k]}^{\tau[k]+t[k]} e^{-\beta \cdot t} \cdot \rho(\pi[k]) \, dt \right] \]

\[ = \lim_{N \to \infty} \sum_{k=0}^{N-1} \left( \sum_{s\in MS_M} E_{s,\sigma} \left[ \int_{\tau[k]}^{\tau[k]+t[k]} e^{-\beta \cdot t} \cdot \rho(s) \, dt \right] + \sum_{s\in PS_M} E_{s,\sigma} \left[ e^{-\beta \cdot \tau[k]} \cdot r(s, \alpha[k]) \right] \right) \]

\[ \leq \lim_{N \to \infty} \sum_{k=0}^{N-1} \left( \sum_{s\in MS_M} E_{s,\sigma} \left[ \int_{\tau[k]}^{\tau[k]+t[k]} e^{-\beta \cdot t} \cdot \rho(s) \, dt \right] \right) \]

\[ \leq \lim_{N \to \infty} \sum_{k=0}^{N-1} \left( \sum_{s\in MS_M} E_{s,\sigma} \left[ \int_{\tau[k]}^{\tau[k]+t[k]} e^{-\beta \cdot t} \cdot \rho(s) \, dt \right] \right) \]

\[ = \lim_{N \to \infty} \sum_{k=0}^{N-1} \sum_{s\in MS_M} E_{s,\sigma} \left[ \int_{\tau[k]}^{\tau[k]+t[k]} e^{-\beta \cdot t} \cdot \rho(s) \, dt + e^{-\beta \cdot \tau[k]} \cdot M \right] \]

\[ = \lim_{N \to \infty} E_{\sigma} \left[ \sum_{k=0}^{N-1} 1_{MS_M}(\pi[k]) \left( \int_{\tau[k]}^{\tau[k]+t[k]} e^{-\beta \cdot t} \cdot \rho(\pi[k]) \, dt + e^{-\beta \cdot \tau[k]} \cdot M \right) \right] \]

\[ = \lim_{N \to \infty} E_{\sigma} \left[ \sum_{k=0}^{N-1} 1_{MS_M}(\pi[k]) \cdot e^{-\beta \cdot \tau[k]} \cdot \left( \int_{0}^{t[k]} e^{-\beta \cdot t} \cdot \rho(\pi[k]) \, dt + M \right) \right] \]

\[ = \lim_{N \to \infty} E_{\sigma} \left[ \sum_{k=0}^{N-1} 1_{MS_M}(\pi[k]) \cdot e^{-\beta \cdot \tau[k]} \cdot \left( \frac{\rho(\pi[k])}{\beta} + E(\pi[k]) + M \right) \right] \]

\[ \leq \lim_{N \to \infty} E_{\sigma} \left[ \sum_{k=0}^{N-1} 1_{MS_M}(\pi[k]) \cdot e^{-\beta \cdot \tau[k]} \cdot C \right] \]

\[ \leq \lim_{N \to \infty} C \cdot E_{\sigma} \left[ \int_{0}^{\infty} e^{-\beta \cdot t} \, dt \right] \]

\[ \leq \frac{C}{\beta} . \]
Therefore

\[ dR_{\mathcal{M},\beta}^{opt} = \sup_{\sigma \in \Gamma \mathcal{M}} \{ dR_{\mathcal{M},\beta}^{\sigma} \} \leq \frac{C}{\beta} \]

and the value \( dR_{\mathcal{M},\beta}^{opt} \) exists and is finite. \( \square \)

### A.2 Proof of Theorem 1

Let \( \mathcal{M} = (S, s_{init}, Act, \rightarrow, \leftarrow, r, \rho) \) be a uniform normalised MRA and \( \mathcal{C}(\mathcal{M}) = (S_{mrk}, s_{init}, Act_{\mathcal{C}}, R_{\mathcal{C}}) \) the CTMDP obtained as shown in Section 3.2. In the following we will denote paths of \( \mathcal{M} \) with symbol \( \pi \) and paths of \( \mathcal{C} \) with symbol \( \varphi \). The following is the list of abbreviations that we will use for this proof:

- Let \( \pi \in \text{Paths}^*_\mathcal{M} \cup \text{Paths}^*_\mathcal{C} \). Then \( \pi_C[i] = s \), s.t. \( s \in S_{mrk} \) and \( s \) is the \( i \)-th state of \( \pi \) that belongs to \( S_{mrk} \).
- \( \text{Paths}^N_{\mathcal{M}}(s) \) denotes the set of finite paths in \( \mathcal{M} \) that have exactly \( N \) positions \( i \) such that \( \pi[i] \in S_{mrk} \).
- \( \text{rew}^{T,N}_{\mathcal{M},\beta}(\pi) := \text{rew}^{T,K}_{\mathcal{M},\beta}(\pi) \), where \( K \) is the length of the prefix of \( \pi \) that contains exactly \( N \) states from \( S_{mrk} \).
- Let \( \varphi \in \text{Paths}^*_\mathcal{C} \), then \( \text{Paths}^*_\mathcal{M}(\varphi) \) denotes all finite paths \( \pi \) in \( \mathcal{M} \), such that \( \forall i : \pi_C[i] = \varphi[i] \).
- Let \( \pi, \pi' \) be two finite paths, such that \( \pi \downarrow = \pi'[0] \), then \( \pi \cdot \pi' \) denotes the concatenation of the two paths.
- Let \( \Pi \) be a set of paths (finite or infinite), then \( \text{ta}(\Pi) := \{ \pi = s_0 \overset{\alpha_0}{\rightarrow} s_1 \overset{\alpha_1}{\rightarrow} \cdots | s_0 \overset{\alpha_0}{\rightarrow} s_1 \overset{\alpha_1}{\rightarrow} \cdots \in \Pi \} \).

We will now redefine the reward collected over a path to take into account a time shift. Let \( T > 0 \), then \( \text{rew}^{T,0}_{\mathcal{M},\beta}(\pi) := 0 \), and:

\[
\text{rew}^{T,N}_{\mathcal{M},\beta}(\pi) := \sum_{k=0}^{N-1} \left[ e^{-\beta \cdot \tau[k]+T} \cdot r_{\mathcal{M}}(\pi[k], \alpha[k]) + \int_{\tau[k]+T}^{\tau[k]+T+\tau[k]+T} e^{-\beta \cdot t} \cdot \rho_{\mathcal{M}}(\pi[k]) \, dt \right].
\]

Obviously \( \text{rew}^{N}_{\mathcal{M},\beta}(\pi) = \text{rew}^{0,N}_{\mathcal{M},\beta}(\pi) \). We additionally redefine the discounted reward to take into account a history \( \pi \in \text{Paths}^*_\mathcal{M} \):

\[
dR_{M,\beta}(\pi, N) := \int_{\text{Paths}_{\mathcal{M}}} \text{rew}_{M,\beta}^{T}(\pi')(N) \cdot \Pr_{\sigma}[d\pi'],
\]

Obviously \( dR_{M,\beta}(N) = dR_{M,\beta}(\pi, N) \). And analogously we redefine respective values for CTMDPs.

We define \( dR_{\mathcal{M}}(\pi, N(\mathcal{C})) := \int_{\text{Paths}_{\mathcal{M}}} \text{rew}_{\mathcal{M}}^{T}(\pi')(N) \cdot \Pr_{\sigma}[d\pi'] \). Let \( \sigma \) be a scheduler in \( \mathcal{M} \). We define scheduler \( \delta \) in \( \mathcal{C} \) as follows:
\[ \delta(\varphi, A) := \int_{\text{Paths}_A(\varphi)} \prod_{s \in P_\text{mark}(\varphi)} \sum_{\pi' \in H_\text{mark}(\varphi, s)} \sigma(\pi \cdot \pi', A(s)) \, d\pi. \]

Let \( \pi, \pi' \in \text{Paths}_A^* \), then \( \sigma(\pi', \alpha) := \sigma(\pi \cdot \pi', \alpha) \). And analogously for \( \delta_\varphi \).

**Lemma 3.** Let \( \varphi \in \text{Paths}_C^* \) and \( \pi \in \text{Paths}_A^* \), then

\[
\begin{align*}
\int_{\text{Paths}_A(\varphi)} \Pr_\sigma [\, d\pi] & \int_{\text{Paths}_A^*(\pi \downarrow)} \text{rew}^{T(\pi), 1(C)}(\pi') \cdot \Pr_\sigma [\, d\pi'] = \\
\int_{\text{Paths}_A^*(\varphi)} \text{rew}^{T(\pi), 1(C)}(\pi') \cdot \Pr_\sigma [\, d\pi'] \cdot \Pr_\sigma [\, d\pi'] =
\end{align*}
\]

Proof. Let \( \pi \in \text{Paths}_A^*(\varphi) \) and \( Y_{\varphi, \pi} : \text{Act}_C(\varphi \downarrow) \rightarrow [0, 1] \) - a random variable, such that \( \Pr[Y_{\varphi, \pi} = A] = \prod_{s \in P_\text{mark}(\varphi \downarrow)} \sum_{\pi' \in H_\text{mark}(\varphi, s)} \sigma(\pi \cdot \pi', A(s)) \).

Then

\[
\begin{align*}
\int_{\text{Paths}_A(\varphi)} \sum_{\pi' \in \text{ta}(\text{Paths}_A^*(\pi \downarrow))} \text{rew}^{T(\pi), 1(C)}(\pi') \cdot \Pr_\sigma [\, d\pi'] =
\end{align*}
\]

where \( \pi'(t) \) is the timed path, obtained from the untimed path \( \pi' \in \text{ta}(\text{Paths}_A^*(\pi \downarrow)) \) by assigning sojourn time to the only Markovian state of \( \pi' \), and sojourn time 0 to all the rest states of the path. Consider he inner summation of the last expression:

\[
\begin{align*}
\sum_{\pi' \in \text{ta}(\text{Paths}_A^*(\pi \downarrow))} \text{rew}^{T(\pi), 1(C)}(\pi'(t)) \cdot \Pr_\sigma [\, d\pi'] =
\end{align*}
\]
where \( 1_{\pi}(A) = 1 \) if \( \forall i : \alpha_{[\pi', i]} = A(\pi'[i]) \) and 0 otherwise. The set of paths time abstract paths \( \pi' \) on which \( 1_{\pi}(A) = 1 \) we denote as \( \text{ta}(\text{Paths}^1_{\mathcal{M}}(\pi, A)) \). Then

\[
\sum_{\pi' \in \text{ta}(\text{Paths}^1_{\mathcal{M}}(\pi, A))} \text{rew}^T_{\mathcal{M}}(\pi, 1)(\pi'(t)) \cdot \Pr_{\sigma_\pi}[\pi'] = \\
\sum_{A \in \text{Act}_\mathcal{C} (s)} P[Y_{\varphi, \pi} = A] \sum_{\pi' \in \text{ta}(\text{Paths}^1_{\mathcal{M}}(\pi, A))} \text{rew}^T_{\mathcal{M}}(\pi, 1)(\pi'(t)) \cdot \Pr[\pi'] = \\
\sum_{A \in \text{Act}_\mathcal{C} (s)} P[Y_{\varphi, \pi} = A] \cdot \text{rew}^T_{\mathcal{C}}(s, A, t).
\]

Therefore

\[
\int_{\text{Paths}_\mathcal{M}(\varphi)} \Pr_\sigma[d\pi] \int_{\text{Paths}^1_{\mathcal{M}}(\pi, A)} \text{rew}^T_{\mathcal{M}}(\pi, 1)(\pi') \cdot \Pr_{\sigma_\pi}[d\pi'] = \\
\int_{\text{Paths}_\mathcal{M}(\varphi)} \Pr_\sigma[d\pi] \int_0^\infty dt \sum_{A \in \text{Act}_\mathcal{C} (s)} P[Y_{\varphi, \pi} = A] \cdot \text{rew}^T_{\mathcal{C}}(s, A, t) = \\
\int_0^\infty dt \sum_{A \in \text{Act}_\mathcal{C} (s)} \delta_{\varphi}(A) \cdot \text{rew}^T_{\mathcal{C}}(s, A, t) = \\
\int_{\text{Paths}^1_{\mathcal{C}}(\varphi)} \text{rew}^T_{\mathcal{C}}(s, A, t) \cdot \Pr_{\delta_\varphi}[d\pi'].
\]

\[\square\]

**Theorem 1.** For any uniform normalised MRA \( \mathcal{M} \) we have \( dR^\text{opt}_{\mathcal{C}, \beta} = dR^\text{opt}_{\mathcal{C}(\mathcal{M}), \beta} \) and there is an optimal scheduler for \( \mathcal{M} \) that is stationary.

**Proof.** For convenience in the following we denote the states of the MRA with the symbol \( s \), and states of the CTMDP with \( q \). The initial states of both models coincide and we therefore denote it as \( s_{\text{init}} \).

We now prove by induction over \( N \), that \( \forall \varphi \in \text{Paths}^*_{\mathcal{C}}, \forall N \in \mathbb{N}_{\geq 0} \):

\[
\int_{\varphi \in \text{Paths}^*_{\mathcal{M}(\varphi)}} \text{d}R^\text{opt}_{\mathcal{M}}(\pi, N(C)) \cdot \Pr[d\pi] = \text{d}R^\text{opt}_{\mathcal{C}}(\varphi, N).
\]

\[\footnote{Here \( \text{d}R^\text{opt}_{\mathcal{C}, \beta} \) denotes discounted reward on a CTMDP \( \mathcal{C} \).}\]
– **Induction Base:** Obvious, since \( \forall \pi \in \text{Paths}_{\Lambda}^* : \ dR_{\Lambda}^\sigma(\pi, 0(C)) = dR_{\Lambda}^\sigma(\pi, 0) = 0 \) and \( \forall \varphi : \ dR_{\Lambda}^\sigma(\varphi, 0) = 0; \)

– **Induction Step:** Let \( \varphi \in \text{Paths}_{\Gamma}^* \) and \( \pi \in \text{Paths}_{\Lambda}^*(\varphi) \). Then

\[
dR_{\Lambda}^\sigma(\pi, N(C)) = \int_{\text{Paths}_{\Lambda}^*(\pi)} \text{Pr}_{\sigma_{\pi}}[d\pi'] \times \\
\int_{\text{Paths}_{\Lambda}^*(\pi')} \text{Pr}_{\sigma_{\pi'}\pi'}[d\pi''] \cdot \text{rew}_{\Lambda}^{T(\pi), 1(C)}(\pi'') \cdot \text{rew}_{\Lambda}^{T(\pi), N(C)}(\pi'') \cdot \text{Pr}_{\sigma_{\pi}}[d\pi]
\]

\[
= \int_{\text{Paths}_{\Lambda}^*(\pi)} \text{Pr}_{\sigma_{\pi}}[d\pi'] \cdot \text{rew}_{\Lambda}^{T(\pi), 1(C)}(\pi') \cdot \text{Pr}_{\sigma_{\pi}}[d\pi'] + \\
\int_{\text{Paths}_{\Lambda}^*(\pi)} \text{rew}_{\Lambda}^{T(\pi), N(C)}(\pi', (N - 1)(C)) \cdot \text{Pr}_{\sigma_{\pi}}[d\pi']
\]

Therefore

\[
\int_{\text{Paths}_{\Lambda}^*(\varphi)} \int_{\text{Paths}_{\Lambda}^*(\pi')} \text{rew}_{\Lambda}^{T(\pi), 1(C)}(\pi') \cdot \text{Pr}_{\sigma_{\pi}}[d\pi'] \cdot \text{Pr}_{\sigma}[d\pi] + \\
\int_{\text{Paths}_{\Lambda}^*(\varphi)} \int_{\text{Paths}_{\Lambda}^*(\pi')} \text{rew}_{\Lambda}^{T(\pi), N(C)}(\pi', (N - 1)(C)) \cdot \text{Pr}_{\sigma_{\pi}}[d\pi'] \cdot \text{Pr}_{\sigma}[d\pi].
\]
Let \( q \in S_{\text{mark}}, A_q \in Act_C(q), t > 0 \) and \( \varphi \xrightarrow{A_q,t} q := \varphi \cdot (\pi \downarrow A_q,t q) \). Then

\[
\int_{\text{Paths}_M(\varphi)} dR_{M}(\pi, N(C)) \cdot \Pr_{\sigma}[d\pi] = \int_{\text{Paths}_M(\varphi)} \int_{\text{Paths}_{M,C}(\varphi)} dR_{M}(\pi, (N-1)(C)) \cdot \Pr_{\sigma}[d\pi] + \sum_{q \in S_{\text{mark}}} \sum_{A_q \in Act_C(q)} \int_{0}^{\infty} dt \int_{\text{Paths}_M(\varphi \xrightarrow{A_q,t} q)} dR_{M}(\pi, N-1(C)) \cdot \Pr_{\sigma}[d\pi]
\]

\[= \int_{\text{Paths}_M(\varphi)} \int_{\text{Paths}_{M,C}(\varphi)} \text{rew}_{M}(\varphi \xrightarrow{A_q,t} q) \cdot \Pr_{\sigma}[d\pi'] \cdot \Pr_{\sigma}[d\pi] + \sum_{q \in S_{\text{mark}}} \sum_{A_q \in Act_C(q)} \int_{0}^{\infty} dt dR_{C}(\varphi \xrightarrow{A_q,t} q, N-1) \cdot \Pr_{\sigma}[d\pi]
\]

\[= \int_{\text{Paths}_C(\varphi)} \int_{\text{Paths}_{C}(\varphi')} \text{rew}_{C}(\varphi \xrightarrow{A_q,t} q, N-1) \cdot \Pr_{\sigma}[d\pi] \cdot \Pr_{\sigma}[d\pi'] + \sum_{q \in S_{\text{mark}}} \sum_{A_q \in Act_C(q)} \int_{0}^{\infty} dt dR_{C}(\varphi, N) \cdot \Pr_{\sigma}[d\pi]
\]

Thus, for each scheduler \( \sigma \in GM \) on the MRA \( M \) we built a scheduler \( \delta \) on \( C(M) \), such that \( dR_{M,\beta} = dR_{C,\beta} \). Therefore, since the optimal scheduler for discounted reward on CTMDPs is stationary deterministic [18], the same holds for Markov reward automata and its discounted reward. \( \Box \)

### A.3 Uniformisation and Normalisation

Here we will show how to uniformise and normalise an arbitrary MRA.
Uniformisation. Markovian states of $\mathcal{M}$ may have different exit rates. In order to achieve uniform exit rates we will use a well-known approach [17] by adding self-loop transitions to them. Let $\eta \geq \max_{s \in S} E(s)$. The uniformisation of $\mathcal{M}$ to the rate $\eta$ is the MRA $\mathcal{M}_{\eta} := (S_{\eta}, s_{\text{init}}, Act, \to, \to_{\eta}, r, \rho)$, such that

$$ s \xrightarrow{\mu} s', s \neq s' \iff s \xrightarrow{\mu_{\eta}} s', s \neq s' $$

$$ \eta \geq E(s) \iff s \xrightarrow{\mu_{\eta}} s, \text{ where } \mu' = \eta - (E(s) - R(s, s)). $$

Lemma 4. $\mathcal{M}_{\eta}$ is uniform, its size is linear in the size of $\mathcal{M}$ and $dR^{\text{opt}}_{\mathcal{M}_{\eta}, \beta} = dR^{\text{opt}}_{\mathcal{M}, \beta}$.

Proof. It is obvious that thus obtained MRA is uniform. Since the transformation adds at most as many transitions as there are Markovian states, the transformation is linear. We will show now that the discounted reward is preserved. Let $\mathcal{M}$ be a Markov reward automaton and $\mathcal{M}_{\eta}$ - the uniformised version of $\mathcal{M}$ with the uniformisation rate $\eta$.

We show now, that for every scheduler $\sigma$ on $\mathcal{M}$ there exists a scheduler $\sigma_{\eta}$ on $\mathcal{M}_{\eta}$ that achieves the same discounted reward value.

We denote with $[\pi]_{\eta} \subseteq \mathit{Paths}_{\mathcal{M}_{\eta}}$ the set of paths $\pi'$ of $\mathcal{M}_{\eta}$ that imitate $\pi$ fully for probabilistic transitions, and partially for Markovian transitions. Namely, for all transitions leading from a state $s \in MS$ to state $s' \in S$, such that $s' \neq s \pi'$ takes exactly the same transition, however it may additionally have several self-loop transitions on $s$ before moving to $s'$.

For each scheduler $\sigma \in GM$ on $\mathcal{M}$ we define a scheduler $\sigma_{\eta} \in GM$ on $\mathcal{M}_{\eta}$, such that $\sigma_{\eta}(\pi', \alpha) = \sigma(\pi, \alpha)$, where $\pi' \in [\pi]_{\eta}$.

Let $E[\text{rew}_{\mathcal{M}, \beta}|v, s' \neq v]$ denote the expected discounted reward, received by $\mathcal{M}$ conditional on starting in a Markovian state $v$ and until taking a transition to state $s' \neq v$.

As shown in [18] (Proposition 11.5.1), $E[\text{rew}_{\mathcal{M}, \beta}|v, s' \neq v] = E[\text{rew}_{\mathcal{M}_{\eta}, \beta}|v, s' \neq v]$. Therefore, the expected total discounted rewards during the residence in $v$ agree for the original and the uniformised processes.

Additionally coincide the distributions of sojourn times for Markovian states in $\mathcal{M}$ and $\mathcal{M}_{\eta}$, as well as the discrete distributions $\mathbb{P}_{\mathcal{M}}[v, s']$ and $\mathbb{P}_{\mathcal{M}_{\eta}}[v, s']$, for all $s' \neq v$.

Since the scheduler $\sigma_{\eta}$ mimics $\sigma$, the expected reward gathered from probabilistic states in $\mathcal{M}_{\eta}$, as well as the discrete probability distributions over successor states, induced by $\sigma_{\eta}$ coincide with those of $\sigma$ on $\mathcal{M}$.

Thus, $dR^{\text{opt}}_{\mathcal{M}_{\eta}, \beta} = dR^{\text{opt}}_{\mathcal{M}, \beta}$. □

Normalisation. We will now normalise $\mathcal{M}$. Informally, in order to achieve the desired effect we simply introduce probabilistic states of zero reward. There are surely many ways of normalising an MRA. Here we present one of the possibilities to do it.
The normalisation of $\mathcal{M}$ is an MRA $\overline{\mathcal{M}} = (S, \overline{s}_{\text{init}}, \overline{\text{Act}} \cup \{\alpha^*\}, \overline{\rightarrow}, \overline{\leftarrow}, \rho, \overline{\pi})$. We start with $\mathcal{M}$ and in the process of transformation will add new states and transitions to it, as well as modify existing ones:

1. Repeat until $\mathcal{M}$ satisfies normalisation properties \(2\) and \(3\)
   
   (a) If $s \in MS_\mathcal{M} \land \text{pred}(s) \cap MS_\mathcal{M} \neq \emptyset$, or $s \in PS_\mathcal{M} \land \text{pred}(s) \cap PS_\mathcal{M} \neq \emptyset \land \text{pred}(s) \cap MS_\mathcal{M} \neq \emptyset$, then create a new probabilistic state $q$ and place it in between $s$ and its Markovian predecessors. Or, formally:
   
   i. set the rewards of all newly added states to 0.
   
   ii. if for some $\pi \in \text{Paths}_\mathcal{M}$ of $\mathcal{M}$, then for each subpath $s \rightarrow s'$ of $\pi$, set $\lambda' := \sum_{s \rightarrow s'} \lambda$ and add transition $s \rightarrow q$.
   
   (b) If $s \in MS_\mathcal{M}$ and $\text{succ}(s) \cap MS_\mathcal{M} \neq \emptyset$, then then create a new probabilistic state $q$ and
   
   i. if $s_{\text{init}} \in PS_\mathcal{M}$ and $\text{pred}(s_{\text{init}}) \subset MS_\mathcal{M}$, then $s_{\text{init}} = s_{\text{init}}$. Otherwise create a new state $s_{\text{init}}'$, add it to $S$, and set $s_{\text{init}} = s_{\text{init}}'$.

2. set the rewards of all newly added states to 0.

Lemma 5. $\overline{\mathcal{M}}$ satisfies the normalisation properties, its size is linear in the size of $\mathcal{M}$ and $dR^\mathcal{M}_{\alpha, \beta} = dR^\overline{\mathcal{M}}_{\alpha, \beta}$.

Proof. Obviously thus constructed MRA is normalised. The amount of added states is not higher than the amount of Markovian transitions of the original MRA $+1$ and the transformation is therefore linear. We will show now that it also preserves the discounted reward value.

We first show that there exists a one-to-one correspondence between the paths of the original MRA $\mathcal{M}$ and its normalisation $\overline{\mathcal{M}}$. Let $\pi = s_0 \rightarrow s_1 \rightarrow \cdots \in \text{Paths}_\mathcal{M}$ be a path in $\mathcal{M}$. Then there exists a path in the normalised MRA $\pi' \in \text{Paths}_{\overline{\mathcal{M}}}$ such that

- let $s_0 \rightarrow s'$ be the very first transition from $s_0$ on $\pi$. If a new initial state $s_0'$ was added to $\mathcal{M}$ during the normalisation, then the corresponding subpath of $\pi'$ is $s_0' \rightarrow s_0 \rightarrow s'$, else $\pi'[0] = \pi[0]$;
- for each subpath $s \rightarrow s'$ of $\pi$, the corresponding subpath of $\pi'$ is $s \rightarrow q \rightarrow s'$;
- if for some $k$ : $\pi[k] = s' \in PS_\mathcal{M}$ and $s'$ has both Markovian and probabilistic predecessors, then for each subpath $s \rightarrow s'$ of $\pi$, the corresponding subpath of $\pi'$ is $s \rightarrow q \rightarrow s'$;
- $\pi'$ repeats $\pi$ in all other cases.
Analogously, for each path $\pi'$ of $\mathcal{M}'$ there exists a path $\pi$ in $\mathcal{M}$. Since all the introduced states are probabilistic states, that have only 1 action leading to only 1 successor, the probability measure of a set of paths $E$ in $\mathcal{M}$ equals the probability measure of the corresponding set of paths $E'$ in $\mathcal{M}'$. Since the introduced states have reward 0, then $\text{rew}(\pi) = \text{rew}(\pi')$. Thus the total expected reward value is preserved under normalisation.

\[ \square \]

Lemma 2. For any MRA $\mathcal{M}, \eta \geq \max_{s \in S} E(s)$ there exists a uniform normalised MRA $\overline{\mathcal{M}}$, s. t. $d^*_{\mathcal{M}} = d^*_{\mathcal{M}}$, and its size is linear in the size of $\mathcal{M}$.

Proof. The proof follows from Lemmas 5 and 4.

A.4 Bellman Equations.

Bellman Equation derived from $\mathcal{C}(\mathcal{M})$

Theorem 4 (Bellman equation via $\mathcal{C}(\mathcal{M})$). Let $\mathcal{M}$ be a normalized uniformized MRA with exit rate $\eta$, $\mathcal{C}(\mathcal{M}) = (S_{\text{mrk}}, s_{\text{init}}^\mathcal{C}, \text{Act}^\mathcal{C}, R^\mathcal{C})$ and $(\rho^\mathcal{C}, r^\mathcal{C})$ – the value preserving CTMDP and the corresponding reward structure. Then the vector $d^*_\mathcal{M} := (d^*_\mathcal{M}(s)), \forall s \in S_{\text{mrk}}$, is the unique solution to the Bellman equation:

\[ \forall s \in S_{\text{mrk}}: \quad v(s) = \text{opt}_{A \in \text{Act}(s)} \left\{ r^\mathcal{C}(s, A_s) + \frac{r^\mathcal{C}(s)}{\beta + \eta} + \frac{\eta}{\beta + \eta} \sum_{s' \in S_{\text{mrk}}} P^\mathcal{C}[s, A_s, s'] \cdot v(s') \right\}. \quad (2) \]

Proof. The proof follows directly from Theorem 1 and the corresponding proof for CTMDPs from [18].


Lemma 6. Let $\mathcal{D}$ and $r_\mathcal{D}$ be a MDP and a reward structure, such that the value $t^*_\mathcal{D}(s)$ exists. Let $N > 0$, then

\[ E_{s,\mathcal{D}} \left[ \sum_{i=0}^{N-1} r_\mathcal{D}(X_i^s, Y_i^s) \right] = \sum_{\pi \in \text{Paths}_\mathcal{D}} \Pr_\mathcal{D}[\pi] \cdot r_\mathcal{D}(\pi), \]

where $\Pr_\mathcal{D}[\pi] = \prod_{i=0}^{|\pi|-1} \Pr_{\mathcal{D}}[\pi[i], D(\pi[i]), \pi[i+1]]$ and $r_\mathcal{D}(\pi) = \sum_{i=0}^{|\pi|-1} r_\mathcal{D}(\pi[i], D(\pi[i]))$. 


Proof.

\[
\begin{aligned}
E_{s,D} \left[ \sum_{i=0}^{N-1} r_D(X^*_i, Y^*_i) \right] &= \sum_{i=0}^{N-1} E_{s,D} [r_D(X^*_i, Y^*_i)] \\
&= \sum_{i=0}^{N-1} \sum_{s' \in S_D} \Pr_D[X^*_i = s'|X^*_0 = s] \cdot r_D(X^*_i, Y^*_i) \\
&= \sum_{i=0}^{N-1} \sum_{s' \in S_D} \Pr_D[\pi] \cdot r_D(s', D(s')) \\
&= \sum_{i=0}^{N-1} \sum_{\pi \in \text{Paths}_D} \Pr_D[\pi] \cdot r_D(\pi[i], D(\pi[i])) \\
&= \sum_{\pi \in \text{Paths}_D} \Pr_D[\pi] \cdot \sum_{i=0}^{N-1} r_D(\pi[i], D(\pi[i])) \\
&= \sum_{\pi \in \text{Paths}_D} \Pr_D[\pi] \cdot D(\pi).
\end{aligned}
\]

\[\square\]

**Theorem 2.** Let \( \mathcal{M} \) be a uniform normalised MRA with exit rate \( \eta \) and \( D(\mathcal{M}) \) the corresponding terminal MDP. Then the vector \( d^{\text{opt}}_{\mathcal{M}, \beta} := (d^{\text{opt}}_{\mathcal{M}, \beta}(s))_{s \in S_{\text{mrk}}} \) is the unique solution to the Bellman equation:

\[
\forall s \in S_{\text{mrk}} : \quad v(s) = t^{\text{opt}}_{D(\mathcal{M}), \text{rew}(\mathcal{M}), v}(s).
\]

**Proof.** Consider the right-hand side of equation 2:

\[
\begin{aligned}
&\ r_C(s, A_s) + \frac{\rho_C(s)}{\beta} + \frac{\eta}{\beta + \eta} \sum_{s' \in S_{\text{mrk}}} \mathbb{P}_C[s, A_s, s'] \cdot v(s') \\
&= \sum_{s' \in S_{\text{mrk}}} \sum_{\pi \in \Pi \setminus S_{\text{mrk}}(s, s')} r(\pi) \cdot \mathbb{P}[\pi] \\
&\quad + \frac{1}{\beta + \eta} \sum_{s' \in S_{\text{mrk}}} \sum_{\pi \in \Pi \setminus S_{\text{mrk}}(s, A_s)} \rho(\pi) \cdot \mathbb{P}[\pi] \\
&\quad + \frac{\eta}{\beta + \eta} \sum_{s' \in S_{\text{mrk}}} \sum_{\pi \in \Pi \setminus S_{\text{mrk}}(s, A_s)} \mathbb{P}[\pi] \cdot v(s') \\
&= \sum_{s' \in S_{\text{mrk}}} \sum_{\pi \in \Pi \setminus S_{\text{mrk}}(s, s')} \mathbb{P}[\pi] \cdot \left[ r(\pi) + \frac{\rho(\pi)}{\beta} + \frac{\eta}{\beta + \eta} \cdot v(s') \right] \\
&\quad \text{Lemma 4} \quad t^{\text{opt}}_{D(\mathcal{M}), \text{rew}(\mathcal{M}), v}(s).
\end{aligned}
\]
A.5 Proof of Theorem 3

Theorem 3. Algorithms 1 and 2 are sound and complete.

Proof. The proof of Theorem 2 shows that the right-hand side of equation 2 equals the right-hand side of equation 1. Thus, a value- or modified policy-iteration algorithm for the Bellman equation from Theorem 4 in which the computation of the right-hand side of 2 is substituted with the right-hand side of 1 is as sound and complete as the original algorithm, given that the expected total reward is quantified precisely. The proof follows from results for respective algorithms for CTMDPs, see [18].