Abstraction-based Computation of Reward Measures for Markov Automata[⋆]

Bettina Braitling[1], Luis María Ferrer Fioriti[2], Hassan Hatefi[2], Ralf Wimmer[1], Bernd Becker[1], and Holger Hermanns[2]

[1] Albert-Ludwigs-Universität Freiburg, Germany
{braitlin | wimmer | becker}@informatik.uni-freiburg.de
[2] Saarland University, Saarbrücken, Germany
{ferrer | hhatefi | hermanns}@cs.uni-saarland.de

Abstract. Markov automata allow us to model a wide range of complex real-life systems by combining continuous stochastic timing with probabilistic transitions and nondeterministic choices. By adding a reward function it is possible to model costs like the energy consumption of a system as well.
However, models of real-life systems tend to be large, and the analysis methods for such powerful models like Markov (reward) automata do not scale well, which limits their applicability. To solve this problem we present an abstraction technique for Markov reward automata, based on stochastic games, together with automatic refinement methods for the computation of time-bounded accumulated reward properties. Experiments show a significant speed-up and reduction in system size compared to direct analysis methods.

1 Introduction

During the last few years Markov automata (MA) [1] have become a popular formalism for modelling stochastic systems. Markov automata are compositional, allowing us to model large systems component-wise and to obtain a model for the whole system by combining the models of the components according to fixed composition rules. Markov automata combine nondeterminism with probabilistic behaviour and continuous stochastic timing. Thus they are a generalisation of discrete-time Markov chains (DTMCs), Markov decision processes (MDPs), probabilistic automata (PA), continuous-time Markov chains (CTMCs), and interactive Markov chains (IMCs [2]). Markov automata form the semantic foundation of generalised stochastic Petri nets (GSPNs) [3] and stochastic activity networks (SANs) [4]. For modelling systems as MA, the Markov automata process

[⋆] This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center AVACS (SFB/TR 14), by the EU 7th Framework Programme under grant agreement no. 295261 (MEALS) and 318490 (SENSATION), and by the CAS/SAFEA International Partnership Program for Creative Research Teams.

algebra (MAPA) [5] has been devised. It is accompanied with tool support: SCOOP [5] transforms descriptions in MAPA into the underlying MA and is able to apply reduction techniques to reduce the MA's size.

Such a powerful modelling formalism is, however, only useful in practice if it is accompanied by efficient analysis algorithms. Model checking of MA against continuous stochastic logic (CSL) has been discussed in [6]. Algorithms for a wide range of properties for MA have been developed: long-run average, expected reachability time, and time-bounded reachability probabilities have been considered in [7, 8]. Markov reward automata (MRA), which is the extension of MA by rewards, have recently been studied in [9]. The analysis algorithms for MA and MRA are implemented in the tool IMCA[1].

Model checking algorithms for many kinds of properties like time-unbounded reachability, expected reachability costs, and long-run averages can be transferred to MA from simpler models like PA and are similarly efficient. In contrast, checking time-bounded properties is much more expensive. The reason is that the methods based on uniformisation, which make checking time-bounded properties on CTMCs efficient, cannot be applied to MA due to the nondeterminism present there. Instead (like for IMCs [10]) one has to resort to discretisation [7–9]: the time until the time bound is split into small intervals such that one can assume that with high probability either none or exactly one step occurs within one interval. For each of these intervals, an unbounded reachability analysis for PA has to be performed. Therefore the analysis of such properties scales badly to large state spaces and is limited to a few thousand states, depending on the structure of the state space and the time bound.

Contributions. To tackle this problem for MRA we present an *abstraction and refinement framework*, the first of its kind. We target *time-bounded accumulated rewards* like "What is the maximal expected cost the system causes within 10 hours of operation?" The MRA at hand is abstracted into a two-player stochastic reward game, which keeps the nondeterminism present in the concrete system separate from the nondeterminism introduced by abstraction. This allows us to compute safe upper and lower bounds on the minimal and maximal reward value of the original system. These bounds are an in-built quality measure for the abstraction, allowing us to refine it. To compute the bounds, we give a fixed point characterisation of time-bounded accumulated rewards on stochastic games, show how to discretise it, and give an estimation of the error caused by the discretisation. Experimental results confirm that our abstraction method yields substantial reductions in system size and reduces the computation times compared to competing tools which work on the concrete state space.

Related Work. This paper continues a series of successful works on abstraction frameworks for simpler probabilistic models, foremost rooted in game-based abstraction for PA [11]. In a preliminary paper [12] we have presented an abstraction framework for time-bounded reachability probabilities for MA, which

[1] The official homepage available at `http://www-i2.informatik.rwth-aachen.de/imca`.

was the first attempt to apply abstraction to MA. Other abstraction methods are restricted to unbounded and step-bounded reachability probabilities in PA: PA-based abstraction [13] abstracts a PA again into a PA whose behaviour is an over-approximation of the behaviour of the original model. It therefore only allows to compute lower bounds on minimal probabilities and upper bounds on maximal probabilities. This is improved by game-based abstraction [11], which yields a probabilistic game. Its advantage is that it yields both upper and lower bounds for minimal and maximal reachability probabilities. Wachter and Zhang [14, 15] have proposed menu-based abstraction for PA, which has the same advantage as game-based abstraction, but yields in many practical cases significantly smaller abstractions.

Structure of the paper. In the next section we briefly review the foundations of MA and stochastic games. In Sec. 3 we present our abstraction and refinement method for MA and show how to compute reward measures for stochastic games. The experimental evaluation follows in Sec. 4. We finally summarise the paper with an outlook to future work in Sec. 5. An extended version of this paper with proofs of the main propositions is available at [16].

2 Preliminaries

We first introduce the necessary foundations on stochastic games (SGs), extended by reward functions, which form the basic formalism used in our abstraction framework. We define the properties we consider and give a fixed point characterisation of them for SGs. Finally we define MA as a special case of SGs with a single player.

We denote the set of real numbers by $\mathbb{R}$, the non-negative real numbers by $\mathbb{R}^{\geq 0}$, and by $\mathbb{R}^{\geq 0}_{\infty}$ the set $\mathbb{R}^{\geq 0} \,\dot\cup\, \{\infty\}$. For a finite or countable set S let $\mathrm{Distr}(S)$ denote the set of probability distributions on S, i. e. of all functions $\mu : S \to [0, 1]$ with $\sum_{s \in S} \mu(s) = 1$. The support of a distribution μ is given by $\mathrm{Supp}(\mu) = \{s \in S \mid \mu(s) > 0\}$; μ is called Dirac if there is $s \in S$ with $\mu(s) = 1$. It is denoted by ξ_s. Given a set $S' \subseteq S$ we write $\mu(S')$ for $\sum_{s \in S'} \mu(s)$.

2.1 Stochastic reward games

Stochastic games are a behavioural model that combines stochastic timing, nondeterminism and probabilistic choices. An SG consists of one or more players who can choose between one or more transitions to change the current state. Each choice may influence the behaviour of the other players. A transition consists of a real-valued or infinite rate $\lambda \in \mathbb{R}^{\geq 0}_{\infty}$ and a probability distribution over the successor states. For our work we need the definition of stochastic two-player games:

Definition 1 (Stochastic game). *A* stochastic (continuous-time two-player) game *(SG) is a tuple* $\mathcal{G} = \big(V, (V_1, V_2), v_{\text{init}}, \mathbf{T}\big)$ *such that* $V = V_1 \,\dot\cup\, V_2$ *is a set of states,* $v_{\text{init}} \in V$ *is the initial state, and* $\mathbf{T} \subseteq V \times \mathbb{R}^{\geq 0}_{\infty} \times \mathrm{Distr}(V)$ *is a transition relation.*

V_1 and V_2 are the states of player 1 and player 2, respectively; we also denote them as V_1- and V_2-states. Transitions $(v, \lambda, \mu) \in \mathbf{T}$ with rate $\lambda < \infty$ are called *Markovian*, transitions with infinite rate *probabilistic*. We denote the set of Markovian and probabilistic transitions by $\mathbf{T}_M$ and $\mathbf{T}_P$, respectively; it holds that $\mathbf{T} = \mathbf{T}_M \,\dot{\cup}\, \mathbf{T}_P$. $\mathbf{T}_M(v)$ and $\mathbf{T}_P(v)$ denote the set of Markovian and probabilistic transitions available at state v, respectively. We use $\mathbf{T}(v) = \mathbf{T}_M(v) \,\dot{\cup}\, \mathbf{T}_P(v)$ as the set of all transitions available at state v.

The game starts in state v_{init}. If the current state is $v \in V_1$, then it is player 1's turn, otherwise player 2's. The current player chooses a transition $(v, \lambda, \mu) \in \mathbf{T}(v)$ for leaving state v. The rate $\theta_r\left((v, \lambda, \mu)\right) = \lambda \in \mathbb{R}^{\geq 0}_{\infty}$ determines how long we stay at v, whereas $\theta_p\left((v, \lambda, \mu)\right) = \mu \in \text{Distr}(V)$ gives us the distribution which leads to the successor states. If $\lambda = \infty$, the transition is taken instantaneously. Otherwise, λ is taken as the parameter of an exponential distribution. In this case, the probability that a transition to state $v' \in V$ happens within $t \geq 0$ time units, is given by $\mu(v') \cdot (1 - \mathrm{e}^{-\lambda \cdot t})$. For conciseness, we write λ_{tr} instead of $\theta_r(tr)$ and μ_{tr} instead of $\theta_p(tr)$ for $tr \in \mathbf{T}$.

Paths. The dynamics of SGs is specified by paths. An infinite path $\pi \in (V \times \mathbb{R}^{\geq 0} \times \mathbf{T})^{\omega}$ is an infinite sequence of states, sojourn times, and transitions. A finite path is such a sequence which is finite and ending in a state, i. e. $\pi \in (V \times \mathbb{R}^{\geq 0} \times \mathbf{T})^{\star} \times V$. We usually write $v \xrightarrow{t, tr}$ instead of $(v, t, tr) \in (V \times \mathbb{R}^{\geq 0} \times \mathbf{T})$. We use $\mathit{Paths}^{\star}$ and Paths^{ω} to denote the set of finite and infinite paths, respectively. Given a finite path $\pi = v_0 \xrightarrow{t_0, tr_0} v_1 \xrightarrow{t_1, tr_1} \cdots v_n$, v_i is the $(i+1)$-th state of π, denoted by $\text{St}(\pi, i)$, t_i is the time of staying at v_i, denoted by $\text{Ti}(\pi, i)$, and $\text{Tr}(\pi, i) = tr_i$ is the executed transition for $i \in \{0, \dots, n-1\}$. Note that v_i is left instantaneously, i. e. $\text{Ti}(\pi, i) = 0$, if $\text{Tr}(\pi, i)$ has an infinite rate. Moreover, $|\pi|$ refers to n, the length of π, and $\text{last}(\pi)$ to v_n, its last state.

Strategies. The nondeterminism which may occur at a state is resolved by a function, which is called a *strategy* (or policy or scheduler). Each player follows its own strategy in order to accomplish its goal. A strategy of player i $(i = 1, 2)$ is a partial function $\sigma_i : \mathit{Paths}^{\star} \nrightarrow \text{Distr}(\mathbf{T})$ such that $\sigma_i(\pi) = \eta$ only if $\text{last}(\pi) \in V_i$ and $\text{Supp}(\eta) \subseteq \mathbf{T}\big(\text{last}(\pi)\big)$. This strategy class is called *generic*, since it uses the complete path history to resolve the nondeterminism.[2] We denote the set of all strategies for player i by Strat_i.

Probability measure. Given strategies σ_1, σ_2 for both players and a state $v \in V$, a probability space on the set of infinite paths starting in v can be constructed. The set of measurable events is thereby a σ-algebra that is induced by a standard cylinder set construction [18] together with a unique probability measure $\text{Pr}_{v, \sigma_1, \sigma_2}$

[2] This class is also known as the class of *early schedulers* [17] because the scheduler makes its choice when entering a state and—in contrast to a late scheduler—may not change its choice while residing in a state. This is the most general scheduler class for MA, since they do not exhibit nondeterminism between Markovian transitions (see Sec. 2.3).

on the events. $\Pr_{v,\sigma_1,\sigma_2}(\Pi)$ is the probability of the set of paths Π, starting from state v, given that player 1 and player 2 play with strategies σ_1 and σ_2, respectively. Both the σ-algebra and the probability measure are constructed by extending the existing techniques used for MA and IMCs. We omit the details here; for more information see, e. g. [6, 19, 20].

Zenoness. It may happen that an SG contains an end component consisting of probabilistic transitions only. Such an end component leads to the existence of infinite paths π with finite sojourn times, i. e. $\lim_{n\to\infty} \sum_{i=0}^{n} \mathrm{Ti}(\pi, i) < \infty$. This phenomenon is called *Zenoness.* Since such behaviour has to be considered unrealistic, we assume that the SGs under consideration are non-Zeno, i. e. that they do not contain such end components. Formally, an SG is non-Zeno iff $\forall v \in V : \forall\sigma_1 \in \mathrm{Strat}_1 \wedge \forall\sigma_2 \in \mathrm{Strat}_2 : \Pr_{v,\sigma_1,\sigma_2}\big(\{\pi \mid \lim_{n\to\infty} \sum_{i=0}^{n} \mathrm{Ti}(\pi, i) < \infty\}\big) = 0$.

For more on strategies and on SGs in general we refer to [21, 22].

Now we extend SGs by rewards (or costs). We consider two kinds of rewards: *transient rewards* for staying in a certain state and *instantaneous rewards* for taking a transition.

Definition 2 (Stochastic reward game). *A* stochastic reward game *(SRG) is a tuple $\mathcal{G}_{rew} = (\mathcal{G}, \rho_t, \rho_i)$ such that $\mathcal{G}$ is an SG, $\rho_t : \mathbf{T} \to \mathbb{R}^{\geq 0}$ the transient reward function, and $\rho_i : \mathbf{T} \to \mathbb{R}^{\geq 0}$ the instantaneous reward function.*

The transient reward $\rho_t(tr)$ of a transition $tr = (v, \lambda, \mu)$ is the cost of staying in v for one time unit before taking transition tr, i. e. residing in state v for Δ time units yields a transient reward of $\Delta \cdot \rho_t(tr)$. Since a state is immediately left if a transition with infinite rate is chosen, we can assume that the transient reward of such transitions is zero.

The instantaneous reward $\rho_i(tr)$ is the cost of the state change using transition $tr \in \mathbf{T}$. The accumulated reward along a path π is the sum of the costs for the transitions and the costs for staying in the states of the path. We are interested in time-bounded rewards, i. e. the costs accumulated until a time bound T is reached. It is denoted by Rew_T.

$$\begin{aligned} Rew_T(\pi) = \sum_{i=0}^{n_T-1} \Big(\rho_t\big(\mathrm{Tr}(\pi, i)\big) \cdot \mathrm{Ti}(\pi, i) + \rho_i\big(\mathrm{Tr}(\pi, i)\big)\Big) \\ + \rho_t\big(\mathrm{Tr}(\pi, n_T)\big) \cdot \big(T - \sum_{i=0}^{n_T-1} \mathrm{Ti}(\pi, i)\big), \end{aligned} \tag{1}$$

where n_T is the largest number such that $\sum_{i=0}^{n_T-1} \mathrm{Ti}(\pi, i) \leq T$. Each player can independently of the other try to maximise or minimise the expectation of this reward by choosing an appropriate scheduler.

$$\mathcal{R}^{\mathrm{opt}_1}_{\mathrm{opt}_2}(v, T) = \underset{\sigma_1 \in \mathrm{Strat}_1}{\mathrm{opt}_1} \; \underset{\sigma_2 \in \mathrm{Strat}_2}{\mathrm{opt}_2} \int_{\pi \in Paths^{\omega}} Rew_T(\pi) \, \mathrm{d}\Pr_{v,\sigma_1,\sigma_2}(\pi) \tag{2}$$

is the opt_1-opt_2 expected time-bounded reward (ETR) when player i tries to optimise according to $\mathrm{opt}_i \in \{\inf, \sup\}$ for $i = 1, 2$ and the game starts in state $v \in V$.

2.2 Time-bounded reward as a fixed point

In this section we provide a fixed point characterisation of the ETR for an SG. In the following we restrict the presentation to the case where player 1 maximises the expected reward, i. e. setting opt_1 to sup in (2). We denote the case by $\mathcal{R}^{\sup}_{\mathrm{opt}}$, where player 2 still has the choice to either minimise or maximise the ETR.

Lemma 1 (Fixed point characterisation). *Given an SRG $\mathcal{G}_{rew} = (\mathcal{G}, \rho_t, \rho_i)$, a time bound $T \geq 0$, $\mathrm{opt} \in \{\inf, \sup\}$, $\mathrm{opt}_v = \sup$ if $v \in V_1$ and $\mathrm{opt}_v = \mathrm{opt}$ otherwise. $\mathcal{R}^{\sup}_{\mathrm{opt}}(\cdot, T)$ is the least fixed point of the higher order operator $\Omega_{\mathrm{opt}} : (V \times \mathbb{R}^{\geq 0} \mapsto \mathbb{R}^{\geq 0}) \mapsto (V \times \mathbb{R}^{\geq 0} \mapsto \mathbb{R}^{\geq 0})$, $\mathrm{opt} \in \{\inf, \sup\}$, such that*

$$\Omega_{\mathrm{opt}}(F)(v, T) = \underset{tr \in \mathbf{T}(v)}{\mathrm{opt}_v} \begin{cases} \left(\rho_i(tr) + \frac{\rho_t(tr)}{\lambda_{tr}}\right)\left(1 - \mathrm{e}^{-\lambda_{tr} T}\right) & \\ + \int\limits_0^T \lambda_{tr} \mathrm{e}^{-\lambda_{tr} t} \sum\limits_{v' \in V} \mu_{tr}(v') F(v', T - t)\, \mathrm{d}t, & \text{if } tr \in \mathbf{T}_M(v), \\ \rho_i(tr) + \sum\limits_{v' \in V} \mu_{tr}(v') F(v', T), & \text{if } tr \in \mathbf{T}_P(v). \end{cases} \tag{3}$$

A similar fixed point characterisation can be attained for the case that player 1 minimises the ETR. Both characterisations, however, yield Volterra integral equation systems which are not directly tractable [23]. We demonstrate in Sec. 3.2 how to approximate $\mathcal{R}^{\sup}_{\mathrm{opt}}$ by applying a discretisation technique. Moreover, the characterisation provides a sound theory for the abstraction of MA subject to time-bounded reward analysis (see Sec. 3).

2.3 Markov automata

For consistency reasons we present MA [1, 24, 25] as a special case of SGs. This is possible under two conditions: First, we consider only *closed* MA, i. e. we assume that the model to be analysed is not subject to further composition operations. Then the actions with which transitions are labelled do not carry any information and can be omitted. Second, we make the *maximal progress assumption* [25], which is typically made for closed MA before analysis. It says that probabilistic transitions (which are carried out immediately without progress of time) have precedence over Markovian transitions, which are delayed by an exponentially distributed amount of time. Together with the restriction that there is no nondeterminism between Markovian transitions in MA, we obtain the following definition:

Definition 3 (Markov (reward) automaton). *An SG* $\mathcal{M} = \big(V, (V_1, V_2), v_{\text{init}}, \mathbf{T}\big)$ *is a* Markov automaton *(MA) if* $V_2 = \emptyset$ *and each state* $v \in V$ *contains either only probabilistic transitions* $(v, \infty, \mu) \in \mathbf{T}(v)$ *or a single Markovian transition* $(v, \lambda, \mu) \in \mathbf{T}(v)$ *with* $\lambda < \infty$*.*

A Markov reward automaton *(MRA) is an SRG* $\mathcal{M}_{rew} = (\mathcal{M}, \rho_t, \rho_i)$ *where* $\mathcal{M}$ *is an MA.*

The fixed point characterisation of the ETR for SRGs is valid for MRA as well. Since the set V_2 is empty for MRA, we can omit the opt subscript from Ω_{opt} as defined in Eq. (3) and simply write it as Ω in this case.

Like SGs can be seen as a generalisation of MA, MA can be seen as the generalisation of some other common models: A (closed) probabilistic automaton (PA) is an MA where all transitions are probabilistic. A PA with $\big|\{\mu \mid (v, \infty, \mu) \in \mathbf{T}(v)\}\big| = 1$ for all $v \in V$ is a discrete-time Markov chain (DTMC). An interactive Markov chain (IMC) is an MA where all distributions occurring in probabilistic transitions are Dirac. A continuous-time Markov chain (CTMC) is an MA with only Markovian transitions.

We partition the state space V of an MA based on the rate of outgoing transitions of the states. A state is called *probabilistic* (*Markovian*) if its outgoing transitions are all probabilistic (Markovian). We assume that there are no *deadlock* states, which have no outgoing transitions. They can be turned into Markovian states by adding a Markovian transition (v, λ, ξ_v) with arbitrary rate $\lambda < \infty$. Then we have $V = \mathit{MS} \mathbin{\dot\cup} \mathit{PS}$ with MS being the Markovian states and PS the probabilistic states.

3 Abstraction and refinement of MRA

In this section we first describe our abstraction of an MA, then how safe bounds on the maximal (minimal) ETR can be computed on SGs (and therefore also on our abstraction). Finally we will show how an abstraction can be refined in case that it is too coarse, i. e. if it yields bounds that are too far apart.

3.1 Abstraction

The abstraction of an MA $\mathcal{M} = (V, (V, \emptyset), v_{\text{init}}, \mathbf{T})$ is based on a partition of V, which is a set $\mathcal{P} \subseteq 2^V \setminus \{\emptyset\}$ with $\bigcup_{B \in \mathcal{P}} B = V$ and $B \cap B' = \emptyset$ for all $B, B' \in \mathcal{P}$ with $B \neq B'$. For $v \in V$ we denote the unique block B of $\mathcal{P}$ with $v \in B$ by $[v]_{\mathcal{P}}$.

Definition 4 (Lifted distribution). *Let* $\mu \in \text{Distr}(V)$ *be a probability distribution over* V *and* $\mathcal{P}$ *a partition of* V*. The* lifted distribution $\overline{\mu} \in \text{Distr}(\mathcal{P})$ *is given by* $\overline{\mu}(B) = \sum_{v \in B} \mu(v)$ *for* $B \in \mathcal{P}$*.*

Definition 5 (Labelling function). *Let Lab be a finite set of labels and* $\mathcal{M}$ *an MA. A* labelling function *is a function* $lab : \mathbf{T} \to Lab$ *which is injective at each state, i. e. for all* $v \in V$ *and all* $\mathrm{tr}, \mathrm{tr}' \in \mathbf{T}(v)$ *we have either* $\mathrm{tr} = \mathrm{tr}'$ *or* $lab(\mathrm{tr}) \neq lab(\mathrm{tr}')$*. Additionally we require* $lab(\mathrm{tr}) = \perp$ *iff* tr *is Markovian.*

For a set $B \subseteq V$ of states, we define $Lab(B) = \{lab(tr) \mid \exists v \in B : tr \in \mathbf{T}(v)\}$ as the set of actions which are enabled in B. We also write $Lab(v)$ instead of $Lab\big(\{v\}\big)$.

A simple, basic way to abstract an MA $\mathcal{M}$ is to use the blocks of a partition $\mathcal{P}$ as abstract states and create the abstract transitions by lifting the distributions as in Def. 4. If two transitions coincide after lifting, they are combined into one abstract transition.

Such an abstraction introduces additional nondeterminism into the system: There is now a nondeterministic choice between transitions which belong to different concrete states, but to the same state in the abstraction. In a basic abstraction, this new, abstract nondeterminism cannot be distinguished from the original, concrete nondeterminism. This makes it impossible to obtain both lower and upper bounds on the actual reward value.

To avoid this effect, we introduced a game abstraction for MA in [12], which we now extend to MRA.

Definition 6 (Game abstraction of MA). *Given an MA $\mathcal{M} = (V, (V, \emptyset), v_{\text{init}}, \mathbf{T})$, a labelling function $lab : \mathbf{T} \to Lab$, and a partition $\mathcal{P} = \{B_1, \ldots B_n\}$ of V such that for all $B \in \mathcal{P}$ either $B \subseteq MS$ or $B \subseteq PS$. We construct the* game abstraction *$\overline{\mathcal{M}}^{\mathcal{P},lab} = (\overline{V}, (\overline{V_1}, \overline{V_2}), \overline{v_{\text{init}}}, \overline{\mathbf{T}})$ with:*

- *$\overline{V} = \overline{V_1} \,\dot\cup\, \overline{V_2}$,*
- *$\overline{V_1} = \mathcal{P}$,*
- *$\overline{V_2} = \big\{(B, \alpha) \in \mathcal{P} \times Lab \,\big|\, \alpha \in Lab(B)\big\} \,\dot\cup\, \{*\}$,*
- *$\overline{v_{\text{init}}} = [v_{\text{init}}]_{\mathcal{P}}$, and*
- *$\overline{\mathbf{T}} = \overline{\mathbf{T}_P} \,\dot\cup\, \overline{\mathbf{T}_M}$ with*

$$\begin{aligned}
\overline{\mathbf{T}_P} = &\big\{\big([v]_{\mathcal{P}}, \infty, \xi_{([v]_{\mathcal{P}}, \alpha)}\big) \,\big|\, v \in V \wedge \alpha \in Lab([v]_{\mathcal{P}})\big\} \\
&\dot\cup \big\{\big(([v]_{\mathcal{P}}, \alpha), \infty, \overline{\mu}\big) \,\big|\, v \in PS \wedge \alpha \in Lab(v) \wedge (v, \infty, \mu) \in \mathbf{T}_P\big\} \\
&\dot\cup \big\{\big(([v]_{\mathcal{P}}, \alpha), \infty, \xi_*\big) \,\big|\, v \in PS \wedge \alpha \in Lab([v]_{\mathcal{P}}) \setminus Lab(v)\big\}, \\
\overline{\mathbf{T}_M} = &\big\{\big(([v]_{\mathcal{P}}, \bot), \lambda, \overline{\mu}\big) \,\big|\, v \in MS \wedge (v, \lambda, \mu) \in \mathbf{T}_M\big\} \\
&\dot\cup \big\{(*, 1, \xi_*)\big\}\,.
\end{aligned}$$

We call $\overline{V_2}$-states of the form $(B, \bot)$ with $B \subseteq MS$ *Markovian* and $\overline{V_2}$-states of the form (B, α) with $\alpha \neq \bot$ and $B \subseteq PS$ *probabilistic*. The $\overline{V_1}$- and $\overline{V_2}$-states strictly alternate.

Player 1 resolves the nondeterminism already present in the concrete MA when it selects at state B a label α present in one of the concrete states of B. Player 2 resolves the nondeterminism introduced by the abstraction by selecting at abstract state (B, α) a concrete state $v \in B$ and firing the lifted transition of state v that has the label α. In case there is no transition with label α in state v, the abstraction goes to a special state $*$ that represents the worst outcome for the property under consideration. This is similar to the menu-based abstraction in [14, 15].

In order to abstract an MRA $\mathcal{M}_{rew} = (\mathcal{M}, \rho_t, \rho_i)$ we have to add *abstract reward functions* to the abstraction. For this we need an additional function

$\nu : \overline{\mathbf{T}} \to 2^{\mathbf{T}}$, which maps the abstract transitions back to the corresponding concrete transitions in $\mathcal{M}_{rew}$. For a transition $\overline{tr} = (\overline{v}, \lambda, \overline{\mu}) \in \overline{\mathbf{T}}$ we get:

$$\nu(\overline{tr}) = \begin{cases} \{(v', \infty, \mu) \in \mathbf{T} \mid v' \in B\}, & \text{if } \overline{v} = (B, \alpha) \wedge \alpha \neq \bot \wedge B \subseteq PS, \\ \{(v', \lambda, \mu) \in \mathbf{T} \mid v' \in B\}, & \text{if } \overline{v} = (B, \bot) \wedge B \subseteq MS, \\ \emptyset, & \text{otherwise.} \end{cases}$$

The choice of the reward function depends on whether we want to compute a lower or an upper bound on the (minimal or maximal) ETR. In case of a lower bound, player 2 chooses the smallest possible reward among all transitions which were mapped onto the same abstract transition. The case of an upper bound is analogous.

We give the definition of the reward structures for the case that player 1 maximises the ETR. If player 1 minimises the ETR, the only change is that $\rho_t\big((*, 1, \xi_*)\big)$ is set to ∞ instead of 0.

Definition 7 (opt-Abstraction-induced SRG). *Given an MRA $\mathcal{M}_{rew} = (\mathcal{M}, \rho_t, \rho_i)$, a partition $\mathcal{P}$ of the state space, a labelling function lab for the transitions, and* opt $\in \{\inf, \sup\}$*. Then the* opt-Abstraction-induced SRG *(or for short* opt-*AISRG) with respect to $\mathcal{P}$ and lab is a tuple* opt-$\overline{\mathcal{M}}_{rew}^{\mathcal{P},lab} = (\overline{\mathcal{M}}^{\mathcal{P},lab}, \overline{\rho_t}^{\text{opt}}, \overline{\rho_i}^{\text{opt}})$*, where $\overline{\mathcal{M}}^{\mathcal{P},lab}$ is the game abstraction of $\mathcal{M}$ obtained from Def. 6 and $\overline{\rho_t}^{\text{opt}}$ and $\overline{\rho_i}^{\text{opt}}$ are* abstract transient and instantaneous reward functions *defined as:*

$$\overline{\rho_t}^{\text{opt}}(\overline{tr}) = \begin{cases} \underset{tr \in \nu(\overline{tr})}{\text{opt}} \ \rho_t(tr), & \textit{if } \nu(\overline{tr}) \neq \emptyset, \\ 0, & \textit{otherwise.} \end{cases}$$

and

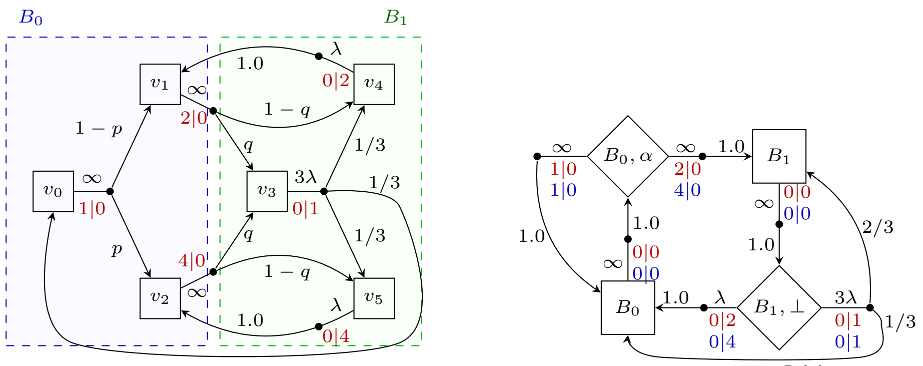
$$\overline{\rho_i}^{\text{opt}}(\overline{tr}) = \begin{cases} \underset{tr \in \nu(\overline{tr})}{\text{opt}} \ \rho_i(tr), & \textit{if } \nu(\overline{tr}) \neq \emptyset, \\ 0, & \textit{otherwise.} \end{cases}$$

respectively, where $\overline{tr} \in \overline{\mathbf{T}}$ is an abstract transition.

We illustrate the abstraction of an MRA in Example 1.

Example 1. Figure 1(a) shows an MRA $\mathcal{M}_{rew}$ together with a partition $\mathcal{P} = \{B_0, B_1\}$. We assume that all probabilistic transitions are labelled with $lab(tr) = \alpha$ and all Markovian transitions with $lab(tr) = \bot$. It holds $PS = \{v_0, v_1, v_2\} = B_0$ and $MS = \{v_3, v_4, v_5\} = B_1$. For each transition tr the rewards are given in the form "$(\rho_i(tr) | \rho_t(tr))$" next to the transition in red colour.

Figure 1(b) shows the resulting game abstraction. The blocks B_0 and B_1 have become $\overline{V_1}$-states, whereas all other states are $\overline{V_2}$-states. We show both reward structures: the values in red colour next to each abstract transition are the minimal rewards in the form "$(\overline{\rho_i}^{\inf}(tr) | \overline{\rho_t}^{\inf}(tr))$"; the blue figures below are the maximal rewards, shown as "$(\overline{\rho_i}^{\sup}(tr) | \overline{\rho_t}^{\sup}(tr))$".



(a) Original MRA $\mathcal{M}_{rew}$ (b) Abstraction $\overline{\mathcal{M}}^{\mathcal{P},lab}_{rew}$

Fig. 1. An example for the game abstraction of an MRA $\mathcal{M}_{rew}$.

The soundness of our framework follows from the fact that the least fixed points of the abstract semantics are over- and underapproaximations of the least fixed point in the concrete semantics.

Theorem 1 (Soundness). *Let* $\mathcal{M}_{rew} = (\mathcal{M}, \rho_t, \rho_i)$ *be an MRA, its high-order operator for the maximal ETR* Ω*, and* opt-$\overline{\mathcal{M}}^{\mathcal{P},lab}_{rew} = (\overline{\mathcal{M}}^{\mathcal{P},lab}, \overline{\rho_t}^{\text{opt}}, \overline{\rho_i}^{\text{opt}})$, $\text{opt} \in \{\inf, \sup\}$*, be a game abstraction with rewards, its high-order operators* $\Omega_{\inf}$*,* $\Omega_{\sup}$*. Then:*

$$\text{lfp}(\Omega_{\inf})([v]_{\mathcal{P}}, T) \leq \text{lfp}(\Omega)(v, T) \leq \text{lfp}(\Omega_{\sup})([v]_{\mathcal{P}}, T)$$

for all $v \in V$*, and* $T \in \mathbb{R}^{\geq 0}$*.*

3.2 Reward computation

In this section we describe how to compute optimal ETRs for the general class of SRGs. For this purpose a discretisation technique is employed, which is then applied to the fixed point characterisation given in Lemma 1. The technique yields a discretised fixed point characterisation accompanied by a stable numerical algorithm with strict error bound for computing the optimal ETRs in SRGs.

Discretisation. As stated before, it is not generally feasible to directly solve the fixed point characterisation in Lemma 1 due to the complex integrals occurring in Eq. (3). Instead the SRG subject to analysis needs to be *discretised*. For this, the interval $[0, T]$ is first split into k time steps of size $\delta = \frac{T}{k}$. The discretisation then simplifies the computation of $\mathcal{R}^{\sup}_{\text{opt}}$ by assuming that with high probability at most one Markovian transition fires within each time step. Finally, we provide lower and upper bounds for the error created by the discretisation.

We aim to simplify the reward computation proposed by Eq. (3). For that, we first express $\mathcal{R}^{\sup}_{\text{opt}}(v, T)$ in terms of its behaviour in the first discretisation step $[0, \delta)$ with $\text{opt} \in \{\inf, \sup\}$. As time passes only if a Markovian transition is taken, we assume w. l. o. g. that v only contains Markovian transitions, i. e. $\emptyset \neq \mathbf{T}(v) \subseteq \mathbf{T}_M$. To see why the assumption does not restrict the generality of the discretisation, note that the simplification only applies to the part of Eq. (3) that contains the integral equation, so the case corresponding to the probabilistic transitions remains untouched. We partition the paths from v that take transition $tr \in \mathbf{T}_M(v)$ into the set $\Pi_{v,tr}^{\delta,0}$ of paths that make no Markovian jump in $[0,\delta)$ and the set $\Pi_{v,tr}^{\delta,>0}$ of paths that do at least one jump in that interval. We therefore write $\mathcal{R}_{\text{opt}}^{\text{sup}}(v,T)$ as the sum of

1. The optimal expected reward attained in $[0,\delta)$ by paths from $\Pi_{v,tr}^{\delta,>0}$
2. The optimal expected reward attained in $[\delta,T]$ by paths from $\Pi_{v,tr}^{\delta,>0}$
3. The optimal expected reward attained in $[0,\delta)$ by paths from $\Pi_{v,tr}^{\delta,0}$
4. The optimal expected reward attained in $[\delta,T]$ by paths from $\Pi_{v,tr}^{\delta,0}$

It is not hard to express the last item in terms of $\mathcal{R}_{\text{opt}}^{\text{sup}}(v,T-\delta)$. We further combine the first three items, denoted by $Acc_{tr}(v,T)$, and finally have:

$$\mathcal{R}_{\text{opt}}^{\text{sup}}(v,T) = \underset{tr \in \mathbf{T}(v)}{\text{opt}_v} \big(Acc_{tr}(v,T) + \mathrm{e}^{-\lambda_{tr}\cdot\delta}\mathcal{R}_{\text{opt}}^{\text{sup}}(v,T-\delta)\big) \tag{4}$$

We can show (see [16]) that $Acc_{tr}(v,T)$ is obtained by:

$$\begin{aligned} Acc_{tr}(v,T) = &\big(\rho_i(tr) + \tfrac{\rho_t(tr)}{\lambda_{tr}}\big)\big(1 - \mathrm{e}^{-\lambda_{tr}\cdot\delta}\big) \\ &+ \int_0^\delta \lambda_{tr}\mathrm{e}^{-\lambda_{tr}\cdot t} \sum_{v' \in V} \mu_{tr}(v')\mathcal{R}_{\text{opt}}^{\text{sup}}(v',T-t)\,\mathrm{d}t \end{aligned} \tag{5}$$

As for the fixed point characterisation in Lemma 1, the exact computation of $Acc_{tr}(v,T)$, opt $\in \{\inf,\sup\}$ is in general intractable. However, if the discretisation constant δ is very small, then, with high probability, at most one Markovian jump happens in each discretisation step. Hence the reward gained by paths carrying multiple Markovian jumps within at least one such interval is negligible and can be omitted from the computation. In other words, the reward gained after the first Markovian jump in each discretisation constant is ignored by this approximation. It naturally induces some error and thereby approximates $Acc_{tr}(v,T)$, denoted by $\widetilde{Acc}_{\delta,tr}(v,k)$ and $\mathcal{R}_{\text{opt}}^{\text{sup}}(v,T)$, denoted by $\tilde{\mathcal{R}}_{\delta,\text{opt}}^{\text{sup}}(v,k)$. As a result we have:

$$\tilde{\mathcal{R}}_{\delta,\text{opt}}^{\text{sup}}(v,k) = \underset{tr \in \mathbf{T}(v)}{\text{opt}_v} \big(\widetilde{Acc}_{\delta,tr}(v,k) + \mathrm{e}^{-\lambda_{tr}\cdot\delta}\tilde{\mathcal{R}}_{\delta,\text{opt}}^{\text{sup}}(v,k-1)\big) \tag{6}$$

$\widetilde{Acc}_{\delta,tr}$ and $\tilde{\mathcal{R}}_{\delta,\text{opt}}^{\text{sup}}$ both count the number of discretisation steps instead of real time. This makes their computation tractable.

$$\begin{aligned}\widetilde{Acc}_{\delta,\mathrm{opt}}(v,k) &= \left(\rho_i(tr) + \tfrac{\rho_t(tr)}{\lambda_{tr}}\right)\left(1 - \mathrm{e}^{-\lambda_{tr}\cdot\delta}\right)\\ &\quad + \int_0^\delta \lambda_{tr}\mathrm{e}^{-\lambda_{tr}\cdot t} \sum_{v'\in V} \mu_{tr}(v')\tilde{\mathcal{R}}^{\sup}_{\mathrm{opt}}(v', k-1)\,\mathrm{d}t \\ &= \Big(\rho_i(tr) + \tfrac{\rho_t(tr)}{\lambda_{tr}} + \sum_{v'\in V} \mu_{tr}(v')\tilde{\mathcal{R}}^{\sup}_{\mathrm{opt}}(v', k-1)\Big)\left(1 - \mathrm{e}^{-\lambda_{tr}\cdot\delta}\right)\end{aligned} \tag{7}$$

By using Eq. (6) instead of the Markovian part in Eq. (3) of the fixed point characterisation in Lemma 1, we get a discretised fixed point characterisation which is directly computable.

Definition 8 (Discretised maximum time-bounded reward). *Given an SRG $(\mathcal{G}, \rho_t, \rho_i)$, a time bound $T \geq 0$, and a step size $\delta > 0$ such that $T = k \cdot \delta$ for $k \in \mathbb{N}$. Let $\mathrm{opt} \in \{\inf, \sup\}$, $\mathrm{opt}_v = \sup$ if $v \in V_1$ and $\mathrm{opt}_v = \mathrm{opt}$ otherwise. $\mathcal{R}^{\sup}_{\mathrm{opt}}(\cdot, T)$ is the least fixed point of the higher order operator $\Omega^\delta_{\mathrm{opt}} : (V \times \mathbb{N} \mapsto \mathbb{R}^{\geq 0}) \mapsto (V \times \mathbb{N} \mapsto \mathbb{R}^{\geq 0})$ such that*

$$\Omega^\delta_{\mathrm{opt}}(F)(v,k) = \underset{tr\in\mathbf{T}(v)}{\mathrm{opt}_v} \begin{cases} \Big(\rho_i(tr) + \frac{\rho_t(tr)}{\lambda_{tr}} + \sum_{v'\in V} \mu_{tr}(v')F(v', k-1)\Big)\cdot\left(1 - \mathrm{e}^{-\lambda_{tr}\delta}\right) & \\ + \mathrm{e}^{-\lambda_{tr}\delta}F(v, k-1), & \text{if } v \in \mathbf{T}_M(v), \\ \rho_i(tr) + \sum_{v'\in V} \mu_{tr}(v')F(v', k), & \text{if } v \in \mathbf{T}_P(v). \end{cases} \tag{8}$$

Discretisation error. This section evaluates the precision of the discretisation technique described above. The discretisation technique can be applied to any kind of SRG respecting Def. 2. Its precision can be accordingly assessed for the general class of SRGs. However, opt-AISRGs obtained from MA abstraction have a special structure, namely all their Markovian transitions are controlled by player 2. For this specific structure it is usually possible to find a tighter error bound for time-bounded analysis (see for example [26]). Hence we restrict ourselves to a subclass of SRGs whose Markovian transitions (if there are any) are controlled by one player. In other words, the discretisation of models in this subclass in general introduces a smaller error compared to the general class of SRGs. The subclass is formally defined as:

Definition 9 (Single Markovian Controller SRG). *$\mathcal{G}_{rew} = (\mathcal{G}, \rho_t, \rho_i)$ is called a single Markovian controller SRG (1MC-SRG) iff either $\forall tr \in \mathbf{T}_M : tr \in \mathbf{T}_M(v) \Rightarrow v \in V_1$ or $\forall tr \in \mathbf{T}_M : tr \in \mathbf{T}_M(v) \Rightarrow v \in V_2$.*

The accuracy of $\tilde{\mathcal{R}}^{\sup}_{\mathrm{opt}}$ depends on some parameters including the step size δ. The smaller δ is, the better is the quality of the discretisation. In order to assess the accuracy of the discretisation we first need to define some parameters of SRGs.

Definition 10. *Given an SRG $\mathcal{G}_{rew} = (\mathcal{G}, \rho_t, \rho_i)$, we define the maximum (finite) exit rate existing in $\mathcal{G}$ as $\bar{\mathsf{e}} = \max_{tr \in \mathbf{T}_M} \lambda_{tr}$, and the maximum transient reward existing in ρ_t as $\bar{\mathsf{r}}_t = \max_{tr \in \mathbf{T}} \rho_t(\mathrm{tr})$. Moreover, let $\bar{\mathsf{r}}_i$ be the maximum instantaneous reward that can be earned between two consecutive Markovian jumps. This value can be efficiently computed via the Bellman equation given in [9, Theorem 1] after assigning zero value to all transient rewards.*

The following theorem quantifies the quality of the discretisation.

Theorem 2. *Given an 1MC-SRG $\mathcal{G}_{rew} = (\mathcal{G}, \rho_t, \rho_i)$, time bound $T > 0$ and discretisation step $\delta > 0$ such that $T = k\delta$ for some $k \in \mathbb{N}$. Then for all $v \in V$ we have:*

$$\tilde{\mathcal{R}}^{\sup}_{\delta,\mathrm{opt}}(v,k) \leq \mathcal{R}^{\sup}_{\mathrm{opt}}(v,T) \leq \tilde{\mathcal{R}}^{\sup}_{\delta,\mathrm{opt}}(v,k) + \tfrac{\bar{\mathsf{e}}T}{2}(\bar{\mathsf{r}}_t + \bar{\mathsf{e}}\bar{\mathsf{r}}_i)(1 + \tfrac{\bar{\mathsf{e}}T}{2})\delta$$

Using Theorem 2 it is possible to find a step size that respects a given predefined accuracy level. The proposed error bound is a linear approximation of the original bound (see [16]), which is a more complicated function with the same set of parameters. Since the original bound is tighter, in practice it is used for finding an appropriate step size by applying Newton's method.

3.3 Initial abstraction and labelling function

An important part of the abstraction and later the refinement process is starting with a suitable initial partition $\mathcal{P}$ and labelling function *lab*. On the one hand, if $\mathcal{P}$ is too coarse the resulting game abstraction requires many refinement steps until the desired accuracy is reached. On the other hand, if $\mathcal{P}$ is unnecessarily fine the resulting abstraction will not be able to reduce the size of the state space sufficiently.

A simple way to obtain a partition [12] is by exploiting the actions of the probabilistic transitions in an MA. They are used for synchronisation of different MA. The partition contains one block with all Markovian states and groups the probabilistic states according to the action labels available in a state. The labelling function is given by the action labels of the transitions.

However, experiments have shown that in such a partition transitions in the same block that are labelled with the same action may exhibit very different behaviour. Therefore they trigger refinement steps which can be avoided by a labelling function that takes the similarities of transitions into account. To achieve this, we first define a new initial partition: The Markovian states form one block of $\mathcal{P}$; the probabilistic states are grouped according to the number of outgoing transitions:

$$\mathcal{P} = \{MS\} \,\dot{\cup}\, \big\{\{v \in PS \,|\, |\mathbf{T}(v)| = |\mathbf{T}(v')|\} \,\big|\, v' \in PS\big\}\,.$$

Based on this partition we compute the labelling function *lab*, using a greedy strategy as follows: All Markovian transitions are labelled with $\perp$ as required by Def. 4. For the transitions of a probabilistic block $B \subseteq PS$ we proceed as follows:

We take an arbitrary state $v \in B$ (actually the first one in our list of states) and assign each transition $tr \in \mathbf{T}(v)$ a unique label $lab(tr) := \alpha_{tr}$. Running over the transitions $tr \in \mathbf{T}(v)$, we choose from each state $v' \in B$ with $v' \neq v$ a transition $tr' \in \mathbf{T}(v')$ which is not labelled yet and minimises $m_{\mathcal{P}}(\mu_{\text{tr}}, \mu_{\text{tr}'})$ defined as follows:

$$m_{\mathcal{P}} : \text{Distr}(V) \times \text{Distr}(V) \to [0, 2] \text{ with } m_{\mathcal{P}}(\mu, \mu') = \sum_{B' \in \mathcal{P}} \left| \mu(B') - \mu'(B') \right| .$$

The function $m_{\mathcal{P}}$ is a *pseudo-metric*[3] that measures the similarity of probability distributions with respect to a partition $\mathcal{P}$. Formally we take for each state $v' \in B$ with $v' \neq v$ an arbitrary

$$tr' \in \operatorname*{arg\,min}_{\substack{tr'' \in \mathbf{T}(v') \\ lab(tr'') \text{ is undefined}}} m_{\mathcal{P}}(\mu_{\text{tr}}, \mu_{\text{tr}''})$$

and set $lab(tr') := \alpha_{tr}$.

By labelling the transitions in this way we ensure that more similar probabilistic transitions belong to the same probabilistic $\overline{V_2}$-state, which prevents unnecessary splitting operations during refinement. Since all states within one probabilistic block B get the same set of labels with this labelling function, we do not need to introduce the $*$-state.

3.4 Refinement

We approximate the maximal ETR $\mathcal{R}^{\text{sup}}$ of an MRA $\mathcal{M}_{rew}$ by computing a lower and an upper bound $\tilde{\mathcal{R}}^{\text{sup}}_{\delta,\text{inf}}$ and $\tilde{\mathcal{R}}^{\text{sup}}_{\delta,\text{sup}}$ with a game abstraction $\overline{\mathcal{M}}^{\mathcal{P},lab}$ and the abstract reward functions $\overline{\rho_t}^{\text{opt}}$ and $\overline{\rho_i}^{\text{opt}}$. If these bounds are too far apart, i. e. $\tilde{\mathcal{R}}^{\text{sup}}_{\delta,\text{sup}} - \tilde{\mathcal{R}}^{\text{sup}}_{\delta,\text{inf}} > \varepsilon$, with ε being a precision threshold, our abstraction is too coarse and needs to be refined. The result of the refinement is a new partition, which in turn leads to a new game abstraction. This refinement-loop is repeated until the intended precision threshold ε is reached.

The reason behind the difference of the bounds can be two different situations: (1) The difference occurs due to different choices in the player 2 strategies σ_2^{inf} and σ_2^{sup}. (2) The difference is a result of the different reward structures. In case (1) we can use a strategy-based refinement strategy like in [12]. In case (2) we have to use a refinement strategy based on the reward values.

For this *value-based* refinement strategy we first have to search for a $\overline{V_2}$-state $\overline{v}$ where the values for $\tilde{\mathcal{R}}^{\text{sup}}_{\delta,\text{inf}}$ and $\tilde{\mathcal{R}}^{\text{sup}}_{\delta,\text{sup}}$ differ and the reward functions give different values. More precisely, this means that we have to search for an abstract transition $\overline{tr} \in \overline{\mathbf{T}}$ which was chosen by σ_2^{inf} and σ_2^{sup} and where $\left[\overline{\rho_i}^{\text{opt}}(\overline{tr}) + \frac{\overline{\rho_t}^{\text{opt}}(\overline{tr})}{\lambda_{\overline{tr}}}\right] = r_{\text{opt}}$, $\text{opt} \in \{\inf, \sup\}$, and $r_{\text{inf}} \neq r_{\text{sup}}$.

[3] The function $m_{\mathcal{P}}$ has the following properties: $m_{\mathcal{P}}(\mu, \mu') = m_{\mathcal{P}}(\mu', \mu)$, $m_{\mathcal{P}}(\mu, \mu'') \leq m_{\mathcal{P}}(\mu, \mu') + m_{\mathcal{P}}(\mu', \mu'')$, and $m_{\mathcal{P}}(\mu, \mu) = 0$ for all distributions μ, μ', μ''. However, it is not a metric because $m_{\mathcal{P}}(\mu, \mu') = 0$ does not imply that $\mu = \mu'$ holds.

If we have found such an abstract transition $\overline{tr}$, we split the preceding block B. For this we compute two sets of concrete states, one ($B_{r_{\inf}}$) containing the states with reward $r_{\inf}$, the other ($B_{r_{\sup}}$) containing the states with reward $r_{\sup}$: $B_{r_{\mathrm{opt}}} = \left\{ v \mid v \in B \wedge (v, \lambda, \mu) = tr \in \nu(\overline{tr}) \wedge \left[\rho_i(tr) + \frac{\rho_t(tr)}{\lambda_{tr}}\right] = r_{\mathrm{opt}} \right\}$ with $\mathrm{opt} \in \{\inf, \sup\}$.

In case of an abstract Markovian transition $\overline{tr}$ it may occur that $B_{r_{\mathrm{opt}}} = \emptyset$ since it is possible that no concrete transition $tr \in \nu(\overline{tr})$ matches the constraints. Should this happen for both $B_{r_{\mathrm{opt}}}$ we use the concrete transitions tr which optimise $\left[\rho_i(tr) + \frac{\rho_t(tr)}{\lambda_{tr}}\right]$ instead.

After we have generated the concrete state sets for the minimal and maximal reward, we replace B with $B_{r_{\inf}}$, $B_{r_{\sup}}$ and $B \setminus \left(B_{r_{\inf}} \dot{\cup} B_{r_{\sup}}\right)$.

Block B is replaced by at least two and at most three new blocks, which leads to a new, strictly finer partition and thus to a new game abstraction of $\mathcal{M}_{rew}$, which in turn can be analysed and refined. This refinement-loop is repeated until the precision threshold ε is reached.

Similar to [12] we apply a "precision trick", i. e. we start with a coarse, temporary precision threshold $\hat{\varepsilon}$ for the refinement-loop. If precision $\hat{\varepsilon}$ is reached, we switch to a higher precision, i. e. we lower $\hat{\varepsilon}$ and continue the refinement-loop. This process is repeated until the final precision ε is reached.

Zenoness. Although we assume the considered MRA $\mathcal{M}_{rew}$ is non-Zeno, i. e. it does not contain probabilistic end components, it may happen that Zenoness is introduced into the abstraction $\overline{\mathcal{M}}_{rew}^{\mathcal{P},lab}$. This occurs, e. g. if a non-cyclic sequence of probabilistic states is partitioned into the same block $B \in \mathcal{P}$. If the instantaneous reward function ρ_i is non-zero within the end component, the value iteration will not terminate since the accumulated reward does not converge.

We avoid this effect with the following method: Before applying value iteration to solve the discretised fixed point characterisation, we employ a standard graph algorithm [27] to search for end components in $\overline{\mathcal{M}}_{rew}^{\mathcal{P},lab}$. If a probabilistic end component is found, we refine the corresponding blocks B into smaller blocks and recompute the abstraction. This process is repeated until all probabilistic end components have been removed. Since there are no probabilistic end components present in $\mathcal{M}_{rew}$, our method will always terminate.

4 Experimental results

We implemented the described abstraction and refinement framework in C++ in a prototype tool called MeGARA. As mentioned earlier, we are currently concentrating on the maximal ETR only, using discretisation (see Sec. 3.2). We compare our experimental results with those of IMCA [10, 7, 9], an analyser for MA and IMCs. For our experiments we used the following case studies:

The **Polling System** (PoS) [7, 28] consists of two stations and one server. Incoming requests are stored within two queues until they are delivered by the

Table 1. Results for the polling system and the queueing system.

Name	IMCA			MeGARA				
	#states	r	time	#states	r_{lb}	r_{ub}	#iter.	time
PoS-3-2	1,547	0.830	0:04	657	0.828	0.830	17	0:02
PoS-3-3	14,322	0.922	1:06	5,011	0.920	0.921	19	1:16
PoS-3-4	79,307	0.985	10:59	11,527	0.982	0.985	20	7:02
PoS-4-2	6,667	0.832	0:20	1,513	0.829	0.831	18	0:12
PoS-4-3	131,529	0.924	13:46	**31,992**	0.922	0.923	22	8:55
PoS-5-2	27,659	0.833	1:47	**2,006**	0.830	0.832	18	0:23
QS-2	103	1.768	0:04	76	1.768	1.768	9	0:01
QS-3	163	2.307	0:10	118	2.306	2.306	10	0:02
QS-4	237	2.679	0:19	167	2.678	2.680	13	0:07
QS-8	673	3.351	1:47	455	3.351	3.351	17	0:33
QS-16	2,217	3.530	12:07	1,039	3.530	3.530	24	**2:33**
QS-32	7,993	TO		3,286	3.532	3.532	40	**57:43**

server to their station. With a probability of $p = 0.1$ a request erroneously stays in the queue after if it was delivered to a station. In our experiments we varied the queue size Q and the number of different request types J. The rewards in this case study represent costs for processing requests and consuming server memory. The model instances are denoted as "PoS-Q-J".

The **Queueing System** (QS) [6] stores requests within two queues of size K, each belonging to a server. Server S_1 handles requests and eliminates them from its queue. Requests processed by server S_2 are either nondeterministically submitted to the queue of S_1, or with probability $q = 0.3$ re-submitted to the queue of S_2, or with probability $(1 - q)$ eliminated from the queue. With the help of rewards we explore the average number of jobs in the queues. For our experiments we varied the queue size K. The model instances are denoted as "QS-K".

We created the model files with SCOOP [5], a modelling tool for MA. All experiments were done on a Dual Core AMD Opteron processor with 2.4 GHz per core and 64 GB of memory. Computations which took longer than one hour were aborted and are marked with "TO". Each computation needed less than 4 GB memory, we therefore do not present measurements of the memory consumption.

For all experiments we used time bound $T = 1$ and precision $\varepsilon = 0.01$ in order to compute the maximal ETR.

Table 1 shows the experimental results. The first column contains the name of the respective benchmark instance, the blocks titled "IMCA" and "MeGARA" present the results from IMCA and our abstraction refinement tool, respectively. The columns headed with "#states" give the number of states of the concrete system (in case of IMCA) and the final abstraction (in case of MeGARA). The benchmarks instances are relatively small since the solving of discretised systems is rather expensive [29].

Column "r" contains the maximal ETR computed by IMCA, whereas the columns "r_{lb}" and "r_{ub}" denote the lower and the upper bounds for the abstraction. The columns titled "time" present the total computation time (in format min:s) needed by IMCA and MeGARA, respectively. For our abstraction refinement tool we do not give more detailed time measurements since the better part of the computation time is needed for the repeated analysis of the abstraction, whereas the time needed for the refinement and re-computation of the abstraction is often negligible. For example, for QS-32 we need 57 min and 43 s, of which 57 min and 41 s are spent on the analysis, whereas the time spent on re-computing and refining the abstraction is less than 2 s.

For the instances of PoS our tool needs about the same amount of time as IMCA or is even faster, for the instances of QS we are always faster. As can be seen from the columns "r_{lb}" and "r_{ub}" the quality of our abstraction is always very good. In some cases the value "r" computed by IMCA is slightly higher than r_{ub}, however this deviation is always well within our precision threshold of $\varepsilon = 0.01$. We always achieve a notable compaction of the state space, even for the smallest instances with only a few hundred states. For the bigger instances we can report on compaction rates up to 92 %, e. g. for PoS-5-J we can reduce the system from 27,659 to 2,006 states.

5 Conclusion

We have presented a new abstraction technique for MRA, based on SRGs. We are able to analyse our abstraction regarding ETR properties. Should the quality of the abstraction be too low, we can apply scheduler- and value-based refinement methods. Our experiments show a significant compaction of the state space and a reduction of computation times.

In the future we plan to explore the possibilities of different initial partitions, labelling functions, and refinement techniques. We also plan to work on additional types of properties, e. g. bounded rewards or long-run average.

References

1. Eisentraut, C., Hermanns, H., Zhang, L.: On probabilistic automata in continuous time. In: Proc. of LICS, IEEE CS (2010) 342–351
2. Hermanns, H.: Interactive Markov Chains – The Quest for Quantified Quality. Vol. 2428 of LNCS. Springer (2002)
3. Eisentraut, C., Hermanns, H., Katoen, J.P., Zhang, L.: A semantics for every GSPN. In: Proc. of Petri Nets. Number 7927 in LNCS, Springer (2013) 90–109
4. Meyer, J.F., Movaghar, A., Sanders, W.H.: Stochastic activity networks: Structure, behavior, and application. In: Proc. of PNPM, IEEE CS (1985) 106–115
5. Timmer, M., Katoen, J.P., van de Pol, J., Stoelinga, M.: Efficient modelling and generation of Markov automata. In: Proc. of CONCUR. Vol. 7454 of LNCS, Springer (2012) 364–379
6. Hatefi, H., Hermanns, H.: Model checking algorithms for Markov automata. ECE-ASST **53** (2012)
7. Guck, D., Hatefi, H., Hermanns, H., Katoen, J.P., Timmer, M.: Modelling, reduction and analysis of Markov automata. In: Proc. of QEST. Vol. 8054 of LNCS, Springer (2013) 55–71
8. Guck, D., Hatefi, H., Hermanns, H., Katoen, J., Timmer, M.: Analysis of timed and long-run objectives for markov automata. Logical Methods in Computer Science **10**(3) (2014)

9. Guck, D., Timmer, M., Ruijters, E., Hatefi, H., Stoelinga, M.: Modelling and analysis of Markov reward automata. In: Proc. of ATVA. Vol. 8837 of LNCS, Springer (2014)
10. Guck, D., Han, T., Katoen, J.P., Neuhäußer, M.R.: Quantitative timed analysis of interactive Markov chains. In: Proc. of NFM. Vol. 7226 of LNCS, Springer (2012) 8–23
11. Kattenbelt, M., Kwiatkowska, M.Z., Norman, G., Parker, D.: A game-based abstraction-refinement framework for Markov decision processes. Formal Methods in System Design **36**(3) (2010) 246–280
12. Braitling, B., Ferrer Fioriti, L.M., Hatefi, H., Wimmer, R., Becker, B., Hermanns, H.: MeGARA: Menu-based game abstraction and abstraction refinement of Markov automata. In: Proc. of QAPL. Vol. 154 of EPTCS (2014) 48–63
13. D'Argenio, P.R., Jeannet, B., Jensen, H.E., Larsen, K.G.: Reduction and refinement strategies for probabilistic analysis. In: Proc. of PAPM-PROBMIV. (2002) 57–76
14. Wachter, B., Zhang, L.: Best probabilistic transformers. In: Proc. of VMCAI. Vol. 5944 of LNCS, Springer (2010) 362–379
15. Wachter, B.: Refined probabilistic abstraction. PhD thesis, Saarland University (2011)
16. Braitling, B., Ferrer Fioriti, L.M., Hatefi, H., Wimmer, R., Hermanns, H., Becker, B.: Abstraction-based computation of reward measures for Markov automata (extended version). Reports of SFB/TR 14 AVACS 106, SFB/TR 14 AVACS (2014) http://www.avacs.org.
17. Neuhäußer, M.R., Zhang, L.: Time-bounded reachability probabilities in continuous-time Markov decision processes. In: Proc. of QEST, IEEE CS (2010) 209–218
18. Ash, R.B., Doléans-Dade, C.A.: Probability & Measure Theory. 2nd edn. Academic Press (1999)
19. Neuhäußer, M.R.: Model checking nondeterministic and randomly timed systems. PhD thesis, RWTH Aachen University and University of Twente (2010)
20. Johr, S.: Model checking compositional Markov systems. PhD thesis, Saarland University, Germany (2008)
21. Shapley, L.S.: Stochastic games. Proceedings of the National Academy of Sciences of the United States of America **39**(10) (1953) 1095
22. Brázdil, T., Forejt, V., Krcál, J., Kretínský, J., Kucera, A.: Continuous-time stochastic games with time-bounded reachability. Information and Computation **224** (2013) 46–70
23. Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time markov chains. IEEE Trans. Software Eng. **29**(6) (2003) 524–541
24. Eisentraut, C., Hermanns, H., Zhang, L.: Concurrency and composition in a stochastic world. In: Proc. of CONCUR. Vol. 6269 of LNCS, Springer (2010) 21–39
25. Deng, Y., Hennessy, M.: On the semantics of Markov automata. Information and Computation **222** (2013) 139–168
26. Fearnley, J., Rabe, M.N., Schewe, S., Zhang, L.: Efficient approximation of optimal control for continuous-time markov games. In: Proc. of FSTTCS. Vol. 13 of LIPIcs, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik (2011) 399–410
27. Chatterjee, K., Henzinger, M.: Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification. In: Proc. of SODA, SIAM (2011) 1318–1336
28. Timmer, M., van de Pol, J., Stoelinga, M.: Confluence reduction for Markov automata. In: Proc. of FORMATS. Vol. 8053 of LNCS, Springer (2013) 243–257
29. Zhang, L., Neuhäußer, M.R.: Model checking interactive Markov chains. In: Proc. of TACAS. (2010) 53–68