On the Markovian Randomized Strategy of Controller for Markov Decision Processes

Taolue Chen[1,2,⋆], Tingting Han[2,3,4,⋆⋆], and Jian Lu[2,⋆⋆⋆]

[1] CWI, Department of Software Engineering, PO Box 94079, 1090 GB Amsterdam, The Netherlands

[2] Nanjing University, State Key Laboratory of Novel Software Technology, Nanjing, Jiangsu, P.R. China 210093

[3] RWTH Aachen, MOVES, Ahornstraße 55, D-52056 Aachen, Germany

[4] University of Twente, Faculty of EEMCS, FMT, PO Box 217, 7500 AE Enschede, The Netherlands

Abstract. This paper focuses on the so called *controller synthesis* problem, which addresses the question of how to limit the internal behavior of a given system implementation to meet its specification, regardless of the behavior enforced by the environment. We consider this problem in the probabilistic setting, where the underlying model has both probabilism and nondeterminism and the nondeterministic choices in some states are assumed to be controllable while the others are under the control of an unpredictable environment. As for the specification, it is defined by *probabilistic computation tree logic*. We show that under the restriction that the controller exploits only *Markovian randomized strategy*, the existence of such a controller is decidable, which is done by a reduction to the decidability of first-order theory for reals. This also gives rise to an algorithm which can synthesize the controller if it does exist.

1 Introduction

In this paper, we focus on *Markovian randomized* strategy of controllers for a *Markov Decision Process* (MDP for short) like model. Let us start by putting this problem in a more general setting: It is well-known that in the system design, to develop a system which satisfies the user requirement is one of the basic goals [1]. Undoubtedly, one hopes to automate this error-prone process as far as possible. To achieve this, one of the goals is to *synthesize* a system based on the requirements. However, this goal is usually too ambitious to be realized, and thus a more practical, but equally important task is to synthesize only a

⋆ This author is partially supported by the Dutch Bsik project BRICKS (Basic Research in Informatics for Creating the Knowledge Society).

⋆⋆ This author is partially supported by the Dutch NWO project QUPES (Verification of Quantitative Properties of Embedded Software).

⋆⋆⋆ This author is partially supported by 973 Program of China (No. 2002CB312002), NNSFC (No. 60233010, No. 60273034, No. 60403014), 863 Program of China (No. 2005AA113160, No. 2004AA112090).

controller which can limit or control the behaviors of an existing system (in terms of control theory, this is usually called *plant*), to meet the given specification. This is often referred to as the *controller synthesis* problem.

In such a framework, the plant acts usually in an *environment*. The goal is to find a *schedule* for the controllable events that guarantees the specification to be satisfied considering all possible environmental behaviors. One can also understand the controller and environment as two *players*, and thus take a game-theoretical point of view on the controller synthesis problem (in the below, we take the liberty to switch between these two points of views). The plant constitutes the game board and controller synthesis becomes the problem of finding a *winning strategy* for the controller that satisfies the specification whatever move the environment does, or in other words, under any adversary [1]. Correspondingly, the requirement specification is the *winning condition* in terms of game theory [11], i.e. the controller wins once the specification is satisfied. We note that the winning condition can be given either internally or externally: the former often imposes restrictions, for instance, on the number of visits of a state in the plant, typical examples include Büchi or Muller condition. The latter is usually temporal logic formulas which are supposed to be satisfied by the controller plant.

As a fundamental problem in control theory and computer science, the controller synthesis problem has attracted a lot of attentions in recent years. For discrete systems, it is well understood [14]. Recently, it also has been studied for timed systems and probabilistic systems. In this paper, we shrink our attention to the probabilistic setting, following [1]. Our underlying model for the plant is Markov Decision Processes [7][12]. However, different from the standard MDP model [12], here we adopt the point of view of [7] and distinguish states that are under control of the plant from those that are under the control of the environment. To put this in the game theory, this model is also known as *turn-based stochastic* $2\frac{1}{2}$*-player games* [4] (we note that the normal MDP is basically, turn-based stochastic $1\frac{1}{2}$-player games, see [4] for details). And by translating the player to other terms, one can construct a lot of examples of the similar spirit. Actually it is a very popular model in planning, AI, control problems, etc. In particular, it has been extensively applied for natural computation, fuzzy systems and knowledge discovery.

In this paper, we study the problem of finding a strategy for the plant such that a given external specification formalized as a *probabilistic temporal logic* formula is fulfilled, no matter how the opponent (environment) behaves. Concretely speaking, here the specification is given by a formula of *probabilistic computation tree logic* ([9], PCTL for short), which is one of the most influential and widely used probabilistic temporal logics. As for strategy, we follow [1], i.e. we consider the following choices:

- The system or the opponent has to choose deterministically (D) or randomly (R); and
- The system or the opponent chooses according to the current state (M, also called stationary or Markovian) or the history of the game played so far (H).

The combination gives rise to (at least) four classes of strategies, i.e. MD, HD, MR and HR strategies. In [1], it is shown that any of the strategy-classes (HD, HR, MD and MR) requires its own synthesis algorithm. Moreover, in that paper, the problem is only solved for MD strategies w.r.t. PCTL properties. However, the other three are left open (We quote a sentence from [1] which says "For other strategy types (MR, HD or HR), the complexity or even the decidability of the controller synthesis problem for PCTL is an open problem"). The aim of this paper is to attack this problem by considering MR strategy. The importance of MR strategy is justified by the fact that based on it, the controller can use random number generator rather than memory to make decisions. In practice, especially for embedded systems, it often leads to some "cheap" solution. It turns out even under this restriction, the problem is non-trivial. In the below, we discuss the main difficulties and our solutions in brief, from both the strategy and the specification perspectives:

– For strategy. As for the MD strategy, the controller synthesis problem can be trivially reduced to the model checking problem for Markov Chain w.r.t. PCTL specification. This is because there are only finitely many MD strategies for a given MDP, and thus one can try out all possibilities. Actually, in [1], this brute forth approach is used. However, for MR strategy, some more sophisticated approach has to be exploited since the total number of MR strategies is infinite (even uncountable since one can easily show the cardinality of the set of MR strategy is $\aleph_1$). To overcome this problem, we appeal to the deep results on the complexity of decision procedures for the first-order theory of reals $(\mathbb{R}, +, \cdot, \leq)$, which is well-known to be decidable [13]. To be more precise, we encode the existence of a MR-controller to $(\mathbb{R}, +, \cdot, \leq)$, thus prove the decidability of the existence of MR-controller.
– For specification. As we suggested before, in this paper we consider external specification, which is expressed by PCTL. In the game theory, this is rather difficult and a common approach to deal with external specification (winning condition) is to turn this into some internal one and at the same time, to transform the underlying model. For example, for linear-time properties, one can "encode" the specification by deterministic Muller automata and then take a product with the game graph (here it is MDP), thus the problem can be reduced to the one with a new MDP w.r.t. Muller winning condition (which is an internal winning condition). However, in the case of branching-time properties considered in this paper, it is not obvious how to adapt this approach, since in order to "encode" PCTL, we have to introduce some notion like "probabilistic tree automata", which has not been well studied yet, due to the knowledge of the authors. Fortunately, we find that our idea to exploit the decidability of $(\mathbb{R}, +, \cdot, \leq)$ can also be used to circumvent this difficulty.

It is worth emphasizing that our underlying model essentially agrees with $2\frac{1}{2}$-player game. However, we allow the two players to use different sorts of strategies in the sense that on the controller's aspect, it is only allowed to take MR strategy while on the environment aspect, we do *not* exert any restriction on the strategy which it can take. We note that this situation is akin to the notion of "modulo checking" [10] which deals with the problem of checking whether a module behaves correctly no matter in which environment it is placed. Actually, the interested reader will detect that in the second step of our algorithm (see Section 3), we implicitly encode the "module checking" problem for MDP into $(\mathbb{R}, +, \cdot, \leq)$. We believe this kind of "asymmetry" makes our result stronger and more general, and it is more useful in some applications. Moreover, it should be noticed that some related work has appeared in [4][5]. In both of papers, essentially the same problem is considered. However, the difference lies in that they considered linear-time specification (which can be regarded as internal winning condition), while we consider branching time specification. As we have suggested before, for controller synthesis problem, branching time specification is much more involved.

The structure of this paper is as follows: Section 2 summarizes some background material on MDP, strategy and PCTL. Section 3 presents our algorithm for MR strategy, and a simple example from [1] is given. Due to space restriction, in this extended abstract, most of proofs and practical examples are omitted and we refer the interested readers to the technical report version of this paper for more details.

2 Preliminaries

Definition 1 (Distribution). A *distribution* on a countable set X denotes a function $\mu : X \to [0, 1]$ such that $\sum_{x \in X} \mu(x) = 1$. We use $\mathrm{Distr}(X)$ to denote the set of all distributions on X.

Definition 2 (Markov Decision Process). A *Markov Decision Process* is a tuple $\mathcal{M} = (S, Act, \mathbb{P}, s_{in}, AP, L)$ where

- S is a countable set of states;
- Act is a finite set of actions;
- $\mathbb{P} : S \times Act \times S \to [0, 1]$ is a three-dimensional transition probability matrix such that for all states $s \in S$ and actions $a \in Act$, $\sum_{t \in S} \mathbb{P}(s, a, t) \in \{0, 1\}$;
- $s_{in} \in S$ is the initial state;
- AP denotes a finite set of atomic propositions;
- $L : S \to \wp(AP)$ is a labelling function which assigns to each state $s \in S$ the set $L(s)$ of atomic propositions that are (assumed to be) valid in s.

For technical reasons, we require that none of the states is terminal, i.e. for each state s, there exists an action a and a state s' with $\mathcal{P}(s, a, s') > 0$. $\mathcal{M}$ is called finite if the state space S is finite. For $T \subseteq S$, $\mathcal{P}(s, a, T) = \sum_{t \in T} \mathcal{P}(s, a, t)$ denotes the probability for s to move to a T-state, provided that action a has been selected in state s. We write $\mathcal{I}_{\mathcal{M}}(s)$ or $\mathcal{I}(s)$ (if $\mathcal{M}$ is clear from the context) for the action set $\{a \in Act \mid \mathcal{P}(s, a, S)\}$.

Definition 3 (Path and Trace). A *path* in $\mathcal{M}$ is a finite or infinite alternating sequence of states and actions $\sigma = s_1 a_1 \cdots a_n s_n$ or $\sigma = s_1 a_1 s_2 a_2 \cdots$ such that

$\mathcal{P}(s_i, a, s_{i+1}) > 0$. We use $\sigma[i]$ to denote the i-th state of σ. For finite path σ, we use $\sigma[\downarrow]$ to denote the last state of σ. Furthermore, we denote by $Path^*$ (resp. $Path^\omega$) the set of finite (resp. infinite) paths of a given MDP and by $Path^*(s)$ (resp. $Path^\omega(s)$) the set of finite (resp. infinite) paths of a given MDP starting at the state s.

For infinite path σ, we use $tr(\sigma)$ (*trace* of σ) to denote the infinite word over the alphabet $\wp(AP)$ which arises from σ by the projection of the induced state-sequence to the sequence of the labelings. For instance, if σ is defined as above, then $tr(\sigma) = L(s_1)L(s_2)\cdots \in (\wp(AP))^\omega$.

In the sequel, we assume that $\mathcal{M}$ is a finite MDP and S_0 a nonempty subset of S consisting of the states which are under the control of the system, i.e. where the system may decide which of the possible actions is executed. The states in $S \setminus S_0$ are controlled by the environment.

Definition 4 (Strategy). A *strategy* of $(\mathcal{M}, S_0)$ is an instance D that resolves the nondeterminism in the S_0 states. We distinguish four types of strategies:

– An MD-strategy is function $D : S_0 \to Act$ such that $D(s) \in \mathcal{I}(s)$;
– An MR-strategy is function $D : S_0 \to \mathrm{Distr}(Act)$ with $D(s) \in \mathrm{Distr}(\mathcal{I}(s))$;
– An HD-strategy is function D that assigns to any finite path σ in $\mathcal{M}$ with $\sigma[\downarrow] = s \in S_0$ and action $D(\sigma) \in \mathcal{I}(s)$;
– An HR-strategy is function D that assigns to any finite path σ in $\mathcal{M}$ with $\sigma[\downarrow] = s \in S_0$ and action $D(\sigma) \in \mathrm{Distr}(\mathcal{I}(s))$;

We note that the MD-strategy is often called *simple* or *purely memoryless* strategy. We refer to the strategies for the environment as *adversaries*. Formally, for X $\in \{\mathrm{MD}, \mathrm{MR}, \mathrm{HD}, \mathrm{HR}\}$, an X-adversary for $(\mathcal{M}, S_0)$ denotes a X-strategy for $(\mathcal{M}, S \setminus S_0)$. The notion of *policy* will be used to denote a decision rule that resolves both the internal nondeterministic choices (to be resolved by the controller) and the nondeterministic choices (to be resolved by the environment). We will use D for strategy, E for adversary and C for policies. Policy will often be written as $C = (D, E)$.

MDPs and Markov Chains Induced by Strategies. Any strategy D for $(\mathcal{M}, S_0)$ induces a MDP $\mathcal{M}_D$ which arises through unfolding $\mathcal{M}$ into a tree-like structure where the nondeterministic choices in S_0-states are resolved according to D. If $S_0 = S$ and $D = C$ is a policy for $\mathcal{M}$ then *all* nondeterministic choices are resolved in $\mathcal{M}_C$. For any HR-policy C, the MDP $\mathcal{M}_C$ is an infinite-state discrete-time Markov chain. If C is a stationary policy, then $\mathcal{M}_C$ can be regarded as a *Discrete Time Markov Chain* (DTMC for short) with state space S. If C is a policy for $\mathcal{M}$, then we write $\mathbb{P}^C_{\mathcal{M}}$ or just $\mathbb{P}^C$ (if $\mathcal{M}$ is clear from the context) to denote the standard probability measure on $\mathcal{M}_C$. Moreover, we use $Path^*_C(s)$ (resp. $Path^\omega_C(s)$) to denote the set of finite (resp. infinite) paths of the Markov chain induced by MDP $\mathcal{M}$ and policy C. Similarly, $Path^*_C(s)$ (resp. $Path^\omega_C(s)$) the set of finite (resp. infinite) paths of the corresponding Markov chain starting at the state s.

2.1 Probabilistic Computation Tree Logic (PCTL)

PCTL, the probabilistic extension of CTL (computation tree logic), was defined by Hansson and Jonsson [9] and is one of the most widely used probabilistic temporal logics. Let $AP = \{p, q, \dots\}$ be a countable infinite set of *atomic propositions*. The syntax of PCTL is given by the following BNF:
State formula:

$$\Phi ::= \top \mid p \mid \neg\Phi \mid \Phi \wedge \Phi \mid [\varphi]_{\bowtie r}$$

Path formula:

$$\varphi ::= \bigcirc\Phi \mid \Phi\mathcal{U}\Phi$$

where $p \in AP$ and $r \in [0, 1]$.

The most interesting part is $[\varphi]_{\bowtie r}$, which intuitively asserts that the probability measure of the paths satisfying φ meets the bound given by $\bowtie r$. The path modalities $\bigcirc$ (next step) and $\mathcal{U}$ (until) have the same meaning as in CTL. Other boolean connectives and the temporal operations $\Diamond$ (eventually) and $\Box$ (always) can be derived as a standard way. In particular, we set $\bot \overset{\text{def}}{=} \neg(\top)$.

Semantics. Given a MDP $\mathcal{M}$ as before, the formal definition of the satisfaction relation $\models$ for PCTL path and state formulas is defined as

$$\begin{array}{lll}
\mathcal{M}, s \models \top & & \\
\mathcal{M}, s \models p & \Leftrightarrow & p \in L(s) \\
\mathcal{M}, s \models \neg\Phi & \Leftrightarrow & s \not\models \Phi \\
\mathcal{M}, s \models \Phi_1 \wedge \Phi_2 & \Leftrightarrow & s \models \Phi_1 \text{ and } s \models \Phi_2 \\
\mathcal{M}, s \models [\varphi]_{\bowtie r} & \Leftrightarrow & \text{for } \textit{all} \text{ policies } C,\ \mathbb{P}^C_{\mathcal{M}}(\{\pi \in Path^\omega(s) \mid \pi \models \varphi\}) \bowtie r \\
 & & \\
\mathcal{M}, \pi \models \bigcirc\Phi & \Leftrightarrow & \pi[1] \models \Phi \\
\mathcal{M}, \pi \models \Phi_1\mathcal{U}\Phi_2 & \Leftrightarrow & \exists j \geq 1. (\pi[j] \models \Phi_2 \wedge \forall 1 \leq i < j. \pi[i] \models \Phi_1)
\end{array}$$

For any formula Φ (resp. φ), the (standard) *closure* of formula $cl(\Phi)$ (resp. $cl(\varphi)$) contains formulas whose truth values can influence the truth value of Φ (resp. φ).

3 Controller Synthesis for PCTL

Let us recall that the *controller synthesis* problem discussed in this paper is formalized by triples $(\mathcal{M}, S_0, Spec)$ where $\mathcal{M}$ is a finite MDP, S_0 a set of controllable states in $\mathcal{M}$ and *Spec* a temporal logical formula. The question is to find a strategy D for $(\mathcal{M}, S_0)$ such that *Spec* holds for the MDP $\mathcal{M}_D$, no matter how the environment (adversary) behaves. As shown in [1], the role of the strategy-type for controller synthesis is completely different from the situation in PCTL model checking, which has been addressed in [2]. While a single algorithm suffices for PCTL model checking, for controller synthesis, any strategy type requires its own synthesis algorithm. In the rest of this section, we focus on MR-strategy.

Definition 5. For any MDP $\mathcal{M}$, $s \in S$, and PCTL state-formula Φ, we define $Path^{\omega}_{C}(s, \Phi) = \{\sigma \in Path^{\omega}_{C}(s) \mid \sigma \models \Phi\}$ and furthermore

- $\mathbb{P}^{+}_{s}(\Phi) \overset{\text{def}}{=} \max\{\mathbb{P}(Path^{\omega}_{C}(s, \Phi)) \mid C \in \text{MD}\}$;
- $\mathbb{P}^{-}_{s}(\Phi) \overset{\text{def}}{=} \min\{\mathbb{P}(Path^{\omega}_{C}(s, \Phi)) \mid C \in \text{MD}\}$.

Where, C is ranged over by policies.

The satisfaction relation for PCTL does not depend on the chosen policy type because maximal and minimal probabilities for PCTL-path formulas under all HR-policies are reached with *simple* policies, as shown in [2]. This fact is stated by the following theorem, which is one of the cornerstones of our algorithm.

Theorem 1. *([2][6]) For any MDP $\mathcal{M}$, $s \in S$, and PCTL path-formula φ, the following properties hold:*

1. *$s \models [\varphi]_{\bowtie r}$ where $\bowtie \in \{\leq, <\}$ if and only if $\mathbb{P}^{+}_{s}(\varphi) \bowtie r$;*
2. *$s \models [\varphi]_{\bowtie r}$ where $\bowtie \in \{\geq, >\}$ if and only if $\mathbb{P}^{-}_{s}(\varphi) \bowtie r$.*

Before presenting our approach in a formal way, we give an overview of the main ideas. Basically, we will construct a closed formula of $(\mathbb{R}, +, \cdot, \leq)$ such that this formula is valid if and only of there exists an MR-controller for the given MDP $\mathcal{M}$ w.r.t. specification Φ, which is a PCTL state-formula. Let us denote the aforementioned MR-controller D, then the formula is of the form

$$\exists D.\mathcal{M}_D, s_{in} \models \Phi$$

Note here $\mathcal{M}_D$ is also an MDP with controllable sets attributed to the environment. Our main challenge is how to encode $\mathcal{M}_D \models \Phi$ in $(\mathbb{R}, +, \cdot, \leq)$, which turns out to be non-trivial and includes several subtle tricks. Now, let us start our construction in the following steps. Note that for convenience, we abuse the notation a little in the sense that for any first-order variable set $X = \{X_1, \cdots, X_n\}$ and formula ψ, we write $\exists X \psi$ as an abbreviation for $\exists X_1 \cdots \exists X_n \psi$.

***Step* 1**

For each state $s \in S_0$ and action $a \in \mathcal{I}(s)$, we introduce a first-order variable $X_{s,a}$. Intuitively, $X_{s,a}$ denotes the probability of choosing the action a in the state $s \in S_0$, thus it represents the strategy D. Then $\exists D.\mathcal{M}_D \models \Phi$ turns out to be

$$\begin{aligned} &\exists \{X_{s,a} \mid s \in S_0, a \in \mathcal{I}(s)\} \\ &\quad \textstyle\bigwedge_{X_{s,a}} (0 \leq X_{s,a} \leq 1) \wedge \bigwedge_{s \in S_0} (\sum_{a \in \mathcal{I}(s)} X_{s,a} = 1) \wedge \mathcal{M}_D, s_{in} \models \Phi \end{aligned} \tag{1}$$

***Step* 2**

The main goal of **Step 2** is to encode $\mathcal{M}_D, s_{in} \models \Phi$. To this end, for every $\psi \in cl(\Phi)$ and $s \in S$, we introduce a first-order variable $Y_{s,\psi}$, which ranges over $\{0, 1\}$. That is, these variables are essentially boolean values and we set

$$Y_{s,\psi} = 1 \text{ iff } s \models \psi$$

However, it is not easy to express $Y_{s,\psi} = 1$ in an inductive way. We then construct formula $W_{s,\psi}$ for every $s \in S$ and $\psi \in cl(\Phi)$, which can be defined inductively on the structure of ψ. Intuitively, $W_{s,\psi}$ denotes $s \models \psi$ and thus we have $Y_{s,\psi} = 1 \Leftrightarrow W_{s,\psi}$. It follows that (1) can be refined to

$$\begin{array}{l} \exists\{X_{s,a} \mid s \in S_0, a \in \mathcal{I}(s)\}. \\ \quad \bigwedge_{X_{s,a}}(0 \le X_{s,a} \le 1) \wedge \bigwedge_{s\in S_0}(\sum_{a\in\mathcal{I}(s)} X_{s,a} = 1) \Rightarrow \\ \qquad \exists\{Y_{s,\psi} \mid s \in S, \psi \in cl(\Phi)\}. \\ \qquad\quad \bigwedge_{Y_{s,\psi}}(Y_{s,\psi} = 0 \vee Y_{s,\psi} = 1) \wedge (Y_{s,\psi} = 1 \Leftrightarrow W_{s,\psi}) \wedge (Y_{s_{in},\Phi} = 1) \end{array} \tag{2}$$

Clearly, the remaining thing is to construct the formula $W_{s,\psi}$, which is the most difficult task. As we mentioned before, this will be done in an inductive way according to the structure of ψ. We proceed by a case analysis on the form of ψ.

1. $\psi = p$. If $p \in L(s)$, then $W_{s,\psi} \overset{\text{def}}{=} \top$ else $W_{s,\psi} \overset{\text{def}}{=} \bot$;
2. $\psi = \neg\psi_1$. Then $W_{s,\psi} \overset{\text{def}}{=} (Y_{s,\psi_1} = 0)$;
3. $\psi = \psi_1 \wedge \psi_2$. Then $W_{s,\psi} \overset{\text{def}}{=} (Y_{s,\psi_1} = 1) \wedge (Y_{s,\psi_2} = 1)$;
4. $\psi = [\bigcirc\psi_1]_{\bowtie r}$. We have two cases:
   – $s \in S_0$. Then
   $$W_{s,\psi} \overset{\text{def}}{=} \sum_{a\in\mathcal{I}(s),t\in S} X_{s,a} \cdot \mathbb{P}(s,a,t) \cdot Y_{t,\psi_1} \bowtie r$$
   – $s \in S \setminus S_0$. Then we have to distinguish the following two subcases according to $\bowtie$.
   - $\bowtie \in \{\le, <\}$. Then by Theorem 1(1), $W_{s,\psi} \overset{\text{def}}{=} \mathbb{P}^+_s(\bigcirc\psi_1) \bowtie r$. According to Definition 5, it is easy to see that
     $$\mathbb{P}^+_s(\bigcirc\psi_1) = \max_{a\in\mathcal{I}(s)}\{\sum_{t\in S}\mathbb{P}(s,a,t) \cdot Y_{t,\psi_1}\}$$
     We can find a solution for the above equation by solving the following linear program:
     $$\begin{array}{l} \min x_s \\ \text{s.t. } \ x_s \ge \sum_{t\in S}\mathbb{P}(s,a,t)\cdot Y_{t,\psi_1} \text{ for any } a \in \mathcal{I}(s) \end{array}$$
     We then introduce first-order variables Z_s for $s \in S \setminus S_0$. It follows that $W_{s,\psi}$ is defined as
     $$\exists\{Z_s \mid s \in S \setminus S_0\}.(0 \le Z_s \le 1) \wedge \bigwedge_{a\in\mathcal{I}(s)} (Z_s \ge \sum_{t\in S}\mathbb{P}(s,a,t)\cdot Y_{t,\psi_1})$$
     $$\wedge(Z_s \bowtie r) \wedge (\forall\{Z'_s \mid s \in S \setminus S_0\}.(0 \le Z'_s \le 1) \wedge \bigwedge_{a\in\mathcal{I}(s)} (Z'_s \ge$$
     $$\sum_{t\in S}\mathbb{P}(s,a,t)\cdot Y_{t,\psi_1}) \Rightarrow (Z'_s \ge Z_s))$$
   - If $\bowtie \in \{\ge, >\}$. This is the duality of the previous subcase.

5. $\psi = [\psi_1 \mathcal{U} \psi_2]_{\bowtie r}$. This case is the most involved one. However, the basic idea remains similar as the previous case. We also have the following two cases:
   - $s \in S_0$. Then clearly it follows that

$$W_{s,\psi} \overset{\text{def}}{=} \begin{cases} \top & \text{if } Y_{s,\psi_2} = 1 \\ \bot & \text{if } Y_{s,\psi_1} = 0 \\ Y_{s,\psi_1} = 1 \wedge \sum_{a \in \mathcal{I}(s), t \in S} X_{s,a} \cdot \mathbb{P}(s,a,t) \cdot Y_{t,\psi} \bowtie r & \text{o.w.} \end{cases}$$

   It is easy to transform this definition into a normal first-order formula in $(\mathbb{R}, +, \cdot, \leq)$ as follows:

$$\begin{aligned} &(Y_{s,\psi_2} = 1 \Rightarrow \top) \wedge (Y_{s,\psi_1} = 0 \Rightarrow \bot) \wedge \\ &(Y_{s,\psi_2} = 0 \wedge Y_{s,\psi_1} = 1 \Rightarrow Y_{s,\psi_1} = 1 \wedge \sum_{a \in \mathcal{I}(s), t \in S} X_{s,a} \cdot \mathbb{P}(s,a,t) \cdot Y_{t,\psi} \bowtie r \end{aligned}$$

   - $s \in S \setminus S_0$. As in the previous case, we have to distinguish the following two subcases according to $\bowtie$.
     - $\bowtie \in \{\leq, <\}$. Then $W_{s,\psi} \overset{\text{def}}{=} \mathbb{P}_s^+(\psi_1 \mathcal{U} \psi_2) \bowtie r$. According to Definition 5, it is easy to see that

$$\mathbb{P}_s^+(\psi) = \begin{cases} 1 & \text{if } Y_{s,\psi_2} = 1 \\ 0 & \text{if } Y_{s,\psi_1} = 0 \\ \max_{a \in \mathcal{I}(s)} \{ \sum_{t \in S \setminus S_0} \mathbb{P}(s,a,t) \cdot \mathbb{P}_t^+(\psi) & \\ \qquad + \sum_{t \in S_0} \mathbb{P}(s,a,t) \cdot Y_{t,\psi} \} & \text{o.w.} \end{cases}$$

     However, different from the previous case, now it is difficult to reduce this problem to solving a pure linear programming, since in the definition, the case distinctions have to be involved. Fortunately, we can still borrow the same idea, because actually what we need is to express the linear inequation in $(\mathbb{R}, +, \cdot, \leq)$ rather than to find the solution concretely. By this observation, we set

$$\begin{aligned} &\min x_s \\ &\text{s.t. for any } a \in \mathcal{I}(s), \\ &\qquad \begin{cases} x_s = 1 & \text{if } Y_{s,\psi_2} = 1 \\ x_s = 0 & \text{if } Y_{s,\psi_1} = 0 \\ x_s \geq \sum_{t \in S \setminus S_0} \mathbb{P}(s,a,t) x_t + \sum_{t \in S_0} \mathbb{P}(s,a,t) Y_{t,\psi} & \text{o.w.} \end{cases} \end{aligned}$$

     For simplicity, for Z_s ($s \in S$), we define $\Im(Z_s, Z_t)$ as

$$\begin{aligned} \Im(Z_s, Z_t) \overset{\text{def}}{=} \; &(Y_{s,\psi_2} = 1 \Rightarrow Z_s = 1) \wedge \\ &(Y_{s,\psi_1} = 0 \Rightarrow Z_s = 0) \wedge \\ &(Y_{s,\psi_2} = 0 \wedge Y_{s,\psi_1} = 1 \\ &\quad \Rightarrow Z_s \geq \sum_{t \in S \setminus S_0} \mathbb{P}(s,a,t) Z_t + \sum_{t \in S_0} \mathbb{P}(s,a,t) Y_{t,\psi}) \end{aligned}$$

It follows that $W_{s,\psi}$ is defined as

$$\exists\{Z_s \mid S \setminus S_0\}.(0 \leq Z_s \leq 1) \wedge \Im(Z_s, Z_t) \wedge (Z_s \bowtie r)$$
$$\wedge(\forall\{Z'_s \mid S \setminus S_0\}.(0 \leq Z'_s \leq 1) \wedge \Im(Z_s, Z_t) \Rightarrow (Z'_s \geq Z_s))$$

- $\bowtie \in \{\leq, <\}$. This is the duality of the previous case.

This completes our construction for formula $W_{s,\psi}$.

With $W_{s,\psi}$ on hand, we can fill the gap in (2), which completes the construction of $\mathcal{M}_D, s_{in} \models \Phi$. □

The correctness of our algorithm can be ensured by the following theorem, whose proof is the "reverse" of our construction shown above and thus is omitted.

Theorem 2. *For MDP $\mathcal{M}$, PCTL formula Φ, there exists an MR-controller for $\mathcal{M}$ if and only if (2) holds.*

Thus, we have the following corollary concerning on the complexity of the algorithm, according to [8][3].

Corollary 1. *For MDP $\mathcal{M}$, specification Φ, the problem of deciding whether there exists an MR-controller is in EXPTIME. Moreover, if such a controller does exist, it can be effectively constructed.*

References

1. C. Baier, M. Größer, M. Leucker, B. Bollig, and F. Ciesinski. Controller synthesis for probabilistic systems. In *Proceeding of IFIP TCS'04.* Kluwer, 2004.
2. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proceeding of FSTTCS'95.* LNCS 1026, 499-513, Springer, 1995.
3. S. Basu, R. Pollack and M. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of ACM*, 43(6):1002-1045, 1996.
4. K. Chatterjee, M. Jurdzinski and T. Henzinger. Simple stochastic parity games. In *Proceeding of CSL'03.* LNCS 2803, 100-113, Springer, 2003.
5. K. Chatterjee, M. Jurdzinski and T. Henzinger. Quantitative simple stochastic partity games. In *Proceeding of SODA'04.* SIAM.
6. C. Courcoubetis and M. Yannakakis. The compleixity of probabilistic verification. *Journal of ACM*, 42(4):857-907, 1995.
7. J. Filar and K. Vrieze. *Competitive Markov Decision Processes.* Springer, 1997.
8. D. Grigoriev. Complexity of deciding Tarski algebra. *Journal of Symbolic Computation*, 5(1-2):65-108, 1988.
9. H. Hansson and B. Jonsson. A Logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5): 512-535, 1994.
10. O. Kupferman, M. Vardi and P. Wolper. Module checking. *Information and Computation.* 164(2): 322-344, 2001.
11. G. Owen. *Game Theory.* Academic Press, 1995.
12. M. Puterman. *Markov Decision Processes.* Wiley, 1994.
13. A. Traski. *A Decision Method for Elementary Algebra and Geometry.* Univ. of California Press, Berkeley, 1951.
14. W. Thomas. Infinite games and verification. In *Proceeding of CAV'03*, LNCS 2725, 58-64, Springer, 2003.