Analysis of Probabilistic Processes and Automata Theory

Kousha Etessami[1]

[1]School of Informatics
University of Edinburgh
10 Crichton Street
Edinburgh, EH8 9AB, Scotland, UK
email: kousha@inf.ed.ac.uk

2010 Mathematics Subject Classification: 68Q45

Key words: Markov chains, Markov decision processes, infinite-state systems, recursive Markov chains, multi-type branching processes, stochastic context-free grammars, probabilistic pushdown automata, quasi-birth-death processes, one-counter automata, ω-regular model checking.

Abstract. This chapter surveys some basic algorithms for analyzing Markov chains (MCs) and Markov decision processes (MDPs), and discusses their computational complexity. We focus on discrete-time processes, and we consider both finite-state models as well as countably infinite-state models that are finitely-presented. The analyses we will primarily focus on are hitting (reachability) probabilities and ω-regular model checking, but we will also discuss various reward-based analyses.

Although it may not be evident at first, there are fruitful connections between automata theory and stochastic processes. Firstly, and not surprisingly, ω-automata play a naturally important role for specifying ω-regular properties of sample paths (trajectories) of stochastic processes. Computing the probability of the event that a random sample path satisfies a given ω-regular property constitutes the (linear-time) model checking problem for probabilistic systems.

Secondly, it turns out that there are close relationships between classic infinite-state automata-theoretic models and classic denumerably infinite-state stochastic processes, even though these models were developed independently in separate mathematical communities. Roughly speaking, some classic stochastic processes share their underlying state transition systems with corresponding classic automata-theoretic models. Furthermore, exploiting these connections to automata theory is fruitful for the algorithmic analysis of such stochastic processes, and for their controlled MDP extensions. This holds even when the analyses are much simpler than model checking, such as computing (optimal) hitting probabilities.

A number of important infinite-state stochastic models connected with automata theory can be captured as (restricted fragments of) *recursive Markov chains* and *recursive Markov decision processes*, which are obtained by adding a natural recursion feature to finite-state MCs and MDPs. Key computational problems for analyzing classes of recursive MCs and MDPs can be reduced to computing the *least fixed point* (LFP) solution of corresponding classes of *monotone* systems of nonlinear equations. The complexity of computing the LFP for such equations is a intriguing problem, with connections to several areas of research in theoretical computer science.

1 Introduction

Markov chains are a fundamental mathematical model for systems that evolve randomly over time. They thus play a central role in stochastic modeling in many fields. In settings where in addition to stochastic behavior we also allow *control* (or non-determinism), so that the system state evolves partly randomly and partly based on decisions by a controller, the resulting model is called a Markov decision process (MDP). MDPs give rise to a variety of stochastic dynamic optimization problems, depending on what objective the controller wishes to optimize.

Historically, automata theory developed entirely separately from the theory of stochastic processes and stochastic optimal control, with each developed by a separate mathematical community having distinct motives. It turns out, however, that there are fruitful connections between these fields. In particular, a number of classic infinite-state automata-theoretic models, such as one-counter automata, context-free grammars, and pushdown automata, are in fact closely related to corresponding classic and well-studied countably infinite-state stochastic processes. Roughly speaking, such automata-theoretic models share the same (or, a closely related) underlying state transition system with corresponding classic stochastic processes.

Upon reflection, it should not be entirely surprising that this is the case. After all, Markov chains are nothing other than probabilistic state transition systems. In order for a class of infinite-state Markov chains to be considered important, it should not only model interesting real-world phenomena, but it should also hopefully be "analyzable" in some sense. Better yet, its analyses should have reasonable computational complexity. But these same criteria also apply to infinite-state automata-theoretic models: their relevance is at least partly dictated by whether we have efficient algorithms for analyzing them.

Clearly, we can not devise effective algorithms for analyzing arbitrary finitely-presented countably infinite-state transition systems. For example, Turing machines are clearly finitely presented, but we can not decide whether a Turing machine halts, i.e., whether we can reach the halting configuration from the start configuration. Furthermore, if we consider *probabilistic* Turing machines (PTMs), we easily see that there can not exist any algorithm that computes *any non-trivial approximation* of the probability that a given probabilistic Turing machine halts.

Researchers working on automata theory and on stochastic processes have, over time, arrived at related classes of "analyzable" infinite-state transition systems, and they have built automata-theoretic structure, or stochastic structure, upon them to suit their own purposes. Let us mention a couple of examples. Consider the derivation graph of a *context-free grammar* (CFG), in which states consist of sequences of terminals and non-terminals and with a simultaneous derivation law defining transitions between states, so all non-terminals in a sequence are expanded at once according to rules associated with those nonterminals. The state transition systems obtained this way are intimately related to the underlying state transition systems of *multi-type branching processes* (BP), a classic stochastic process ([33]). Basically, the transition system for the BP corresponding to a CFG is the quotient of the CFG's transition system under the equivalence that equates any two sequences of terminals and nonterminals that contain the same number of occurrences of each nonterminal symbol in them (see [28] for a detailed explanation).

Likewise, *one-counter automata* share essentially the same state transition system

with *quasi-birth-death processes* (QBDs) (see [21] for the details). QBDs are a class of stochastic processes heavily studied in queuing theory, where the counter can basically be used to keep track of the number of jobs in the queue. A generalization of QBDs, referred to as *tree-like QBDs* in the queuing theory literature, turns out to share its state transition graphs with *pushdown automata* (again, see [21] for the precise correspondence).

The aforementioned stochastic models (in discrete-time) can all be formulated as subcases, in precise ways, of a model obtained by adding a natural recursion feature to finite-state Markov chains, called *recursive Markov chains* (RMCs) [28]. RMCs are also essentially equivalent to *probabilistic pushdown systems* [17] (see [28] for the precise sense of this equivalence). RMCs and RMDPs constitute natural abstract models of the control flow of probabilistic procedural programs with recursion.

Of course, being analyzable as automata does not automatically imply that the corresponding class of probabilistic transition systems or MDPs is also analyzable, nor the other way around. For some classes of transition systems, effective/efficient "analyzability" does coincide in the two settings, whereas for others it does not. We shall see examples of both.

This chapter surveys some basic algorithmic results for the analysis of Markov chains (MCs) and Markov decision processes (MDPs), in both finite-state settings, as well as in finitely presented countably-infinite state settings. We will consider a few different analyses, focusing on computation of hitting (reachability) probabilities and on model checking. But we will also discuss important reward-based analyses. We will also emphasize computational complexity considerations for the relevant problems. Finally, we shall very briefly mention the extension from MDPs to stochastic games and give some references to the relevant literature.

Algorithmic analyses of MCs and MDPs, including transient analyses, steady state analyses, optimal reward analyses, and model checking, play an important role in many application areas. A sampling of the many application areas where stochastic modeling and analysis play a role includes: queueing theory, computational biology, natural language processing, verification, economics, finance, and operations research in general.

Automata-theoretic models and methods come into play for analysis of stochastic systems in several ways. To begin with, we can view a Markov chain as a probabilistic state transition system (or probabilistic automaton). For *model checking* of MCs (and, respectively, MDPs), one is interested in determining the (optimal) probability with which a random walk on the MC (respectively, on the MDP using a chosen strategy for the controller) satisfies a given temporal property. The temporal property may be specified, for example, as a Linear Temporal Logic (LTL) formula, or as an ω-automaton. In the latter case the connection to automata theory is very direct: the properties are given by automata, or formalisms closely related to automata, so automata-theoretic methods are largely unavoidable.

Even for classic analyses of MCs and MDPs, as already indicated, there are deeper connections between the transition graphs of models studied originally in automata theory, such as context-free grammars, one-counter automata, and pushdown automata, and classic stochastic models that have been studied extensively in the stochastic processes literature over many decades, such as (multi-type) branching processes and (quasi-)birth-death processes.

Recently, these connections have been exploited to develop efficient algorithms for

analyzing such stochastic models, and to obtain results about the computational complexity of such analyses. We will survey some of this work. The literature on analysis of such Markov chains and MDPs is large and growing, even when restricted to aspects involving automata-theoretic connections. Thus, in this brief survey I can only hope to cover a very limited selection of the many models and algorithms. We will restrict our attention entirely to finite or countable-state discrete-time Markov chains (MCs) and MDPs.

After providing some basic background, in Section 2.1 we will define formally a number of important analysis problems for MCs and MDPs, and discuss carefully the different computation and decision problems that they give rise to, and we give some examples of analyses on finite-state MCs and MDPs in section 2.2, to help build the intuition of the reader. We then proceed in subsequent sections to discuss algorithms for and complexity of these analyses, beginning in section 3, then proceeding in section 4 to finite-state MDPs. We then define *recursive Markov chains* and *recursive MDPs* in section 5. As already discussed, these recursive models subsume a number of stochastic models and MDPs which have tight automata-theoretic connections. We then briefly discuss algorithms and complexity of analyzing RMCs and RMDPs, and provide pointers to the by now large relevant literature.

One of the themes that will emerge in this survey is that for key analyses of both finite-state MCs and MDPs, as well as for analysing classes of infinite-state recursive MCs and MDPs, a basic ingredient in their algorithms will be to find a solution to a corresponding system of equations. In the case of MDPs, these equations correspond to the appropriate *Bellman optimality equations* for the classes of MDPs involved. In particular, in several settings we will need to find the *least fixed point* (least non-negative) solution to a *monotone* system of equations. As the models become richer, these systems of equations become richer and more involved, e.g., going from linear to non-linear and requiring richer sets of algebraic operators (e.g., going from operators $\{+\}$ to $\{+, \max\}$, or to $\{+, *\}$, and then to $\{+, *, \max\}$, etc.). The computational complexity of finding solutions to such systems of equations, which turn out to be very intriguing problems with interesting connections to several areas of research, are thus intimately connected to the computational complexity of basic analysis problems for such stochastic models.

Finally, although we do not have room to discuss it in this chapter, let us briefly mention that one can also study the complexity of analysis problems for the extension of MDPs to *stochastic games*. In particular, in a two-player zero-sum stochastic game, there is not just one controller, but also an *adversary*, whose objective is the opposite of that of the controller. In *turn-based* stochastic games, also referred to as *simple stochastic games* (SSGs), and first studied by Condon [10], the two players control different states. Condon [10] showed that deciding whether the value is $\geqslant 1/2$ for a given SSG with the objectives of maximizing and minimizing the probability of hitting a target state for the two adversarial players is in NP$\cap$co-NP, and it is a major open problem whether this problem can be decided in P-time. (The problem is well known to be at least as hard as solving *parity games* and *mean payoff games*; see e.g. [46].) Although we shall not have room to discuss it this survey, we note that, again, key computational questions for stochastic games boil down to finding a solution for certain equation systems, and again, these equations become richer as the class of stochastic game models becomes richer, for example, going from $\{+, \max, \min\}$ to $\{+, *, \max, \min\}$, and to $\{+, \mathsf{Val}\}$ and to $\{+, *, \mathsf{Val}\}$, where Val is the *value* operator $\mathsf{Val}(M)$ that gives the minimax value of a 2-player zero-sum matrix

game with matrix M. Note that Val clearly generalizes both $\max$ and $\min$. Equations over $\{+, \mathsf{Val}\}$ were already used by Shapley [39] to characterize the value of his original 2-player zero-sum stochastic games (which, in the parlance used in this paper, constituted stochastic games with a discounted total payoff objective). Shapley's discounted equations for these defined a *contraction* mapping whose Banach fixed point gives the value of the stochastic game starting at each state. In other settings, e.g., in (concurrent) stochastic reachability (hitting) games, the equations define a *monotone* mapping whose Tarski *least fixed point* defines the value vector (note that Val). These games further generalize to infinite-state recursive settings and require monotone equations over $\{+, *, \mathsf{Val}\}$ for their value [25]. The reader interested in learning more about the stochastic game extensions of some of the models we discuss in this chapter can consult [30, 27, 22, 18].

Warning: This chapter is *certainly not* a comprehensive survey of algorithms for analysis and verification of Markov chains and MDPs and their connections to automata theory. These are vast and rapidly growing subjects, with a huge existing theoretical and practical literature. No comprehensive survey is feasible now, and it is not our intention to attempt one. This chapter only highlights a few basic topics, based largely on the author's own research interests, focusing on some connections between probabilistic processes and automata theory, and on recent research on algorithms for analyzing infinite-state recursive probabilistic systems. We do not mention many important related subjects. For example, we do not discuss existing software tools for analysis and model checking of probabilistic systems. There are many; see, e.g. [34]. Also, some software already exists for analysing recursive probabilistic systems; see, e.g., [44]. We also do not mention verification of probabilistic models against *branching-time* temporal logics like PCTL (see, e.g., Chapter 10 of [2] for one treatment of this in a textbook). We also do not discuss probabilistic (bi)simulation and related topics (again, see Chapter 10 of [2] for a brief treatment of this). There are many other topics related to both algorithms for analysis of probabilistic processes and to automata theory that we shall not mention at all.

2 Definitions and Background

Although we will endeavour to provide most of the formal definitions needed for our purposes, our subject is vast and we will need to assume some familiarity with basic notions and facts from probability theory, the theory of Markov chains, and the theory of Markov decision processes. For background on these topics the reader is referred, for example, to the following excellent textbooks [9, 36, 38].

Recall that a *σ-algebra* over a set Ω is a set $\mathcal{F} \subseteq 2^\Omega$ of subsets of Ω, such that $\Omega \in \mathcal{F}$, and such that $\mathcal{F}$ is closed under countable union and under complementation with respect to Ω. Recall that a *probability space*, $(\Omega, \mathcal{F}, \mathbb{P})$, consists of a set of *outcomes*, Ω (i.e., the *sample space*), a σ-algebra $\mathcal{F} \subseteq 2^\Omega$ of *events* over Ω, and a *probability measure*, $\mathbb{P} : \mathcal{F} \to [0, 1]$. For a real-valued random variable (r.v.), $X : \Omega \to \mathbb{R}$, over a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, the *expected value* of X, when it exists, is denoted $\mathbb{E}(X) \doteq \int_\Omega X \, d\mathbb{P}$. Note that, when $\mathbb{E}(X)$ is defined, $\mathbb{E}(X) \in \overline{\mathbb{R}} \doteq [-\infty, +\infty]$. We will sometimes need to consider *extended-real*-valued r.v.'s, $X : \Omega \to \overline{\mathbb{R}}$, and their expectation. The theory

for these r.v.'s is readily available (see, e.g., [9]), and consists of natural extensions to the definitions for real-valued r.v.'s and their expectation. A *probability distribution* over a finite or countably-infinite set, U, is a function $F : U \to [0,1]$ such that $\sum_{u \in U} F(u) = 1$. The *support* of the distribution F is the set $\texttt{support}(F) := \{u \in U \mid F(u) > 0\}$.

Markov chains. We view a (denumerable, discrete-time, time homogeneous) *Markov chain* (MC) as being given by a pair, $\mathcal{M} = (S, P)$, consisting of a countable (or finite) set of states, S, and a probabilistic transition function $P : S \times S \to [0,1]$, such that for all $s \in S$, $\sum_{s' \in S} P(s,s') = 1$. P is also referred to as the *transition probability matrix* of $\mathcal{M}$, and for $s, s' \in S$ we often use the notation $P_{s,s'}$ as an alternative to $P(s,s')$. When $|S| = n$ is finite, we will indeed find it convenient to view P as an $(n \times n)$ matrix, and we will often find it convenient to view the countable (or finite) state set S as consisting of (an initial segment of) the positive integers $\mathbb{N}_+ = \{1, 2, \ldots\}$. P is thus, by definition, a *stochastic matrix*, meaning it is non-negative and all its rows sum to 1. We use $\Delta \subseteq S \times S$ to denote the underlying *transition relation* of the Markov chain $\mathcal{M}$, defined by $\Delta = \{(s,s') \mid P(s,s') > 0\}$. The state set S together with Δ defines the *underlying directed graph*, $G = (S, \Delta)$, of the Markov chain $\mathcal{M}$. For every $s \in S$, define $\texttt{successors}(s) = \{s' \mid (s,s') \in \Delta\}$. Clearly, for all $s \in S$, $\texttt{successors}(s) \neq \emptyset$, so all states have at least one successor in Δ. We use the notation $s \to s'$ as an alternative to $(s,s') \in \Delta$, and we use $s \overset{*}{\rightsquigarrow} s'$ to denote that (s,s') is in the transitive closure Δ^* of Δ, i.e., that there is a (possibly empty) directed path in G from s to s'. We use $s \overset{+}{\rightsquigarrow} s'$ (respectively, $s \overset{k}{\rightsquigarrow} s'$) to denote there is a directed path of positive length (respectively, of length k) from s to s'. The Markov chain is called *irreducible* if for all states $s, s' \in S$, $s \overset{*}{\rightsquigarrow} s'$ holds. In other words, irreducibility means the graph G has one strongly-connected component (SCC). Recall that an SCC is a maximal subset $C \subseteq S$ such that for all $s, s' \in C$, $s \overset{*}{\rightsquigarrow} s'$. The structure of the strongly-connected components of G plays an important role in the analysis of finite-state Markov chains $\mathcal{M}$. Particularly important are *bottom strongly-connected components* (BSCCs). A BSCC, $C \subseteq S$, of G is an SCC such that for all $s \in C$ there is no state $s' \notin C$ such that $s \overset{*}{\rightsquigarrow} s'$. For $s \in S$, we use P_s to denote the function $P_s : S \to [0,1]$ defined by $P_s(s') := P(s,s')$ for all $s' \in S$. Note that, for all $s \in S$, P_s defines a probability distribution on S.

A Markov chain $\mathcal{M} = (S, P)$, together with an *initial* probability distribution on states, $\mathcal{I} : S \to [0,1]$, defines a probability space $(\Omega, \mathcal{F}, \mathbb{P}_{\mathcal{I}})$ where the sample space $\Omega = S^\omega$ consists of the set of infinite *trajectories*, or *sample paths*, or *runs* of $\mathcal{M}$.[1] A trajectory $\pi = \pi_0 \pi_1 \ldots \in \Omega = S^\omega$ is simply an infinite word (ω-word) over the alphabet S. For a finite string $w \in S^*$, let $\texttt{C}_{\mathcal{M}}(w) := wS^\omega \subseteq \Omega$ denote the set of trajectories that have the string w as an initial prefix. The (Borel) σ-algebra $\mathcal{F} \subseteq 2^\Omega$ of measurable events associated with trajectories of the MC, $\mathcal{M}$, is the (unique) σ-algebra generated by (i.e., the smallest σ-algebra containing) all basic open sets or *basic cylinders*, given by $\{\texttt{C}_{\mathcal{M}}(w) \mid w \in S^*\}$. The probability measure $\mathbb{P}_{\mathcal{I}} : \mathcal{F} \to [0,1]$, which is parametrized by the initial distribution $\mathcal{I}$, is uniquely determined by specifying, as follows, the probabilities of all basic cylinders, $\texttt{C}_{\mathcal{M}}(w)$. Firstly, for the empty string $w = \epsilon$, we have $\texttt{C}_{\mathcal{M}}(\epsilon) = S^\omega = \Omega$, so of course we define $\mathbb{P}_{\mathcal{I}}(\texttt{C}_{\mathcal{M}}(\epsilon)) := 1$. For any non-empty string $w =$

[1] In the probability theory literature the word *run* is not often used to refer to sample paths. We use it here to highlight the close correspondence with the notion of runs in automata theory.

$w_0 w_1 \ldots w_k \in S^+$, where $w_i \in S$, $i = 0, \ldots, k$, $k \geqslant 0$, we define $\mathbb{P}_{\mathcal{I}}(\mathtt{C}_{\mathcal{M}}(w)) := \mathcal{I}(w_0) \cdot \prod_{i=1}^{k} P(w_{i-1}, w_i)$. This definition extends uniquely to all events in the σ-algebra $\mathcal{F}$. When the initial distribution $\mathcal{I}$ assigns probability 1 to a single state, s, we will sometimes use $\mathbb{P}_s$ instead of $\mathbb{P}_{\mathcal{I}}$ to denote the associated probability measure.

A more common formulation of Markov chains, encountered in the probability theory literature, is the following: a Markov chain $\mathcal{M}$, together with initial distribution $\mathcal{I}$, defines a *discrete-time stochastic process*, $(X_i : i \in \mathbb{N})$, consisting of a sequence of random variables $X_i : \Omega \to S$ over the probability space $(\Omega, \mathcal{F}, \mathbb{P}_{\mathcal{I}})$, where each X_i maps a trajectory, $\pi = \pi_0 \pi_1 \pi_2 \ldots \in S^\omega = \Omega$, to the i'th state along that trajectory, i.e., $X_i(\pi) := \pi_i$. Clearly, according to these definitions, $\mathbb{P}(X_0 = s) = \mathcal{I}(s)$, for all $s \in S$, and furthermore, $(X_i)_{i \in \mathbb{N}}$ satisfied the *Markov property*, i.e., for any finite sequence of states $s_0, s_1, \ldots, s_k, s_{k+1}$, where $k \geqslant 0$, we have:

$$\mathbb{P}(X_{k+1} = s_{k+1} \mid X_0 = s_0, \ldots, X_k = s_k) = \mathbb{P}(X_{k+1} = s_{k+1} \mid X_k = s_k) = P(s_k, s_{k+1})$$

Clearly, these properties also uniquely characterize the Markov chain $\mathcal{M}$ (and initial distribution $\mathcal{I}$), so they can alternatively be taken as the definition of the Markov chain.

Let us observe here that, for any finite-state MC, $\mathcal{M}$, with any initial distribution $\mathcal{I}$, with probability 1, a trajectory of $\mathcal{M}$ will eventually enter some *bottom strongly connected component* (BSCC) $C \subseteq S$ of G, and will forever thereafter stay in C. In other words, if the BSCCs of the underlying graph G of a finite-state MC, $\mathcal{M}$, are given by $C_1, C_2, \ldots, C_k$, then $\mathbb{P}_{\mathcal{I}}(\bigvee_{j=1}^{k} \exists t \geqslant 0 : \forall t' \geqslant t : X_{t'} \in C_j) = 1$.

We will sometimes wish to consider a *labelled Markov chain*, $\mathcal{M} = (S, P, l)$, where $l : S \to \Sigma$ is a mapping that assigns to each state $s \in S$ a symbol $l(s) \in \Sigma$ from some alphabet Σ. The labels on distinct states need not be distinct. Sometimes, we may wish to associate *rewards* (*payoffs*) to states, in which case the labeling function $l : S \to \Sigma$ assigns numerical values to states. For example, we may have $\Sigma = \mathbb{Z}$. We associate with every trajectory $\pi = \pi_0 \pi_1 \pi_2 \ldots \in S^\omega$ of $\mathcal{M}$, an ω-word $l(\pi) \in \Sigma^\omega$ over the alphabet Σ, defined by $l(\pi) \doteq l(\pi_0) l(\pi_1) l(\pi_2) \ldots$.

For a random variable $Y : \Omega \to \overline{\mathbb{R}}$ over the probability space $(\Omega, \mathcal{F}, \mathbb{P}_{\mathcal{I}})$ of trajectories generated by a Markov chain $\mathcal{M}$ with initial distribution $\mathcal{I}$, we use $\mathbb{E}_{\mathcal{I}}(Y) \doteq \int_\Omega Y \, d\mathbb{P}_{\mathcal{I}}$, to denote the *expected value* of Y, assuming it exists, parametrized by initial distribution $\mathcal{I}$. If $\mathcal{I}$ assigns probability 1 to a state s, then we typically write $\mathbb{E}_s(Y)$ instead of $\mathbb{E}_{\mathcal{I}}(Y)$.

Example 2.1. A simple example of a labeled finite-state Markov chain, $\mathcal{M}_1 = (S, P, l)$, with 6 states, $S = \{s_1, \ldots, s_6\}$, is depicted in Figure 1. This 6-state MC has the following transition probability matrix, $P = (P_{i,j})_{i,j \in \{1,\ldots,6\}}$:

$$P = \begin{pmatrix} 0 & 1/3 & 1/2 & 0 & 1/6 & 0 \\ 2/5 & 0 & 1/5 & 0 & 2/5 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Each state s has a label $l(s) \in \Sigma = \{a, b, c\}$, and these are depicted in red in Figure 1. So, for example, $l(s_1) = a$.

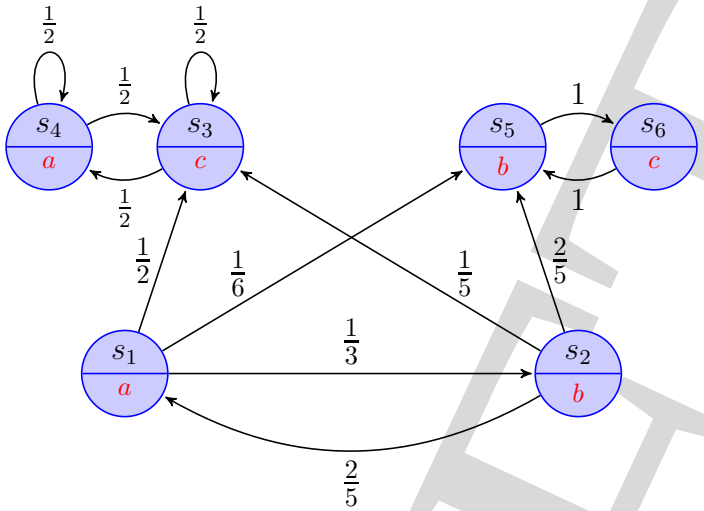

Figure 1. A simple 6-state labeled Markov chain, $\mathcal{M}_1$.

Let us consider *hitting probabilities*, in this MC. It is clear that in the MC $\mathcal{M}_1$, regardless of what state a trajectory starts in, with probability 1 the trajectory will eventually hit (reach) one of the two states s_3 or s_5, and will thereafter infinitely-often return to that state. Consider the *hitting* (or *reachability*) *probabilities*, $q^*_{i,j}$, where $q^*_{i,j}$ is defined as the probability of eventually hitting vertex s_j starting at vertex s_i, with $i, j = 1, \ldots, 6$. What, for example, is the probability $q^*_{1,3}$ for $\mathcal{M}_1$ in Figure 1? This hitting probability happens to be $17/26$. How can we compute it? We will come back to this question in section 3. For finite-state MCs, such probabilities can be computed easily by solving corresponding systems of linear equations. For this example, the probabilities $(q^*_{1,3}, q^*_{2,3})$ constitute the unique solution vector to the linear system of equations in two variables, (x_1, x_2), given by $x_1 = (1/3) * x_2 + (1/2)$; $x_2 = (2/5) * x_1 + (1/5)$. Hitting probabilities form a basic ingredient for many other kinds of analyses of MCs, including model checking. □

Markov decision processes. A (finite-state or countable-state) *Markov decision process (MDP)* is a tuple $\mathcal{D} = (S, (S_0, S_1), \Delta, P)$, where S is a (finite or countable) set of *states*; (S_0, S_1) is a partition of S into *random* states, S_0, and *controlled* states S_1, i.e., $S = S_0 \cup S_1$ and $S_0 \cap S_1 = \emptyset$; $\Delta \subseteq S \times S$ is a *transition relation*; and finally $P : S_0 \times S \to [0, 1]$ is a probabilistic transition function out of random states. For every $s \in S$, define $\texttt{successors}(s) = \{s' \mid (s, s') \in \Delta\}$. We assume that for all states $s \in S$, $\texttt{successors}(s) \neq \emptyset$, so all states have at least one successor in Δ. For each $s \in S_0$, we again use P_s to denote the function $P_s : S \to [0, 1]$ defined by letting $P_s(s') := P(s, s')$. We furthermore assume that for each $s \in S_0$, P_s defines a probability distribution (i.e., $\sum_{s' \in S} P_s(s') = 1$), and that $\texttt{support}(P_s) = \texttt{successors}(s)$. In other words, the transitions that are assigned positive probability are precisely transitions to those states that are immediate successors of s according to the transition relation Δ, and these probabili-

ties must of course sum to 1.

We will be focusing on either finite-state MDPs, or countably infinite-state MDPs that are finitely presented. Every specific family of MCs and MDPs that we consider is *finitely-branching*, meaning that for all $s \in S$, the set $\texttt{successors}(s)$ is finite. Indeed, all families of MDPs that we consider are *boundedly-branching*, meaning there is an integer $k \geqslant 1$ (depending on the MDP) such that for all $s \in S$, $|\texttt{successors}(s)| \leqslant k$.

An MDP represents a partially controlled stochastic process. The *controller* (a.k.a. *player*) exerts its control by choosing a *strategy* (a.k.a. *policy*, a.k.a. *scheduler*). A *strategy* (*policy*) is a function σ that, to each string $ws \in S^*S_1$ ending in a controlled state $s \in S_1$, assigns a probability distribution on the neighbors of s, $\sigma(ws) : \texttt{successors}(s) \to [0,1]$. We say that a strategy σ is *memoryless* if $\sigma(ws)$ depends only on the last vertex s. In this case we can denote the strategy by a function which assigns to every state $s \in S_0$ a probability distribution $\sigma(s) : \texttt{successors}(s) \to [0,1]$.

We say that a strategy σ is *deterministic* if for every $ws \in S^*S_1$, these is some $s' \in S$ such that $\sigma(ws)(s') = 1$, in other words, $\sigma(ws)$ assigns probability 1 to some neighbor of s. When σ is deterministic, we write $\sigma(ws) = s'$ instead of $\sigma(ws)(s') = 1$. Likewise, for a memoryless deterministic strategy σ, we write $\sigma(s) = s'$ instead of $\sigma(ws)(s') = 1$. Strategies that are not necessarily memoryless (respectively, deterministic) are called *history-dependent* (respectively, *randomized*).

Given an MDP, $\mathcal{D} = (S, \Delta, (S_0, S_1), P)$, fixing a strategy σ for the controller determines a unique Markov chain, $\mathcal{D}(\sigma) = (S^+, P^\sigma)$, for which the set of states is S^+ (i.e., the non-empty strings over S), and where, for all $w, w' \in S^*$ and $s, s' \in S$:

$$P^\sigma(ws, w's') := \begin{cases} P(s,s') & \text{if } s \in S_0, \text{ and } (s,s') \in \Delta, \text{ and } w' = ws \\ \sigma(ws) & \text{if } s \in S_1, \text{ and } (s,s') \in \Delta, \text{ and } w' = ws \\ 0 & \text{otherwise} \end{cases}$$

Note that states of $\mathcal{D}(\sigma)$ essentially store the entire history of states of $\mathcal{D}$ that are encountered during a run, starting from some initial state (or even some initial "history"). Let $\Omega(\sigma) = (S^+)^\omega$ denote the set of trajectories (sample paths) of the Markov chain $\mathcal{D}(\sigma)$. To every trajectory of $\mathcal{D}(\sigma)$ in $\Omega(\sigma) = (S^+)^\omega$, $\xi = (\xi_0)(\xi_1)\ldots \in \Omega(\sigma)$, we associate a corresponding *play*, $\pi \in S^\omega$. Namely, $\pi^\xi = \pi_0^\xi \pi_1^\xi \ldots \in S^\omega$, where if $\xi_i = w's$, for some $w' \in S^*$ and $s \in S$, then $\pi_i^\xi = s$. In other words, π_i^ξ is the state of S currently *visited* by the history ξ_i, i.e., it is the last "letter" of the string $\xi_i \in S^+$.

An MDP, $\mathcal{D}$, with an initial distribution $\mathcal{I} : S \to [0,1]$, and a strategy σ for the controller, together determine a probability space $(\Omega(\sigma), \mathcal{F}(\sigma), \mathbb{P}^\sigma_\mathcal{I})$ of trajectories of $\mathcal{D}(\sigma)$.

We also want to consider *labelled MDPs*, $\mathcal{D} = (S, \Delta, (S_0, S_1), P, l)$, where, again, $l : S \mapsto \Sigma$ assigns to each state a label from the alphabet Σ, and again the symbols in Σ may denote rewards, e.g., we may have $\Sigma = \mathbb{Z}$. Given an MDP, $\mathcal{D}$, and a strategy σ, we can also associate labels $l(ws)$ to the states $ws \in S^+$ of the resulting Markov chain $\mathcal{D}(\sigma)$, by lifting the labels from $\mathcal{D}$. Thus, if $ws \in S^+$, with $s \in S$, then we overload notation and let $l(ws) := l(s)$. We can then associate with every trajectory $\xi = \xi_0\xi_1\xi_2\ldots \in (S^+)^\omega$ of $\mathcal{D}(\sigma)$, an ω-word $l(\xi) \in \Sigma^\omega$, defined by $l(\xi) \doteq l(\xi_0)l(\xi_1)l(\xi_2)\ldots$. Likewise, we also associate the same word $l(\xi)$ to the corresponding play π^ξ, namely, we let $l(\pi^\xi) = l(\xi)$.

Given an MDP, $\mathcal{D}$, and an initial distribution $\mathcal{I}$, we will often be interested in the optimal probability of some family of events, $E(\sigma)$, parametrized by the strategy σ used by the controller, where $E(\sigma)$ is an event over the probability space $(\Omega(\sigma, \mathcal{F}(\sigma), \mathbb{P}^\sigma_\mathcal{I})$ of

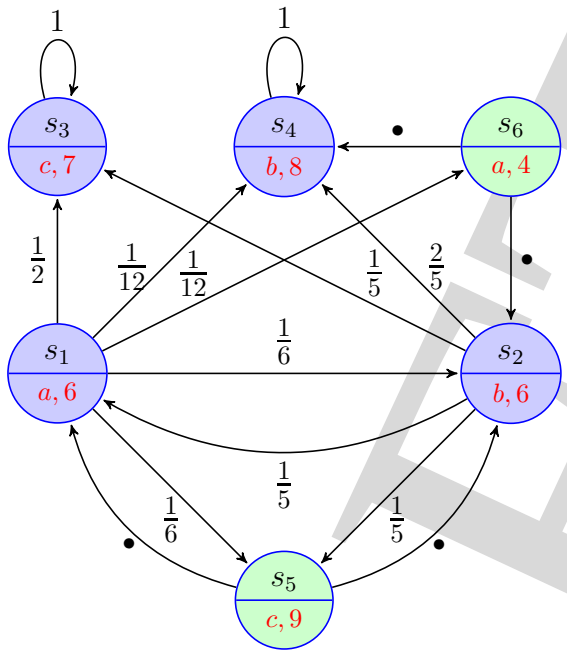

Figure 2. A 6-state labeled MDP, $\mathcal{M}_2$.

trajectories with initial distribution $\mathcal{I}$ generated by the strategy σ. For instance, the event $E(\sigma)$ could be the event specifying that the trajectory eventually hits a set of target states, or that the trajectory satisfies some temporal property. When it is clear from the context we usually simplify notation and refer to the family of events, $E(\sigma)$, as simply *the* event E, and we use the notation $\mathbb{P}^{\sigma}_{\mathcal{I}}(E)$ to denote the probability of event $E(\sigma)$ in the probability space of trajectories generated by $\mathcal{D}(\sigma)$. Likewise, we will often be interested in the optimal expected value of a family of random variables, $Y(\sigma) : \Sigma(\sigma) \to \overline{\mathbb{R}}$, parametrized by the strategy σ. When it is clear from the context, we will use $\mathbb{E}^{\sigma}_{\mathcal{I}}(Y) \doteq \int_{\Omega(\sigma)} Y \, d\mathbb{P}^{\sigma}_{\mathcal{I}}$ to denote the expected value of $Y(\sigma)$, parametrized by the strategy σ, in the probability space of trajectories generated by $\mathcal{D}(\sigma)$, and we refer to the family $Y(\sigma)$ of r.v.'s as simply *the* random variable Y. We shall consider several important analyses in Section 2.1.

Example 2.2. An example of a 6-state labeled MDP, $\mathcal{M}_2 = (S, (S_0, S_1), \Delta, P, l)$, with states $S = \{s_1, \ldots, s_5, s_6\}$, is depicted in Figure 2. The states are partitioned into a set of random states $S_0 = \{s_1, \ldots, s_4\}$, colored blue, and a set of controlled states $S_1 = \{s_5, s_6\}$, colored green. Note that the controller has only two choices at each of the two controlled state s_5 and s_6: from s_5 it can either move next to state s_1 or s_2, and from s_6 it can either move next to state s_2 or s_4.
For this MDP, each state s has *two* labels, one of which is a label $l(s) \in \Sigma$ from the alphabet $\Sigma = \{a, b, c\}$, and the other of which is a numerical label $r(s) \in \mathbb{Z}$. These two

labels are depicted in red in Figure 2. So, for example, $l(s_1) = a$ and $r(s_1) = 6$. [2]
Consider the *maximum (supremum) hitting probabilities*, $q^*_{\max,i,j} = \sup_\sigma \mathbb{P}^\sigma_{s_i}(\exists t \geqslant 0 : X_t = s_j)$, defining the supremum probability (over all strategies) of eventually hitting vertex s_j starting at vertex s_i, $i, j = 1, \ldots, 6$. What is $q^*_{\max,5,3}$ for the MDP $\mathcal{M}_2$ in Figure 2? The maximum hitting probability happens to be $q^*_{\max,5,3} = 3/4$, and the memoryless strategy that always chooses to move from state s_5 to state s_1, and from state s_6 to state s_2, achieves this optimal probability. Indeed, for finite-state MDPs, there is always a memoryless optimal strategy for maximizing (or minimizing) the probability of eventually hitting given target states. How can we compute such probabilities? We will come back to this question in section 4. For general finite-state MDPs these maximum (minimum) probabilities can be computed by solving corresponding systems of max(min)-linear *Bellman equations*. Such equations can be solved in polynomial time, using linear programming. Optimal hitting probabilities again form a basic ingredient for many other kinds of analyses of MDPs, including model checking. □

Quick review of Büchi automata, ω-regular languages, and Linear Temporal Logic. In order to discuss model checking problems for MCs and MDPs, we now review basic facts about, and fix notation for, ω-automata and linear temporal logic, which are topics covered in more detail in Chapter 6 of this Handbook ([43]). Two standard formalisms for specifying languages of ω-words, are Büchi automata and Linear Temporal Logic. A *Büchi automaton (BA)* is given by a tuple $\mathcal{B} = (Q, \Sigma, q_0, \delta, F)$, where Q is a finite set of states, Σ is a finite alphabet, $q_0 \in Q$ is an initial state, $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation, and $F \subseteq Q$ is a set of accepting states. We can assume without loss of generality (if necessary, by adding an extra dummy state) that the transition relation δ is *total* in the sense that for every state $q \in Q$, and every letter $a \in \Sigma$ of the alphabet, there is some state $q' \in Q$ such that there is a transition $(q, a, q') \in \delta$. The Büchi automaton is called *deterministic* if for every state q and every $a \in Q'$ there exists at most one state q' such that $(q, a, q') \in \delta$. Otherwise, it is *nondeterministic*. A *run* of $\mathcal{B}$ is a sequence $\pi = q_0 v_0 q_1 v_1 q_2 \ldots$ of alternating states $q_i \in S$ and letters $v_i \in \Sigma$, $i \geqslant 0$, such that for all $i \geqslant 0$ $(q_i, v_i, q_{i+1}) \in \delta$. The ω-word associated with run π is $\mathcal{L}(\pi) = v_0 v_1 v_2 \ldots \in \Sigma^\omega$. The run π is *accepting* if for infinitely many i, $q_i \in F$. We define the ω-regular language associated with $\mathcal{B}$ by $L(\mathcal{B}) = \{\mathcal{L}(\pi) \mid \pi \text{ is an accepting run of } \mathcal{B}\}$. Note that $L(\mathcal{B}) \subseteq \Sigma^\omega$.

It is well known that any ω-regular language can be described as the language of ω-words associated with a (nondeterministic) Büchi automaton. Indeed, we can take this as the definition ω-regular languages. However, unlike the fact that deterministic finite automata (DFAs) suffice to capture all regular languages over finite strings, *deterministic* BAs *do not* suffice for expressing all ω-regular languages. For example, the language of ω-words over the alphabet $\{a, b\}$ that contains only a finite number of b's can not be described by any deterministic BA. To capture all ω-regular languages using deterministic automata, we need more sophisticated acceptance conditions, like *Müller*, *Rabin*, *Streett*, or *Parity* acceptance conditions. (See Chapter 6: [43].)

In particular, the standard *subset construction*, which when applied to any (nondeterministic) finite automaton yields a deterministic finite automaton that accepts the same

[2] Of course we can also simply view the labels $l(s)$ as assigning to each state s a pair (y_s, z_s) consisting of a label from $y_s \in \Sigma = \{a, b, c\}$ and a payoff $z_s \in \mathbb{Z}$.

language of finite strings, does not work for ω-automata: it may yield an ω-automaton that accepts a strictly larger language of ω-words.

Remarkably however, it turns out that in a certain sense the standard subset construction *does work* for the purpose of model checking of ω-regular properties of labeled Markov chains and Markov decision processes. This is one of several key insights first revealed in the *tour-de-force* papers by Courcoubetis and Yannakakis [11, 12, 13, 14]. These papers also established the best complexity bounds available (and best possible, subject to complexity-theoretic assumptions), for model checking finite-state Markov chains and MDPs. We will highlight some of these results.

Another major insight in the Courcoubetis-Yannakakis papers relates to model checking Linear Temporal Logic properties of MCs. Recall that *Linear Temporal Logic* (LTL) [37] formulas are built from a finite set $\mathsf{Prop} = \{\mathsf{P}_1, \ldots, \mathsf{P}_k\}$. of propositions, using the usual Boolean connectives, $\neg, \vee$, and $\wedge$, the unary temporal connective *Next* (denoted $\bigcirc$) and the binary temporal connective *Until* (U); thus, if ξ, ψ are LTL formulas then $\bigcirc\,\xi$ and $\xi\ \mathsf{U}\ \psi$ are also LTL formulas, as are $\neg\xi$, $\xi \vee \psi$, as well as $\xi \wedge \psi$. This constitutes an inductive definition of temporal formulas. Note that other useful temporal connectives can be defined using U. The formula $\mathsf{True}\ \ \mathsf{U}\ \psi$ means "eventually ψ holds" and is abbreviated $\Diamond\,\psi$. The formula $\neg(\Diamond\neg\psi)$ means "always ψ holds" and is abbreviated $\Box\,\psi$.

An LTL formula specifies a language of ω-words over the alphabet $\Sigma = 2^{\mathsf{Prop}}$, as follows. If $w = w_0, w_1, w_2 \ldots \in \Sigma^\omega$ is an ω-word, and φ is an LTL formula, then first we define satisfaction of the formula by w *at position* i, where $i \geqslant 0$, denoted $(w, i) \models \varphi$. We define this inductively on the structure of the formula φ as follows.

- $(w, i) \models \mathsf{p}$ for $\mathsf{p} \in \mathsf{Prop}$ iff $\mathsf{p} \in w_i$.
- $(w, i) \models \neg\xi$ iff not $(w, i) \models \xi$.
- $(w, i) \models \xi \vee \psi$ iff $(w, i) \models \xi$ or $w, i \models \psi$.
- $(w, i) \models \xi \wedge \psi$ iff $(w, i) \models \xi$ and $w, i \models \psi$.
- $(w, i) \models \bigcirc\xi$ iff $(w, (i + 1)) \models \xi$.
- $(w, i) \models \xi\ \mathsf{U}\ \psi$ iff $\exists j \geqslant i : (\,(w, j) \models \psi$ and $\forall k\,(i \leqslant k < j) : (w, k) \models \xi\,)$.

The ω-language specified by an LTL formula φ, is $\mathcal{L}(\varphi) := \{w \in \Sigma^\omega \mid (w, 0) \models \varphi\}$. The language specified by every LTL formula is ω-regular, and in fact any LTL formula can be converted to an equivalent (albeit, exponentially bigger) nondeterministic Büchi automaton that accepts the same language (see, e.g., [42] and Chapter 6 [43]).

2.1 Some important analysis problems for MCs and MDPs

We now formally define a variety of important algorithmic analyses that one might wish to perform on MCs and MDPs. Given an MDP, $\mathcal{D}$, initial distribution $\mathcal{I}$, and strategy σ, let X_i denote the random variable that assigns to a trajectory ξ of the Markov chain $\mathcal{D}(\sigma)$, the state $X_i(\xi) = s_i \in S$ of $\mathcal{D}$ that is visited by the play at time i. (In other words, $X_i(\xi) = s_i$ if $\xi_i = ws_i$, for some $w \in S^*$ and $s_i \in S$.) The controller's goal is to optimize the (expected) value of some random variable, or the probability of some event, both of which could be a function of the entire random trajectory. There are a wide variety of objectives that have been studied in the MDP literature. We now list some important analyses that have been considered.

Note that all of the analyses listed below are also applicable to purely stochastic Markov chains, because MCs are just special cases of MDPs, where there are no controlled nodes. In other words, in an MC the controller has only one (vacuous) strategy, which is to do nothing.

I. MP: **Mean payoff**[3]: the labelling function l is a *payoff* function, $l : S \to \mathbb{Q}$, which associates to every state s a (rational valued) payoff $l(s) \in \mathbb{Q}$.[4] The goal of the controller is to maximize[5] the expected *mean payoff* of the play $\pi = s_0 s_1 s_2 s_3 \ldots$:

$$\mathbb{E}^{\sigma}_{\mathcal{I}}(\liminf_{n\to\infty} \frac{\sum_{i=0}^{n-1} l(X_i)}{n})$$

Note that in the case of irreducible finite-state MCs, mean payoff analysis subsumes, as a very special case, computation of the *invariant (stationary) distribution* of the MC. Recall, the invariant distribution for an irreducible MC, $\mathcal{M} = (S, P)$, with $S = \{1, \ldots, n\}$, is the unique probability distribution λ on states, given by a non-negative row vector $\lambda = (\lambda_1, \ldots, \lambda_n)$ with $\sum_i \lambda_i = 1$, such that $\lambda P = \lambda$. When the finite-state MC is *ergodic* (irreducible and aperiodic), the invariant distribution λ is the *steady-state* distribution, giving the long-run probability of being in any particular state, regardless of the initial distribution. Consider a state $j \in S$, and consider the following labeling of the states of $\mathcal{M}$ with payoffs: let $l(j) := 1$, and for all other states $j' \in S \setminus \{j\}$, let $l(j') := 0$. Then $\lambda_j = \mathbb{E}_j(\lim_{n\to\infty} \frac{\sum_{i=0}^{n-1} l(X_i)}{n})$.

II. DTP: **Discounted total payoff:** Given a payoff function $l : S \to \mathbb{Q}$ labelling the states, and given a rational **discount** factor $0 < \beta < 1$, the goal is to maximize the expected *discounted total payoff*:

$$\mathbb{E}^{\sigma}_{\mathcal{I}}(\lim_{n\to\infty} \sum_{i=0}^{n} \beta^i l(X_i))$$

The limit in the expression exists under mild conditions on the MDP (e.g., it suffices if the payoffs labeling states are bounded in absolute value). Discounted payoff objectives play an important role, e.g., in economics and finance, where the discount factor β can often be viewed as being given by the rate of inflation, i.e., the rate at which the present value of money depreciates over time.

III. NTP: **Non-negative total payoff:** There is no discount, the states are labeled by *non-negative* payoffs, $l : S \to \mathbb{Q}_{\geqslant 0}$. The goal is to either maximize or minimize the expected *total* reward, which may in general be $+\infty$:

$$\mathbb{E}^{\sigma}_{\mathcal{I}}(\lim_{n\to\infty} \sum_{i=0}^{n} l(X_i)) \tag{2.1}$$

Sometimes the structure of the MCs or MDPs implies that this expectation is finite.

[3] This objective is also known as the ***limiting-average payoff*** objective in the MDP literature.

[4] We restrict to rational payoffs in $\mathbb{Q}$, rather than payoffs in $\mathbb{R}$, for computational reasons. We wish to analyze the complexity of algorithms also in terms of the encoding size of the input coefficients.

[5] Note that maximizing expected mean payoff (or discounted payoff), when payoffs can be both positive and negative rational values, is computationally equivalent to minimizing expected mean payoff, because minimizing the mean payoff amounts to maximizing the mean payoff when all payoffs labeling states are negated.

Analyzing expected non-negative total reward includes, as a special case, analysis of the expected *hitting time* of a set of target states. Consider an MDP, $\mathcal{M} = (S, (S_0, S_1), P, \Delta)$, with a set $F \subseteq S$. Turn all target states in F into *absorbing* random states, meaning re-define the random states as $S_0' := S_0 \cup F$, and controlled states as $S_1' := S_1 \setminus F$, and let $P_{s,s} := 1$ for $s \in F$. Define the payoff labels at states as follows: for $s \in F$, let $l(s) := 0$, and for $s \in S \setminus F$, let $l(s) := 1$. Let H_F denote the random variable (family) defining the *hitting time* of the target set F. Then clearly, for every strategy σ, $\mathbb{E}^\sigma_{\mathcal{I}}(H_F) = \mathbb{E}^\sigma_{\mathcal{I}}(\lim_{n\to\infty} \sum_{i=0}^{n} l(X_i))$.

IV. HP: **Hitting probability of desired (or undesired) target states:** Given a set of target states $F \subseteq S$, the goal is to maximize (or minimize) the probability of eventually hitting a state $s \in F$. In other words, we wish to choose a strategy σ to maximize, or minimize: $\mathbb{P}^\sigma_{\mathcal{I}}(\exists i \geqslant 0 : \ X_i \in F)$.

Let us denote the supremum and infimum of these probabilities by

$$q^*_{\max,\mathcal{I},F} \equiv \sup_\sigma \mathbb{P}^\sigma_{\mathcal{I}}(\exists i : \ X_i \in F) \quad \text{and} \quad q^*_{\min,\mathcal{I},F} \equiv \inf_\sigma \mathbb{P}^\sigma_{\mathcal{I}}(\exists i : \ X_i \in F)$$

It need not in general be the case that there exists any optimal strategy σ^* such that $q^*_{\max,\mathcal{I},F} = \mathbb{P}^{\sigma^*}_{\mathcal{I}}(\exists i : \ X_i \in F)$; and likewise, for infinitely-branching MDPs, there need not exist any strategy σ^* such that $q^*_{\min,\mathcal{I},F} = \mathbb{P}^{\sigma^*}_{\mathcal{I}}(\exists i : \ X_i \in F)$. Indeed, one can easily construct examples of infinite-state MDPs where no optimal strategy exists for maximizing/minimizing the probability of hitting a set of target states.[6] In such cases, there only exist ϵ-optimal strategies, for every $\epsilon > 0$.

The objective of optimizing hitting probability can also be easily reformulated as a special case of NTP, i.e., of optimizing expected total non-negative reward, as follows: remove all out-going transitions from states in F, and replace them with a single transition from each state in F to a new state s^*. Let $\chi(s) = 1$ for all $s \in F$, and let $\chi(s) = 0$ otherwise. Then the goal of maximizing/minimizing the probability of eventually hitting the target states F is equivalent to the goal of maximizing/minimizing the undiscounted expected total non-negative payoff, when the payoffs labelling the states are given by χ.

However, the ability to label non-absorbing states with reward 0 is crucial for this. In fact, in some MDP settings, analysing expected total reward when all non-absorbing states are labeled by *strictly positive* rewards is substantially easier than analyzing hitting probability (see, in particular, [22] for an example).

V. MoCh: **Model checking of ω-regular or LTL properties.** Given a labelled MDP, $\mathcal{D} = (S, \Delta, P, l)$, and initial distribution $\mathcal{I}$, where $l : S \to \Sigma$, and given an ω-regular language $\mathcal{L}$ over the alphabet Σ, specified by giving a Büchi automaton $\mathcal{B}$ or LTL formula φ, so that $\mathcal{L} = L(\mathcal{B})$ or $\mathcal{L} = L(\varphi)$, the goal of the controller is to choose a strategy σ so as to maximize (or minimize) the probability that the trajectory π of $\mathcal{D}(\sigma)$ generates an ω-word $l(\pi) \in \mathcal{L}$. In other words, we can, associate with the ω-regular language $\mathcal{L}$, the corresponding event (family) $E_{\mathcal{L}}(\sigma) = \{\pi \in \Omega(\sigma) \mid l(\pi) \in \mathcal{L}\}$ in the probability space generated of trajectories of the MC $\mathcal{D}(\sigma)$ generated by the MDP $\mathcal{D}$ and the strategy σ. It can be checked that, regardless of what the

[6] In the case of minimization, such examples require infinitely-branching infinite-state MDPs, but for maximization, simple finitely-presented boundedly-branching infinite-state MDPs suffice to show that no optimal strategy for hitting the target states exists.

strategy σ is used, for any ω-regular language $\mathcal{L}$, the property $E_{\mathcal{L}}(\sigma)$ does indeed constitute an event in the σ-algebra $\mathcal{F}(\sigma)$. (This was noted already, e.g., in [41].)
When it is clear from the context, we overload notation and use $\mathcal{L}$ to refer to the event family $E_{\mathcal{L}}(\sigma)$, parametrized by the strategy σ. The goal of model checking[7] for MDPs is thus to maximize (or minimize) the probability $\mathbb{P}^{\sigma}_{\mathcal{I}}(\mathcal{L})$.

For analyses like HP and MoCh, which involve computing the (optimal) probability of some event, the associated computational problems can be further subdivided and classified as either *qualitative* or *quantitative* analyses, as we now discuss.

Sometimes we may not need to know the (optimal) probability of the event in question, and we may instead just be satisfied to know whether or not the event holds *almost surely*, i.e., with (maximum) probability 1, or equivalently whether the complement event has (infimum) probability 0. These constitute what are generally referred to as *qualitative analyses*, whereas *quantitative analyses* involve computing the (optimal) probability of the event in question. However, particularly for MDPs, there are subtle distinctions between different forms of qualitative analysis, and also between different forms of quantitative analysis. In some settings these distinctions can make a big difference in terms of the computational complexity of the problems involved. So we now examine these distinctions more carefully.

(1) **Qualitative analysis of MCs and MDPs:** Given an (MC or) MDP, $\mathcal{D}$, and an initial distribution, $\mathcal{I}$, for an event E (again, strictly speaking, a family of event, $E(\sigma)$, in the respective probability spaces of trajectories of the MCs, $\mathcal{D}(\sigma)$, parametrized by the strategy σ), and for set Ψ of strategies constraining the strategies that the controller may use (e.g., Ψ may simply be *all* strategies, or only *memoryless* ones, or *deterministic* ones, etc.), consider the following decision problem:

$$\text{Decide whether} \qquad \exists \sigma \in \Psi : \mathbb{P}^{\sigma}_{\mathcal{I}}(E) = 1. \tag{2.2}$$

This decision problem is referred to as the *qualitative* **almost-sure** *decision problem* for the event E (and with respect to the strategy constraint Ψ). This problem is of course equivalent to asking whether $\exists \sigma \in \Psi : \mathbb{P}^{\sigma}_{\mathcal{I}}(\overline{E}) = 0$, where $\overline{E} = \Omega \setminus E$ denotes the complement event. (Again, strictly speaking $\overline{E}(\sigma) = \Omega(\sigma) \setminus E(\sigma)$ is a family of events parametrized by σ.)
If such a strategy σ exists, then we may also want to compute (some representation of) such a strategy, in which case this is no longer just a decision problem.
A closely related, but in general *not* equivalent, problem is:

$$\text{Decide whether} \qquad \sup_{\sigma \in \Psi} \mathbb{P}^{\sigma}_{\mathcal{I}}(E) = 1. \tag{2.3}$$

This is referred to as the *qualitative* **limit-sure** *decision problem* for the event E.[8]

[7] In the context of MDPs, as phrased here, this is an optimization problem, and not a decision problem, so the word "model checking" is a bit of a misnomer. But we will adhere to this terminology.

[8] The term *limit-sure* was first used in [15], where they considered the distinct almost-sure and limit-sure decision problems in the context of concurrent (stochastic) reachability games. As we shall see, the distinction between almost-sure and limit-sure qualitative analyses is relevant in various other contexts, including for important classes of finitely-presented infinite-state MDPs.

Although the almost-sure and limit-sure decision problems are related, and although they are obviously equivalent if the model is simply a Markov chain, these problems are certainly not equivalent for all MDPs, because as already discussed in relation to HP, in general there need not exist any optimal strategy σ that achieves probability 1 for the event $\texttt{Hit}_F \doteq (\exists i : X_i \in F)$, and yet there may exist a sequence of strategies $\sigma_1, \sigma_2, \sigma_3, \ldots$, which achieve probabilities arbitrarily close to 1. For example, we could have $\mathbb{P}^{\sigma_i}_{\mathcal{I}}(\texttt{Hit}_F) = 1 - \frac{1}{2^i}$. In such a case, the limit-sure condition (2.3) holds while the almost-sure condition (2.2) does not.

We also in general need to consider, as distinct qualitative problems for MDPs, the following *duals* of the above problems, which are not in general equivalent, namely, decide whether: $\forall \sigma \in \Psi : \mathbb{P}^{\sigma}_{\mathcal{I}}(E) = 1$. This is of course the complement of deciding whether $\exists \sigma \in \Psi : \mathbb{P}^{\sigma}_{\mathcal{I}}(E) < 1$, which is equivalent to:

$$\text{Decide whether} \qquad \exists \sigma \in \Psi : \mathbb{P}^{\sigma}_{\mathcal{I}}(\overline{E}) > 0 \tag{2.4}$$

Note however that, in this dual setting, there is no distinction between the almost-sure and the limit-sure cases. The above problems are also equivalent to deciding whether $\inf_{\sigma \in \Psi} \mathbb{P}^{\sigma}_{\mathcal{I}}(E) < 1$, and to deciding whether $\sup_{\sigma \in \Psi} \mathbb{P}^{\sigma}_{\mathcal{I}}(\overline{E}) > 0$.

We refer to problem (2.4) as the *qualitative witness-positivity*[9] *decision problem* for (the family of) events $\overline{E}$.

Let us also mention some "*qualitative*" problems that can be associated with objectives such as NTP, where the objective is optimize the expected total non-negative payoff. It is possible, for example, that $\exists \sigma \in \Psi : \mathbb{E}^{\sigma}_{\mathcal{I}}(\lim_{n\to\infty} \sum_{i=0}^{\infty} l(X_i)) = +\infty$ holds true, or else that $\sup_{\sigma \in Psi} \mathbb{E}^{\sigma}_{\mathcal{I}}(\lim_{n\to\infty} \sum_{i=0}^{\infty} l(X_i)) = +\infty$. Again, the latter may hold true while the former does not, because there may be no optimal strategy. These problems are clearly analogous to the almost-sure and limit-sure qualitative decision problems for the probability of an event E. We will call them the *qualitative witness-infinity problem* and the *qualitative limit-infinity problem* for the expectation of the associated random variable (family) Y. In many settings, such "qualitative" problems are not relevant because the random variable Y is guaranteed to have bounded expectation. For example, this holds for finite-state MDPs with MP and DTP, namely mean payoff and discounted total payoff objectives.

(2) **Quantitative analysis of MCs and MDPs:**

Quantitative analysis problems can be considered for all of the problems (I.-V.) on on our list, and not just for those relating to the (optimal) probability of an event. In general, for quantitative analysis we want to compute the optimal (supremum or infimum) expected value of some random variable family Y or the optimal probability of some event family E.

However, it may not always be possible to compute the quantity in question exactly. This may be because of the computational complexity doing so. It may also be because of a more basic reason: in a variety of stochastic models we can consider, the optimal (supremum or infimum) value over all $\sigma \in \Psi$ may be *irrational*, even when all of the finite data describing the Markov chain or MDP consists of rational values. In such cases, we can still consider *approximating* the optimal value within some desired error bound, or *deciding* whether the optimal value is at least a given

[9] Or, *witness-less-than-one*, where appropriate.

rational value $r \in \mathbb{Q}$. Again, there are some subtle distinctions, so let us formulate these problems more precisely:

(a) *Quantitative decision problems:* Given an MDP, $\mathcal{D}$, and initial distribution $\mathcal{I}$, and some event (family) E, and given a rational value $r \in \mathbb{Q}$:

$$\text{Decide whether} \qquad \exists \sigma \in \Psi : \mathbb{P}^{\sigma}_{\mathcal{I}}(E) \geqslant r \tag{2.5}$$

Or, if the objective is to optimize the expected value of a r.v. Y, we may want to decide whether or not $\exists \sigma \in \Psi : \mathbb{E}^{\sigma}_{\mathcal{I}}(E) \geqslant r$.
Of course, if such a strategy σ exists, we may also wish to compute (some representation of) such a strategy. A different decision problem is:

$$\text{Decide whether} \qquad \exists \sigma \in \Psi : \mathbb{P}^{\sigma}_{\mathcal{I}}(E) \leqslant r \tag{2.6}$$

And analogously, decide whether $\exists \sigma \in \Psi : \mathbb{E}^{\sigma}_{\mathcal{I}}(E) \leqslant r$.
Note that decision problem (2.5) is concerned with the goal of *maximizing* the probability of the event E (or expectation of the r.v. Y): does there exist a strategy that obtains a value of *at least* r? Whereas, decision problem (2.6) is concerned with the goal of *minimizing* the probability of E (or expectation of r.v. Y): does there exist a strategy that obtains an value of *at most* r?
Sometimes, the above decision problems are too hard computationally, whereas the corresponding *approximation* problems are not as hard.

(b) *Quantitative ϵ-approximation problems:* We are given an MC or MDP, $\mathcal{D}$, and initial distribution $\mathcal{I}$, some event (family) E whose probability we are interested in, or a random variable (family) Y whose expectation we are interested in. Let $v^* = \sup_{\sigma \in \Psi} \mathbb{P}^{\sigma}_{\mathcal{I}}(E)$, or $v^* = \inf_{\sigma \in \Psi} \mathbb{E}^{\sigma}_{\mathcal{I}}(Y)$, in the respective cases. We are also given a rational positive error threshold $\epsilon > 0$. We wish to[10]:

$$\text{Compute an } \epsilon\text{-approximate value, } v \in \mathbb{Q}, \text{ such that } |v^* - v| < \epsilon. \tag{2.7}$$

We may then also wish to compute (a representation of) an *ϵ-optimal strategy*: a strategy σ' such that $|v^* - \mathbb{P}^{\sigma'}_{\mathcal{I}}(E)| < \epsilon$ or $|v^* - \mathbb{E}^{\sigma'}_{\mathcal{I}}(Y)| < \epsilon$, respectively.

2.2 More examples of analyses for finite-state MCs and MDPs

We now reconsider the example MC and MDP given in Figures 1 and 2, and consider other analyses for these.

Example 2.3. Let us consider again the labeled 6-state finite-state Markov chain, $\mathcal{M}_1 = (S, P, l)$, depicted in Figure 1, and let us consider some other analyses for that MC.
MoCh: Consider the following model checking problem. The LTL formula $\Box \diamond b$, expresses the property that the symbol b occurs infinitely often in the ω-word. What is the the probability $\mathbb{P}_{s_1}(L(\Box \diamond b))$? It is not difficult to see, by inspection of $\mathcal{M}_1$, that $\mathbb{P}_{s_1}(L(\Box \diamond b))$ is precisely equal to the probability of eventually hitting state s_5 starting in state s_1. In other words, $\mathbb{P}_{s_1}(L(\Box \diamond b)) = q^*_{1,5}$. Furthermore, since we know that starting from state s_1, with probability 1 we will eventually hit either state s_3 or s_5, i.e.,

[10] Note: such an ϵ-approximation may be impossible with $v \in \mathbb{Q}$, e.g., because $v^* = \sup_{\sigma \in \Psi} \mathbb{E}^{\sigma}_{\mathcal{I}}(Y) = \infty$.

that $q^*_{1,3} + q^*_{1,5} = 1$, and since we have already noted that $q^*_{1,3} = 17/26$, we can conclude that $\mathbb{P}_{s_1}(L(\Box \diamond b)) = 9/26$.

Note that in this case model checking was boiled down to computing hitting probabilities. The general algorithms for model checking Markov chains against ω-regular properties are much more involved, but as we shall see they also ultimately reduce the problem to computing hitting probabilities on certain associated Markov chains.

MP: Now let us use hitting probabilities to do *mean payoff* analysis on the MC, $\mathcal{M}_1$. In particular, suppose that the labels on states are associated with payoffs, as follows: $a := 4$, $b := -3$, $c := 7$. Let $v^*_i = \mathbb{E}_{s_i}(\liminf_{n\to\infty} \frac{\sum_{i=0}^{n-1} l(X_i)}{n})$ denote the expected mean payoff when starting in state s_i. In the MC $\mathcal{M}_1$, what is v^*_1? Let G_1 denote the underlying graph of $\mathcal{M}_1$. The two BSCCs of G_1 are $C_1 = \{s_3, s_4\}$ and $C_2 = \{s_5, s_6\}$. Clearly, starting in state s_1 of $\mathcal{M}_1$, with probability 1 we will eventually hit one of these two BSCCs and stay in that BSCC forever thereafter. We already know that we will eventually hit C_1 with probability $q^*_{1,3} = 17/26$, and that we will hit C_2 with probability $q^*_{1,5} = 9/26$. Note that the MC defined by restricting $\mathcal{M}_1$ to the nodes of BSCC C_1 is ergodic, and that its unique steady-state distribution is clearly $(1/2, 1/2)$. Likewise, although the MC defined by restricting $\mathcal{M}_1$ to the nodes of C_2 is not ergodic, it is irreducible, and its unique invariant distribution is $(1/2, 1/2)$. In other words, in the case of both BSCCs C_1 and C_2, once we enter such a BSCC, in the long run we spend $1/2$ the time in each of the two states of that BSCC. Thus v^*_1, the long-run mean payoff starting in state s_1, can be calculated via the following expression: $v^*_1 = (17/26) \times (1/2 \times 7 + 1/2 \times 4) + (9/26) \times (1/2 \times -3 + 1/2 \times 7) = 217/52$. □

Example 2.4. Now let us reconsider the 6-state labeled MDP, $\mathcal{M}_2 = (S, (S_0, S_1), \Delta, P, l)$, with states $S = \{s_1, \ldots, s_6\}$, depicted in Figure 2.

MoCh: Consider, in particular, the following model checking problem. What is the supremum probability $\sup_\sigma \mathbb{P}^\sigma_{s_5}(L(\Box \diamond b))$?

It is not difficult to see, by inspection of $\mathcal{M}_2$, that regardless what strategy σ is used, $\mathbb{P}^\sigma_{s_5}(L(\Box \diamond b))$ is precisely equal to the probability $\mathbb{P}^\sigma_{s_5}(\exists i : \ X_i = s_4)$ of eventually hitting state s_4 starting at state s_5. It can furthermore be seen that the probability of hitting state s_4 is maximized by the simple memoryless strategy, σ^*, that always moves to state s_2 whenever in state s_5, and always moves to state s_4 whenever in state s_6. And, furthermore the (maximum) probability that this strategy achieves of eventually hitting state s_4 is $13/22$. In other words, $\sup_\sigma \mathbb{P}^\sigma_{s_1}(\mathcal{M}_1 \models \diamond \Box b) = \mathbb{P}^{\sigma^*}(\mathcal{M}_1 \models \diamond \Box b) = q^*_{\max,5,4} = 13/22$.

This example is too simple in at least one sense: the maximum probability in this case is attained by a deterministic memoryless strategy, but in general for obtaining the maximum probability of an LTL or ω-regular property on a finite-state MDPs it need not suffice to use a deterministic memoryless strategy (in particular, memory may be required).

MP: Finally, let us consider the *mean payoff* objective on the MDP, $\mathcal{M}_2$, in Figure 2, where the aim is to maximize the expected limiting ($\liminf$ of the) average payoff per step, where the one-step reward at state s is given by the function $r(s)$. In other words, the aim is to maximize $\mathbb{E}^\sigma_{\mathcal{I}}(\liminf_{n\to\infty} \frac{\sum_{i=0}^{n-1} r(X_i)}{n})$. Note that in the MDP, $\mathcal{M}_2$, regardless of what strategy is employed by the controller, with probability 1 the trajectory will eventually enter one of the two states s_3 or s_4, and stay there forever thereafter. Once it is in

one of these two states, the (expected) limiting average payoff thereafter is simply the payoff at that state, which is $r(s_3) = 7$ for state s_3 and and $r(s_4) = 8$ for state s_4. Thus, since $r(s_4) > r(s_3)$, in order to maximize the expected mean payoff starting at any other state, we simply need to maximize the probability of eventually hitting state s_4. We already know from our previous calculations that, starting at state s_5, the maximum probability of eventually hitting state s_4 is $13/22$, and this is achieved by the deterministic memoryless strategy that always moves from state s_5 to state s_2, and from state s_6 to state s_4. Thus the maximum expected mean payoff is $(13/22) * 8 + (1 - 13/22) * 7 = (167/22)$, and this is achieved by the same deterministic memoryless strategy. For finite-state MDPs, it is always the case that there exists an optimal deterministic memoryless strategy for maximizing the expected limiting average payoff (see, e.g., Theorem 9.1.8 in [38]), and one can compute the optimal limiting average payoff, and an optimal memoryless strategy, in polynomial time using linear programming (see, e.g., chapters 8 and 9 of [38]). □

3 Analysis of finite-state Markov chains

In this section we review some algorithms for analyzing finite-state MCs, and discuss their complexity. Let us already summarize the known facts: for all of the analyses (I.-V.) listed in section 2.1, all qualitative and quantitative decision and computation problems are solvable in strongly[11] polynomial time, as a function of the encoding size of the given finite-state MC, $\mathcal{M}$. For qualitative analyses, the algorithms only involve graph-theoretic analysis of the underlying transition graph G of the MC, $\mathcal{M}$. For quantitative analyses, the algorithms additionally involve solving corresponding systems of linear equations. For model checking (MoCh) the complexity is polynomial in the encoding size of $\mathcal{M}$ but exponential in the encoding size of the ω-regular language, $\mathcal{L}$, and remarkably this is so whether $\mathcal{L}$ is specified by a non-deterministic Büchi automaton (BA), $\mathcal{B}$, or as an LTL formula φ (as shown by Courcoubetis and Yannakakis [11, 13]). This is despite the fact that worst-case exponential blow-up is unavoidable when translating LTL formulas to BAs.

We shall only discuss analyses (III.-V.) in more detail. We will also observe that some key facts used for analyzing finite-state MCs hold more generally, for all denumerable MCs. Suppose we are "given" a MC, $\mathcal{M} = (S, P)$, where for now we allow the set S to be countably infinite. Later, for computational purposes, we will assume S is finite. For convenience, we equate S with (an initial segment of) the positive natural numbers $\mathbb{N}_+ = \{1, 2, \ldots\}$. We let $n \equiv |S|$. Thus, if $n \in \mathbb{N}_+$, then $S = \{1, \ldots, n\}$, and otherwise if $n = \infty$ (i.e., if $n = \omega$), then $S = \mathbb{N}_+$.

HP: Suppose we are "given" a subset $F \subseteq S$ of target states, and suppose we wish to compute the probabilities, q_i^*, of eventually hitting a target state in F starting from initial

[11] Recall that a problem whose input instances are represented by a vector of rational values is said to be solvable in *strongly* polynomial time if the problem can be solved by an algorithm that both: (i) runs in polynomial time, as a function of the dimension n of the input vector, in the unit-cost (arbitrary precision) arithmetic RAM model of computation, where standard arithmetic operations $\{+, *\}$ on, and comparisons of, arbitrary rational numbers require unit-cost, and (ii) runs in polynomial space as a function of the encoding size of the input vector, where the rational coordinates are encoded as usual, with numerator and denominator given in binary.

state $i \in S$. In other words, $q_i^* \doteq \mathbb{P}_i(\exists t \geqslant 0 : X_t \in F)$.

We first observe that hitting probabilities for a denumerable MC can be "computed" by "solving" the following linear system of equations (albeit, with infinitely many equations, if there are infinitely many states). There is one variable, x_i, and one equation, for every state $i \in S$:

$$\begin{array}{ll} x_i = 1 & \text{for all } i \in F; \\ x_i = \sum_{j \in S} P_{i,j} \cdot x_j & \text{for all } i \in S \setminus F. \end{array} \tag{3.1}$$

The vector of variables is denoted $x = (x_i : i \in S)$. Note that if $n \equiv |S| = \infty$, then the infinite sums in (3.1) are always uniquely defined, because the coefficients are non-negative and we interpret the variables x_j only over non-negative reals (indeed, over probabilities). We can denote the entire system of equations, in vector notation, as:

$$x = R(x)$$

where $R(x)$ denotes the linear (affine) map given by the right hand sides of the linear equations in (3.1). Note that since all coefficients and constants in the linear maps defining $R(x)$ are non-negative, $R : \mathbb{R}^n_{\geqslant 0} \to \mathbb{R}^n_{\geqslant 0}$ defines a *monotone* mapping from non-negative vectors to non-negative vectors. That is, for all $x \geqslant y \geqslant \mathbf{0}$, we have $R(x) \geqslant R(y) \geqslant 0$. It is easy to see that the hitting probabilities $q^* = (q_i^* : i \in S)$ must be a solution of $x = R(x)$. Indeed, if $i \in F$, then clearly $q_i^* = 1$, and if $i \in S \setminus F$, then clearly $q_i^* = \sum_{j \in S} P_{i,j} q_j^*$, because starting at $i \notin F$, in order to eventually hit F, we first have to take one step and thereafter eventually hit F, and $\sum_{j \in S} P_{i,j} q_j^*$ captures the probability of eventually hitting F after one step, starting at i.

Unfortunately, in general the equations $x = R(x)$ can have multiple solutions, for trivial reasons. To see this, consider the trivial 2-state Markov chain with states $S = \{1, 2\}$, with transition probabilities defined by $P_{1,1} = P_{2,2} = 1$, and $P_{i,j} = 0$ for $i \neq j$, and where the target state is $F = \{1\}$. The equations $x = R(x)$ are thus given by ($x_1 = 1$; $x_2 = x_2$). Obviously, any pair $(1, r)$ for $r \in \mathbb{R}$ is a solution.

It turns out the hitting probabilities $q^* = (q_i^* : i \in S)$ are always the least non-negative solution of $x = R(x)$, which is the *least fixed point* (LFP) of the monotone operator $R : \mathbb{R}^n_{\geqslant 0} \to \mathbb{R}^n_{\geqslant 0}$. Let us state this more precisely. For a vector $y \in \mathbb{R}^n$, and $k \geqslant 1$, let $R^0(y) = y$, and for $k \geqslant 1$, let $R^{k+1}(y) = R(R^k(y))$. For any $k \geqslant 0$, let q_i^k denote the probability of hitting target set F starting in initial state i, in *at most k time steps*. In other words, $q_i^k \doteq \mathbb{P}_i(\exists t\, (0 \leqslant t \leqslant k) : X_t \in F)$. Note that $\lim_{k \to \infty} q_i^k \uparrow = q_i^*$, meaning q_i^k converges monotonically from below to q^*, as $k \to +\infty$. Let $q^k = (q_i^k : i \in S)$ denote the corresponding vector. We shall use $\mathbf{0}$, or just 0, to denote an all-zero vector of the appropriate dimensions, when this is clear from the context. The following key Proposition, 3.1, is well-known and easy to prove: part (1) can be proved by induction on k, and the rest follows. (We will later learn that variants of Proposition 3.1 hold in much more general settings, when the symbols in the proposition are interpreted differently.)

Proposition 3.1.

(1) For all $k \geqslant 0$, $q^k = R^{k+1}(\mathbf{0})$, and thus $R^{k+1}(\mathbf{0}) \leqslant q^*$, and $\lim_{k \to \infty} R^k(\mathbf{0}) \uparrow = q^*$.

(2) $q^* = R(q^*)$, and if $q' \in \mathbb{R}^n_{\geqslant 0}$ and $q' = R(q')$ then $q^* \leqslant q'$.

In other words, q^* is the Least Fixed Point *(LFP) of $R(x)$.*

Now suppose that $\mathcal{M} = (S, P)$ is a finite-state Markov chain, so $n \equiv |S| < \infty$, and that we are given the transition probability matrix P explicitly. How can we use Proposition (3.1) to compute the hitting probabilities q^*? We have to compute the least non-negative solution to the linear system of equations $x = R(x)$. One (not very efficient) way to do this in polynomial time is to formulate this as a linear programming problem. Namely, the vector q^* is the unique optimal solution to the following LP.

$$\begin{array}{ll} \textbf{minimize:} & \sum_{i \in S} x_i \\ \textbf{subject to:} & \\ R(x) \leqslant x\ ; & \\ x \geqslant 0. & \end{array} \tag{3.2}$$

Note that the inequality $R(x) \leqslant x$ stands for a system of inequalities $R_i(x) \leqslant x_i$, $i \in S$, and likewise $x \geqslant 0$ stands for $x_i \geqslant 0$, $i \in S$.

Although this already shows we can compute q^* in P-time, we can do much better. Namely, let us denote by $G = (S, \Delta)$ the underlying directed graph of the MC, $\mathcal{M}$. Note that $q_i^* = 0$ if and only if there is no path in G from i to any state $j \in F$. We can thus easily compute the set $S_{Zero} = \{i \mid q_i^* = 0\}$ in P-time by a simple depth-first search in G. We can then remove the equations corresponding to variables x_i, $i \in S_{Zero}$, from the system of equations $x = R(x)$, and replace occurrences of variables $x_i \in S_{Zero}$ by 0 on the right hand side of any other equations $x_j = R_j(x)$ where they occur. For convenience in what is to follow, we also remove the variables x_i for $i \in F$, and their equations $x_i = 1$, and replace the occurrences of variables $x_i \in F$ by 1 on the RHS of any other equations $x_j = R_j(x)$ where they occur.

This gives us a new system of linear equations $\hat{x} = \hat{R}(\hat{x})$, in fewer variables. It turns out that this new system has a *unique* solution, corresponding to the remaining (positive) coordinates of q^*, and furthermore, if the equation is written in matrix notation as $\hat{x} = \hat{P}\hat{x} + b$, then the matrix $(I - \hat{P})$ is guaranteed to be invertible, and the (positive) coordinates of q^* that were not eliminated are given by the solution $(I - \hat{P})^{-1}b$. Thus, we can compute q^* in (*strongly*) polynomial time by first doing some simple graph-theoretic analysis on G, and then solving a linear system of equations.

We note that it follows from basic facts in matrix theory that $(I - \hat{P})^{-1} = \sum_{k=0}^{\infty} \hat{P}^k$. We can use this to put a probabilistic interpretation on the calculation $q^* = (I - \hat{P})^{-1}b = \sum_{k=0}^{\infty} \hat{P}^k b$. Note that $\hat{P}^k_{i,j} = \mathbb{P}_i(X_k = j)$ is the probability that, in Markov chain $\hat{\mathcal{M}}$ derived from $\mathcal{M}$, which excludes all states in $S_{Zero} \cup F$, and replaces them with dead-end absorbing states, starting in state i, at time k the trajectory is in state j. Thus, for $k \geqslant 0$, $(\hat{P}^k b)_i$ is the probability of entering a state in F for the first time at time $k + 1$. It is thus clear, by a probabilistic argument, that $q^* = \sum_{k=0}^{\infty} \hat{P}^k b = (I - \hat{P})^{-1}b$.

A more basic method for computing q^* numerically is already immediately suggested by Proposition 3.1, and it "works" even for infinite-state MCs. Namely, we can simply iteratively compute a sequence of vectors $y^k = R^k(0)$, $k = 0, 1, \ldots$, letting $y^0 := 0$, and $y^{k+1} := R(y^k)$. By Proposition 3.1, the sequence $y^k = R^k(0)$ converges monotonically to q^*. This well-known method is called *value iteration*. Of course, one issue is that we do not *a priori* know how many iterations of value iteration are required as a function of the input matrix P in order to converge to within a desired error bound of the vector q^*. It turns out that in the worst case there are bad examples for finite-state MCs, where convergence of value iteration can be very slow. For example, consider the MC $\mathcal{M} =$

(S, P), where $S = \{1, \ldots, n\}$, and where the target set is $F = \{n\}$, and where for all $i \in \{1, \ldots, n-1\}$, $P_{i,1} = 1/2$ and $P_{i,i+1} = 1/2$, and $P_{n,n} = 1$, and all other transition probabilities are of course 0. Note that $q^*_i = 1$ for all $i \in S$. Now by Proposition 3.1, $q^k_i = L^k_i(0)$. However, it can be seen that for all $k \leqslant 2^n$, $q^k_1 \leqslant 1 - (1 - 1/2^n)^{2^n} \leqslant (1 - (1/e))$, where $e = 2.71828\ldots$ is the base of the natural log. Thus, we need at least $k \geqslant 2^n$ value iterations before $|L^k_1(0) - q^*_1| \leqslant 1/3 \leqslant (1/e)$. However, value iteration works reasonably well on many instances of MCs, and optimized variants of it are widely used in practice (also for MDPs).

NTP: Let us now consider non-negative total payoff analysis of MCs which, as already noted, generalizes hitting probability analysis. We shall now reuse symbols q^* and $R(x)$, with a different interpretation, for reasons that will become clear shortly. Suppose we have a non-negative payoff-labeled MC, $\mathcal{M} = (S, P, l)$, with $n \equiv |S|$ states (possibly infinite), and with $l : S \to \mathbb{N}$. We wish to compute $q^*_j = \mathbb{E}_j(\lim_{k\to\infty} \sum_{i=0}^{k} l(X_i))$. We can again write a linear system of equations for this, with one equation per variable x_i, over variables $x = (x_i \mid i \in S)$, as follows.

$$x_i = l(i) + \textstyle\sum_{j \in S} P_{i,j} \cdot x_j\,, \quad \text{for all } i \in S. \tag{3.3}$$

We can again denote this system of linear equations, in vector notation, as $x = R(x)$. Since $l(i) \geqslant 0$, the operator $R(x)$ is again monotone, and it turns out that again the vector q^* of expected total payoffs is the least non-negative solution of $x = R(x)$, except with the difference that we now must also allow for the possibility that some coordinates of q^* are $+\infty$. Formally, we can work over the ordered semi-ring $\overline{\mathbb{R}}_{\geqslant 0} = \mathbb{R}_{\geqslant 0} \cup \{+\infty\}$, where by definition $+\infty * 0 = 0$, and $+\infty + r = +\infty$, and $+\infty \geqslant r$, for all $r \in \hat{\mathbb{R}}_{\geqslant 0}$. Let $q^k_j = \mathbb{E}_j(\sum_{t=0}^{k} l(X_t))$. Then, it turns out that

Proposition 3.2. *The statement of Proposition 3.1 holds true, verbatim, for the above re-interpretations of $x = R(x)$, q^*, and q^k.*

Thus the expectation vector q^* is the *least fixed point* of the monotone operator $R : \overline{\mathbb{R}}^n_{\geqslant 0} \to \overline{\mathbb{R}}^n_{\geqslant 0}$. Thus, by Proposition 3.2, value iteration $y^k := R^k(\mathbf{0})$ converges monotonically, in the limit, to the expected total payoff vector q^*. However, since some coordinates of q^* may now be $+\infty$, the value iterates y^k may never actually get "close enough" to q^*. We can nevertheless again compute expectations q^* in strongly polynomial time for finite-state MCs, including determining those coordinates that are $+\infty$, using a variant of what was described earlier for computing hitting probabilities. First, consider the underlying graph $G = (S, \Delta)$ of $\mathcal{M}$. For any bottom-SCC, $C \subseteq S$, of G, if there is some $j \in C$ such that $l(j) > 0$, then clearly $v^*_{j'} = +\infty$ for all $j' \in C$, and for all $j' \in S$ such that $j' \overset{*}{\rightsquigarrow} j$. Indeed, this describes all states such that $v*_{j'} = +\infty$, because with probability 1 the trajectory will eventually hit some BSCC, and thereafter stay in that BSCC forever. We can thus use depth-first search to decompose G into its DAG of SCCs, and find and remove from the equations $x = R(x)$ any variable x_i such that $q^*_i = +\infty$. Likewise, by simple reachability analysis on G we can find and remove all variables x_i such that $q^*_i = 0$, by just noting that $q^*_i = 0$ iff there is no state $j \in S$ such that both $l(j) > 0$ and $i \overset{*}{\rightsquigarrow} j$. After we remove, as indicated, both $+\infty$ and 0 variables from the equations, we are left either with an empty list of equations or a system of linear equations on the

remaining variables whose *unique* solution is a positive real-valued vector that yields the remaining coordinates of the vector q^*. We can thus compute these remaining coordinates (in strongly polynomial time) by solving the remaining linear equations.

MoCh: Suppose we are given a labeled finite-state MC, $\mathcal{M} = (S, P, l)$, an initial distribution $\mathcal{I}$, and a Büchi automaton, $\mathcal{B} = (Q, \Sigma, q_0, \delta, F)$. Suppose we wish to compute the probability $p^* = \mathbb{P}_{\mathcal{I}}(L(\mathcal{B}))$. We now describe an algorithm for computing p^*, due to Courcoubetis and Yannakakis [13], which runs in time polynomial in the encoding size $|\mathcal{M}|$ of $\mathcal{M}$, and exponential in the encoding size $|\mathcal{B}|$ of $\mathcal{B}$.

We can assume, without loss of generality that $\Sigma = S$, i.e., the alphabet of $\mathcal{B}$ is the set of states of $\mathcal{M}$. We can do so because we can always update the transition relation δ of $\mathcal{B}$, refining it so that if (q, a, q') was in δ, and for some $s \in S$ we have $l(s) = a$, then we put (q, s, q') in the new transition relation. It is clear that the probability that $\mathcal{M}$ generates a trajectory accepted by the new BA is the same as the probability that $\mathcal{M}$ generates a trajectory labeled by an ω-word in $L(\mathcal{B})$. So from now on, we assume $\Sigma = S$.

We first perform a naive *subset construction* on the BA, $\mathcal{B}$, to obtain a deterministic BA. Recall however that the subset construction doesn't in general preserve the ω-regular language of a BA, and that in fact ω-regular languages accepted by some nondeterministic BAs are not accepted by any deterministic BA. Nevertheless, it was shown by [11, 13] that the subset construction "works" in a suitable way for probabilistic model checking. Let $\mathcal{B}' = (2^Q, \Sigma, \{q_0\}, \delta', F')$, be the deterministic BA obtained by performing the usual subset construction on $\mathcal{B}$. The states of $\mathcal{B}'$ are 2^Q, the alphabet is $\Sigma = S$, the start state is $\{q_0\}$, and $\delta' \subseteq 2^Q \times \Sigma \times 2^Q$ is a *deterministic* transition relation defined by $\delta' := \{(T, a, T') \mid T' = \{q' \in Q \mid \exists q \in T : (q, a, q') \in \delta\}$. Finally, we let $F' = \{T \subseteq Q \mid T \cap F \neq \emptyset\}$.

Next, we define the *product* MC, $\mathcal{M} \otimes \mathcal{B}' = (S \times 2^Q, \tilde{P})$, obtained from the MC $\mathcal{M}$, and the deterministic Büchi automaton, $\mathcal{B}'$. The states of $\mathcal{M} \otimes \mathcal{B}'$ are pairs (s, T), where $s \in S$ and $T \in 2^Q$, The transition probability function $\tilde{P}$ is defined as follows:

$$\tilde{P}((s,T),(s',T')) = \begin{cases} P(s,s') & \text{if } (T,s',T') \in \delta' \\ 0 & \text{otherwise} \end{cases}$$

Note that $\mathcal{M} \otimes \mathcal{B}'$ is indeed an MC, whose trajectories are a *refinement* of the trajectories of $\mathcal{M}$. In particular, projecting a trajectory $\xi \in (S \times 2^Q)^\omega$ on to its left coordinates yields a trajectory of $\mathcal{M}$. Let $G_{\mathcal{M}\otimes\mathcal{B}'}$ denote the underlying directed graph of the MC, $\mathcal{M} \otimes \mathcal{B}'$. Finally, for a pair $(s, T) \in S \times 2^Q$, which defines a state of $\mathcal{M} \otimes \mathcal{B}'$, and thus also a node of $G_{\mathcal{M}\otimes\mathcal{B}'}$, let $G_{\mathcal{M}\otimes\mathcal{B}'}((s, T))$ denote the directed subgraph of $G_{\mathcal{M}\otimes\mathcal{B}'}$ induced by the set of nodes consisting of all of the nodes $(s', T') \in S \times 2^Q$ of $G_{\mathcal{M}\otimes\mathcal{B}'}$ that are reachable from (s, T), i.e., such that $(s, T) \overset{*}{\rightsquigarrow} (s', T')$.

The following important definitions are key to the algorithm. A pair $(s, q) \in S \times Q$ is called *special*[12] if $q \in F$ and some bottom-SCC C of $G_{\mathcal{M}\otimes\mathcal{B}'}((s, \{q\}))$ contains a node $(s, T) \in C$ with $q \in T$. For a bottom-SCC, $C \subseteq S \times 2^Q$ of $G_{\mathcal{M}\otimes cB'}$ (and thus also of $\mathcal{M} \otimes B'$) we shall call C *accepting* if there exists some $(s, T) \in C$ such that there exists $q \in T \cap F$ such that (s, q) is a special pair. The following theorem from [11, 13] reduces the MoCh problem for finite-state MCs to HP problems on (larger) finite-state MCs:

[12] In [13] "recurrent" was used, but "recurrent" has other meanings so we use "special" instead, as in [29].

Theorem 3.3 ([11, 13]). *Given a labeled finite-state MC, $\mathcal{M} = (S, P, l)$, with initial state $s \in S$, and given a non-deterministic BA, $\mathcal{B}$, with initial state q_0, the probability $\mathbb{P}_s(L(\mathcal{B}))$ is equal to the probability that in the MC $\mathcal{M} \otimes B'$, starting from initial state $(s, \{q_0\})$, the trajectory eventually reaches an accepting bottom-SCC of $\mathcal{M} \otimes B'$.*

Thus, in order to compute $\mathbb{P}_s(L(\mathcal{B}))$, we first need to do graph-theoretic analysis on the directed graphs $G_{\mathcal{M}\otimes B'}$, and also analysis of various subgraphs $G_{\mathcal{M}\otimes B'}((s', \{q\}))$, for $s' \in S$ and $q \in F$, so as to compute *special* pairs $(s', q) \in S \times Q$, and use that to compute all *accepting* bottom-SCCs of $G_{\mathcal{M}\otimes B'}$. We can then consider all nodes in such accepting bottom-SCCs as target nodes, and compute the probability of hitting a target node starting from the initial state $(s, \{q_0\})$ of the MC $\mathcal{M} \otimes B'$, which yields the probability $\mathbb{P}_s(L(\mathcal{B}))$ that we are after. To compute the hitting probabilities we of course use the methods already described for solving HP. Note that this algorithm does not involve full-fledged determinization of Büchi automata (such as Safra's construction) which involves a $2^{|\mathcal{B}| \log |\mathcal{B}|}$ blow-up in size and requires more sophisticated acceptance conditions such as Rabin or Müller conditions.

Overall, this algorithm runs in *strongly* polynomial time as a function of $|\mathcal{M}|$ (assuming $\mathcal{B}$ is fixed) and exponential time as a function of $|\mathcal{B}|$, when $\mathcal{B}$ is nondeterministic (and polynomial in $|\mathcal{B}|$ when $\mathcal{B}$ is deterministic). It was furthermore shown in [11, 13] that, given MC, $\mathcal{M}$, and nondeterministic BA, $\mathcal{B}$, as input, the qualitative problem of deciding whether $\mathbb{P}_s(L(\mathcal{B})) = 1$ is in PSPACE, and it was already shown in [41] that the problem is PSPACE-hard, so the qualitative problem is PSPACE-complete.

Courcoubetis-Yannakakis [11, 13] also considered model checking of finite-state MCs with respect to properties specified by LTL formulas and, remarkably, they showed that both the quantitative problem and the qualitative problem for LTL model checking of MCs has the same complexity as that of model checking an ω-regular property given by a nondeterministic BA. This was surprising, because it is well-known that in general translating an LTL formula to a BA requires worst-case exponential blow-up. Their algorithm involves iterative constructions of larger and larger finite-state MCs, starting from $\mathcal{M}$, built up via a structural induction on the subformulas of the LTL formula. The transition probabilities of the new MCs in the iterative construction are obtained by computing certain hitting probabilities on the old MCs. See [13] for details.

4 Analysis of finite-state MDPs

We now review some algorithms for analyzing finite-state MDPs, and discuss their complexity. Many analogies with the algorithms for finite-state MCs will soon become clear. In fact, we have deliberately stated some equations and facts for finite-state MCs in a general enough form so as to be able to reuse them here (and also later, for recursive MCs and 1-recursive MDPs).

Let us already summarize the known facts: again, for all of the analyses (I.-V.), listed in section 2.1, all qualitative and quantitative decision and computation problems are solvable in polynomial time as a function of the encoding size of the given MDP (but the known P-time algorithms for all of them require solving linear programming problems,

and thus none of them are currently known to be solvable in *strongly* polynomial time[13]). For qualitative analyses, the algorithms only involve and-or game graph analysis on the underlying transition graph G of the MDP, $\mathcal{D}$, which can be done in P-time. For quantitative analyses, the algorithms additionally involve solving corresponding max/min-linear ***Bellman optimality equations***, which can be solved in P-time using linear programming. For model checking (MoCh) the complexity is polynomial in the encoding size of $\mathcal{D}$, but again exponential in the encoding size $|\mathcal{B}|$ if the ω-regular property, $\mathcal{L} = L(\mathcal{B})$, is given by a nondeterministic Büchi automaton, $\mathcal{B}$. However for finite-state MDPs, unlike finite-state MCs, if $\mathcal{L} = L(\varphi)$ is given by an LTL formula, φ, then the complexity is double-exponential as a function of the encoding size of φ. These complexity bounds can not be improved, because the problems are EXPTIME-hard and 2EXPTIME-hard, respectively. These results on model checking finite-state MDPs were established by Courcoubetis and Yannakakis in [11, 13, 12, 14].

Analyses, I. MP, and II. DTP, are standard for finite-state MDPs, and algorithms for them can be found in any textbook on MDPs. See, e.g., [38] for a thorough treatment.

Let us mention that for analyses (I.-IV.) on finite-state MDPs, it is well-known that there always exist deterministic memoryless optimal strategies (see [38]). For model checking (V. MoCh), memoryless strategies do not suffice in general for optimizing the probability of an ω-regular property, but bounded-memory strategies do suffice ([14]).

We shall only discuss analyses HP and MoCh further. Suppose we are "given" a MDP, $\mathcal{D} = (S, (S_0, S_1), \Delta, P)$, where for now we allow the set S to be countably infinite. Again, for convenience, we equate S with (an initial segment of) the positive natural numbers $\mathbb{N}_+ = \{1, 2, \ldots\}$, and let $n \equiv |S|$. We will furthermore assume that every state $i \in S_1$ is *boundedly branching*, meaning there is some $k \in \mathbb{N}$ (depending on the MDP), such that for every $i \in S_1$, $|\texttt{successors}(i)| \leqslant k$. This allows us to use $\max$ and $\min$ operators in the Bellman optimality equations, whereas we would otherwise require $\sup$ and $\inf$.

HP: Suppose we are "given" a subset $F \subseteq S$ of target states, and suppose we wish to compute the supremum probabilities, $q^*_{\max,i}$, or the infimum probabilities, $q^*_{\min,i}$, of eventually hitting a target state in F starting from initial state $i \in S$. In other words, $q^*_{\max,i} \doteq \sup_\sigma \mathbb{P}^\sigma_i(\exists t \geqslant 0 : X_t \in F)$, and $q^*_{\min,i} \doteq \inf_\sigma \mathbb{P}^\sigma_i(\exists t \geqslant 0 : X_t \in F)$,

Maximum (minimum, respectively) hitting probabilities for a denumerable MDP can be "computed" by "solving" the following max-(min-)linear system of equations, called their *Bellman optimality equations*. There is one variable, x_i, and one equation, for every state $i \in S$. Let $\mathsf{opt} = \max$ or $\min$, according to whether we are maximizing or minimizing hitting probability. The equations are given by:

$$\begin{array}{ll} x_i = 1 & \text{for all } i \in F; \\ x_i = \sum_{j \in S} P_{i,j} \cdot x_j & \text{for all } i \in S_0 \setminus F; \\ x_i = \mathsf{opt}_{j \in \texttt{successors}(i)} x_j & \text{for all } i \in S_1 \setminus F. \end{array} \tag{4.1}$$

Note that, as in the case of MCs, if $n \equiv |S| = \infty$, then the infinite sums for variables $i \in S_0$ in (3.1) are always well defined because of non-negativity. Furthermore, since we

[13] A notable exception is the case of DTP where the discount factor is a *fixed constant*, which was shown in [45] to be solvable in strongly polynomial time. See also [32] for a generalization to turn-based discounted stochastic games

have assumed $|\texttt{successors}(i)| \leqslant k < \infty$ for all $i \in S_1$, the $\max$ (or $\min$) operators in the equations for variables $i \in S_1$ are well defined for any real values assigned to the variables x_j. We can denote the entire system of equations, in vector notation, as:

$$x = R(x)$$

where $R(x)$ denotes the $\max$-($\min$)-linear map given by the right hand sides of the equations in (4.1). Note that since all coefficients and constants defining $R(x)$ are non-negative, $R : \mathbb{R}^n_{\geqslant 0} \to \mathbb{R}^n_{\geqslant 0}$ again defines a *monotone* map from non-negative vectors to non-negative vectors. Let $q^* = (q^*_{\mathsf{opt},i} : i \in S)$, where $\mathsf{opt} = \max$ or $= \min$, respectively. For any $k \geqslant 0$, let $q^k_{\mathsf{opt},i}$ denote the optimal probability of hitting target set F starting in initial state i, in *at most k time steps*. Let $q^k = (q^k_{\mathsf{opt},i} : i \in S)$ denote the corresponding vector of optimal probabilities. The following is again easy to prove by induction on k.

Proposition 4.1. *The statement of Proposition 3.1 holds true, verbatim, for the above re-interpretations of $x = R(x)$, q^*, and q^k.*

Thus the optimal hitting probabilities q^* are the LFP of $x = R(x)$. Now suppose that $\mathcal{D} = (S, (S_0, S_1), \Delta, P)$ is a finite-state MDP. How can we use Proposition (4.1) to compute the optimal hitting probabilities q^*? We have to compute the least non-negative solution to the linear system of equations $x = R(x)$. One way to do this in polynomial time for *maximizing* MDPs is to formulate this as a linear programming problem. Namely, the vector $q^*_{\max}$ is the unique optimal solution to the LP given in (3.2), with this new interpretation of $R(x)$. However, to express the constraints $R(x) \leqslant x$ as an LP, and recalling that for $i \in S_1$, $R_i(x) \equiv \max_{j \in \texttt{successors}(i)} x_j$, we need to rewrite the corresponding constraints, $R_i(x) \leqslant x_i$, as a system of linear inequality constraints $(x_i \leqslant x_i \mid j \in \texttt{successors}(i))$. With this modification (3.2) again defines an LP, and the vector $q^*_{\max}$ is the unique optimal solution to this LP.

For *minimizing* MDPs, computing $q^*_{\min}$ can also be reduced to linear programming, but this case involves some more preprocessing. In order to express the problem as an LP one first needs to do a little graph-theoretic analysis. Specifically, we first need to identify and remove all states i such that $q^*_{\min,i} = 0$. We can do this by a simple and-or game graph analysis on the underlying graph G of the MDP. Once this is done, it turns out that on the remaining MDP one can solve for $q^*_{\min}$ as the unique optimal solution of a different LP, namely the LP given by *maximize:* $\sum_i x_i$; *subject to:* $R(x) \geqslant x$, $x \geqslant 0$, where in this case when we have $R_i(x) = \min_{j \in \texttt{successors}(i)} x_j$, we have to rewrite the constraint $R_i(x) \geqslant x_i$, as a system of constraints $(x_j \geqslant x_i \mid j \in \texttt{successors}(i))$.

A more basic method for computing q^* is again already immediately suggested by Proposition 3.1: *value iteration*. By Proposition 3.1, the sequence $y^k = R^k(0)$ converges monotonically to q^*. As we already saw, even for finite-state MCs, value iteration can be slow to converge in the worst case, but it is widely used in practice, also for MDPs.

Another standard method for solving HP for maximizing MDPs, as well as for solving many other classes of MDPs, is called *policy iteration* or *strategy improvement*. It involved initially fixing an arbitrary (memoryless) strategy for the controller, and evaluating q^* on the resulting MC, and then updating the strategy (at every state) by choosing a neighbor whose value is strictly greater than that of the currently chosen neighbor chosen by the previous fixed strategy, if such a strictly greater neighbor exists. See, e.g., [38] for

much more on policy iteration for MDPs.

It is worth mentioning that answering the *qualitative* questions of whether $q^*_{\max,i} = 0, 1$, or whether $q^*_{\min,i} = 0, 1$, requires only (game) graph theoretic analyses that do not depend on the actual probabilities of transitions in the given MDP, and so do not require solving LPs. Thus, these qualitative questions for HP can be answered in *strongly* polynomial time. (See, e.g., [13, 14].)

MoCh: Given a labeled finite-state MDP, $\mathcal{D} = (S, (S_0, S_1), \Delta, P, l)$, an initial state $s_0 \in S$, and a Büchi automaton, $\mathcal{B} = (Q, \Sigma, q_0, \delta, F)$, we wish to compute the optimum (w.l.o.g., maximum) probability $p^* = \sup_\sigma \mathbb{P}^\sigma_{s_0}(L(\mathcal{B}))$. Qualitative decision problems associated with this were studied in [13, 41], and quantitative decision problems where studied in [14]. We briefly mention the main results of [14].

As in the case of MoCh for MCs, we can assume, w.l.o.g., that $\Sigma = S$, and we let $\mathcal{B}' = (2^Q, \Sigma, \{q_0\}, \delta', F')$, be the deterministic BA obtained by performing the usual (naive) subset construction on $\mathcal{B}$. Next, as for MCs, we define the *product* MDP, $\mathcal{D} \otimes \mathcal{B}' = (S \times 2^Q, (S_0 \times 2^Q, S_1 \times 2^Q), \tilde{\Delta}, \tilde{P})$. Note that there is a one-to-one correspondence between strategies σ on $\mathcal{D}$ and strategies σ on $\mathcal{D} \otimes \mathcal{B}'$ (because $\mathcal{B}'$ is deterministic). Using more involved analysis than for the case of MCs, employing the notion of *controllably recurrent* pairs $(s, q) \in S \times Q$ (which we will not define here) that roughly correspond to the *special* pairs in the case of MCs, [14] showed how one can compute a set of *target* states $Z \subseteq S \times 2^Q$ of $\mathcal{D} \otimes \mathcal{B}'$, such that in order to optimize the probability $\mathbb{P}^\sigma_{\mathcal{I}}(L(\mathcal{B}))$ in $\mathcal{D}$, it suffices for the strategy σ to first optimize the probability of hitting a target set Z in $\mathcal{D} \otimes \mathcal{B}'$ and once a target state in $z \in Z$ is hit, the strategy σ should then switch to a different strategy σ_z that thereafter assures that with probability 1 the infinite trajectory is accepted by $\mathcal{B}$ (which is made possible, by definition of the target states Z). In this way, the problem MoCh is reduced to (much larger) instances of the problem HP, which as we saw can be solved using linear programming. Let us note however that, whereas for HP we always have memoryless deterministic (positional) optimal strategies, the optimal strategies obtained this way for MoCh by [14] are not positional, and in fact it is easy to see that optimal positional strategies for MoCh need not exist. The complexity of [14]'s algorithm for computing $p^* = \max_\sigma \mathbb{P}^\sigma_{s_0}(L(\mathcal{B}))$ is polynomial in $|\mathcal{D}|$ and exponential in the size $|\mathcal{B}|$ for a nondeterministic Büchi automaton $\mathcal{B}$. It was previously shown in [13] that even the qualitative decision problem of determining whether $p^* = 1$ is EXPTIME-complete, and thus we can not improve substantially on this complexity upper bound. If the ω-regular property is specified as an LTL formula instead, it was shown in [13] that the resulting qualitative problem of determining whether $p^* = 1$ is already 2EXPTIME-complete.

5 Adding Recursion to MCs and MDPs

As mentioned in the introduction, a number of important classes of countably infinite-state MCs and MDPs that are closely related to automata-theoretic models are subsumed, in precise senses, by adding a natural *recursion* feature to MCs and MDPs. in a manner similar to allowing potentially recursive subroutine calls in procedural programs. The resulting formal models, called *recursive Markov chains* (RMCs) and *recursive Markov*

Decision Processes (RMDPs) were defined and studied in [28, 29] and in [30, 27], respectively. RMCs and RMDPs provide natural abstract models for probabilistic procedural programs with recursion (and this indeed partly motivated their study). RMCs (and RMDPs), and various of their subclasses, capture probabilistic and controlled extensions of classic infinite-state automata theoretic models, including pushdown automata, context-free grammars, and one-counter automata. Indeed, RMCs and RMDPs can equivalently be viewed as probabilistic and MDP extensions of pushdown automata. We refer the reader to [28] and [30] for detailed formal definitions and results about RMCs and RMDPs, respectively.

A (not-necessarily finitely-presented) *Recursive Markov Chain (RMC)*, A, is a tuple $A = (A_1, \ldots, A_k)$, where each *component* $A_i = (N_i, B_i, Y_i, En_i, Ex_i, \delta_i)$ consists of:

- A (countable, or finite) set N_i of *nodes*.
- A subset of *entry* nodes $En_i \subseteq N_i$, and a subset of *exit* nodes $Ex_i \subseteq N_i$.
- A (countable, or finite) set B_i of *boxes*, and a mapping $Y_i : B_i \mapsto \{1, \ldots, k\}$ that assigns to every box (the index of) one of the components, $A_1, \ldots, A_k$. To each box $b \in B_i$, we associate a set of *call ports*, $Call_b = \{(b, en) \mid en \in En_{Y_i(b)}\}$ corresponding to the entries of the corresponding component, and a set of *return ports*, $Return_b = \{(b, ex) \mid ex \in Ex_{Y_i(b)}\}$, corresponding to the exits of the corresponding component.
- A probabilistic transition relation δ_i, where transitions are of the form $(u, p_{u,v}, v)$ where:
  (1) the source u is either a non-exit node $u \in N_i \backslash Ex_i$, or a return port $u = (b, ex)$ of a box $b \in B_i$,
  (2) The destination v is either a non-entry node $v \in N_i \setminus En_i$, or a call port $u = (b, en)$ of a box $b \in B_i$,
  (3) $p_{u,v} \in \mathbb{R}_{>0}$ is the transition probability from u to v,
  (4) *Consistency of probabilities*: for each u, $\sum_{\{v' | (u, p_{u,v'}, v') \in \delta_i\}} p_{u,v'} = 1$, unless u is a call port or exit node, neither of which have outgoing transitions, in which case by default $\sum_{v'} p_{u,v'} = 0$.

When we want to ensure that an RMC is finitely-presented for computational purposes, we assume that all the sets involved (like nodes N_i and boxes B_i) are finite, and we assume that the transition probabilities $p_{u,v}$ are rational numbers, given as the ratio of two integers, and we measure their size by the number of bits in the numerator and denominator. The size, $|A|$, of a given finitely-presented RMC, A, is the number of bits needed to specify it (including the encoding size of the transition probabilities). As in the case of MCs and MDPs, some general theorems used for analysis of RMCs hold true even when sets defining them like nodes N_i and boxes B_i are (countably) infinite.

We will use the term *vertex* of A_i to refer collectively to its set of nodes, call ports, and return ports, and we denote this set by Q_i. Thus, the transition relation δ_i is a set of probability-weighted directed edges on the set Q_i of vertices of A_i. We will use all the notations without a subscript to refer to the union over all the components of the RMC A. Thus, $N = \cup_{i=1}^{k} N_i$ denotes the set of all the nodes of A, $Q = \cup_{i=1}^{k} Q_i$ the set of all vertices, $B = \cup_{i=1}^{k} B_i$ the set of all the boxes, $Y = \cup_{i=1}^{k} Y_i$ the map $Y : B \mapsto \{1, \ldots, k\}$ of all boxes to components, and $\delta = \cup_i \delta_i$ the set of all transitions of A.

Example 5.1. Figure 3 depicts a example RMC (taken from [29]). This RMC has two components A_1, A_2, each with one entry and two exits (in general different components

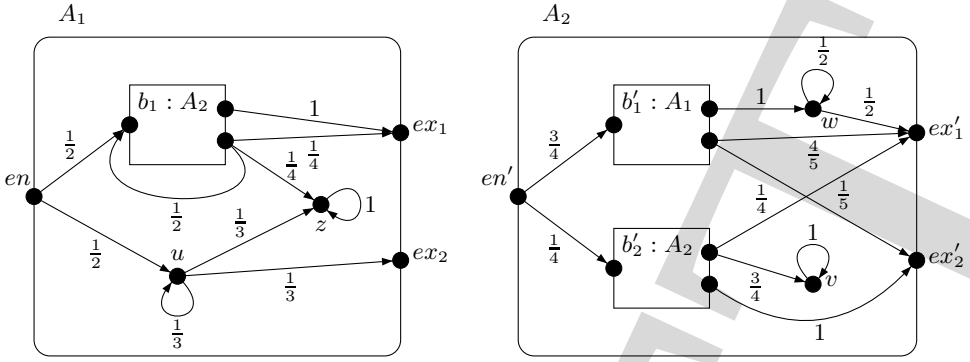

Figure 3. *A sample Recursive Markov Chain (taken from [29])*

may have different numbers of entries and exits). Component A_2 has two boxes, b'_1 which maps to A_1 and b'_2 which maps to A_2. □

An RMC A defines a global denumerable Markov chain $\mathcal{M}_A = (V, P_A)$ as follows. The global *states* $V \subseteq B^* \times Q$ are pairs of the form $\langle \beta, u \rangle$, where $\beta \in B^*$ is a (possibly empty) sequence of boxes and $u \in Q$ is a *vertex* of A, denoting the *call stack*. More precisely, the states $V \subseteq B^* \times Q$ and transition probabilities, P_A, of $\mathcal{M}_A$ are defined inductively as follows:

(1) $\langle \epsilon, u \rangle \in V$, for $u \in Q$. (ϵ denotes the empty string.)
(2) if $\langle \beta, u \rangle \in V$ and $(u, p_{u,v}, v) \in \delta$, then $\langle \beta, v \rangle \in V$ and $P_A(\langle \beta, u \rangle, \langle \beta, v \rangle) = p_{u,v}$.
(3) if $\langle \beta, (b, en) \rangle \in V$, where $(b, en) \in \mathit{Call}_b$, then $\langle \beta b, en \rangle \in V$ and $P_A(\langle \beta, (b, en) \rangle, \langle \beta b, en \rangle) = 1$.
(4) if $\langle \beta b, ex \rangle \in V$, where $(b, ex) \in \mathit{Return}_b$, then $\langle \beta, (b, ex) \rangle \in V$ and $P_A(\langle \beta b, ex \rangle, \langle \beta, (b, ex) \rangle) = 1$.

(1) corresponds to the possible initial states, (2) corresponds to a transition within a component, (3) corresponds to a recursive call when a new component is entered via a box, (4) corresponds to the end of a recursive call when the process exits a component and control returns to the calling component.

Some states of $\mathcal{M}_A$ are *terminating*, having no outgoing transitions. These are precisely the states $\langle \epsilon, ex \rangle$, where ex is an exit. If we want to view $\mathcal{M}_A$ as a proper Markov chain, we can consider terminating states of $\mathcal{M}_A$ to be *absorbing* states, with a self-loop transition to themselves having probability 1.

Unrestricted RMCs are essentially equivalent, in a precise sense, to *probabilistic pushdown automata* (pPDAs) (see [28] for the precise equivalence). An RMC where every component has at most one exit is called a *1-exit* RMC, or just *1-RMC*. 1-RMCs correspond, in a precise sense, to the stochastic process generated by left-most derivations of a *stochastic context-free grammar* (SCFG), and they also intimately related to *multi-type branching processes* (see [28] for details of these relationships). An RMC where there is only one box in the entire RMC is call a *1-box* RMC. As shown in [23], these correspond to probabilistic 1-counter automata, and to (discrete-time) quasi-birth death processes.

Termination probability analysis (VI. TP): We now define a key analysis for RMCs, namely computation of *termination probabilities*, which plays a central role in many other

analyses of RMCs. For an RMC, $A = (A_1, \ldots, A_k)$, given a vertex $u \in Q_i$ and an exit $ex \in Ex_i$, both in the same component A_i, let $q^*_{(u,ex)}$ denote the probability of eventually reaching the state $\langle \epsilon, ex \rangle$, starting at the state $\langle \epsilon, u \rangle$. Computation of termination probabilities $q^*_{(u,ex)}$ plays an important role for many other analyses of RMCs, including for MoCh, in a way analogous to the role that HP plays for analysing (finite-state) MCs.

Considering the termination probabilities $q^*_{(u,ex)}$ as unknowns, we can set up a system of *non-linear* polynomial equations, such that the probabilities $q^*_{(u,ex)}$ are the *Least Fixed Point* (LFP) solution of this system. Use a variable $x_{(u,ex)}$ for each unknown probability $q^*_{(u,ex)}$. We will often find it convenient to index the variables $x_{(u,ex)}$ according to a fixed order, so we can refer to them also as $x_1, \ldots, x_n$, with each $x_{(u,ex)}$ identified with x_j for some j. Of course, if N_i or B_i are infinite for some component A_i, then we have an infinite vector $\mathbf{x} = (x_1 \ldots x_j \ldots)$ of variables, rather than an n-vector $\mathbf{x} = (x_j \mid j \in \{1, \ldots, n\})$, for some $n < \infty$.

Given RMC $A = (A_1, \ldots, A_k)$, we define a system of polynomial equations, $x = R(x)$, over the variables $x_{(u,ex)}$, where $u \in Q_i$ and $ex \in Ex_i$, for $1 \leqslant i \leqslant k$. The system contains one equation $x_{(u,ex)} = R_{(u,ex)}(\mathbf{x})$, for each variable $x_{(u,ex)}$, where $R_{(u,ex)}(\mathbf{x})$ is a multivariate polynomial with positive rational coefficients. $x = R(x)$ is defined as follows: There are several based on the "type" of vertex u. Let $[k] = \{1, \ldots, k\}$.

$$
\begin{array}{ll}
x_{(u,ex)} = 1 & \text{if } u = ex \in Ex_i \text{ for } i \in [k] \\
x_{(u,ex)} = 0 & \text{if } u, ex \in Ex_i, \text{ and } u \neq ex, \text{ for } i \in [k] \\
x_{(u,ex)} = \sum_{\{v \mid (u, p_{u,v}, v) \in \delta\}} p_{u,v} \cdot x_{(v,ex)} & \text{if } u \in N_i \setminus \{ex\} \text{ or } u = (b, ex') \text{ for } b \in B_i,\ i \in [k] \\
x_{(u,ex)} = \sum_{ex' \in Ex_{Y(b)}} x_{(en,ex')} \cdot x_{((b,ex'),ex)} & \text{if } u = (b, en), \text{ for } b \in B_i,\ i \in [k]
\end{array}
\tag{5.1}
$$

Given a (finitely-presented) RMC A, we can obviously construct the system $\mathbf{x} = R(\mathbf{x})$ in polynomial time. $R(\mathbf{x})$ has size $O(|A|\theta^2)$, where θ denotes the maximum number of exits of any component. Let $\mathbf{q}^* \in \mathbb{R}^n$ denote the n-vector of probabilities $q^*_{(u,ex)}$, using the same indexing as used for $\mathbf{x}$. The map $R : \mathbb{R}^n_{\geqslant 0} \mapsto \mathbb{R}^n_{\geqslant 0}$ is clearly monotone on $\mathbb{R}^n_{\geqslant 0}$, and furthermore, the following analog of Proposition 3.1 holds.

Theorem 5.1. *(see [28] [14]) The termination probability vector $\mathbf{q}^*$ for an RMC is the* least fixed point *of $\mathbf{x} = R(\mathbf{x})$. Thus, $\mathbf{q}^* = R(\mathbf{q}^*)$, and for all $\mathbf{q}' \in \mathbb{R}^n_{\geqslant 0}$, if $\mathbf{q}' = R(\mathbf{q}')$, then $\mathbf{q}^* \leqslant \mathbf{q}'$. Furthermore, $R^k(0) \leqslant R^{k+1}(0) \leqslant \mathbf{q}^*\ \forall k \geqslant 0$, and $\mathbf{q}^* = \lim_{k \to \infty} R^k(0)$.*

For (finitely-presented) RMCs the termination probabilities q^* are in general irrational, so we can't compute them "exactly". However, using decision procedures for the *existential theory of reals*, we can decide, e.g., whether $q^*_j \geqslant r$, for any given rational value r, in PSPACE (see [28]). It was shown in [28] that for general RMCs *any non-trivial approximation* of the probabilities q^* is at least as hard as long standing open problems in the complexity of numerical computation, namely, the *square-root sum problem* and a harder arithmetic circuit decision problem known as PosSLP ([1]), both of which are not even known to be decidable in NP nor in the polynomial time hierarchy.

In [28], a decomposed multivariate *Newton's method* is studied and shown to converge

[14] In [28] this theorem is only claimed for finitely-presented RMCs, where the sets of nodes and boxes are finite, but exactly the same proofs establish the result when the sets of nodes and boxes can be countably infinite.

monotonically to the LFP, q^*, of $x = R(x)$ for an arbitrary RMC, starting from $\mathbf{0}$, and more generally this holds for any *monotone polynomial system of equations* (MPS), $x = R(x)$. The convergence behavior of Newton's method for MPSs was subsequently studied further in [16], yielding some important insights. Firstly, [16] gave examples of (not strongly connected) 1-exit RMCs, on whose system of equations $x = R(x)$ Newton's method would require an exponential number of iterations as a function of the encoding size of the 1-RMC (and of $x = R(x)$) to converge to within even 1 bit of precision of the LFP vector q^* starting from $\mathbf{0}$, and on the other hand, in certain strongly-connected cases of RMCs [16] gave exponential upper bounds on the number of iterations required to obtain a desired approximation to q^* as a function of the encoding size of $x = R(x)$ for RMCs. For arbitrary MPSs, [16] gave no upper bounds on the number of iterations of Newton required as a function of the encoding size of the input MPS. Recently, in [40] an exponential worst-case upper bound was established for Newton's method for as a function of the encoding size of the MPS for computing its LFP to desired precision. The bound in [40] is essentially optimal in several important parameters of the problem.

In the case of 1-exit RMCs, the corresponding equation system $x = R(x)$ is a *probabilistic polynomial system of equations* (PPS). These consist of equations of the form $x_i = R_i(x)$, where $R_i(x)$ is a *probabilistic polynomial*, meaning a multivariate polynomial in the variables x whose monomial coefficients and constant term are all nonnegative and their sum is (at most) 1. A recent result in [19] shows that Newton's method, combined with P-time methods from [28] for qualitative analysis of termination for 1-exit RMCs, can be used to obtain a P-time algorithm for PPSs and 1-exit RMCs (in the standard Turing model of computation) for approximating q^* to within arbitrary desired precision 2^{-j}, for j given in unary. This result also has important consequences for *multi-type branching processes* (BPs) and *stochastic context-free grammars* (SCFGs). For instance, it yields the first P-time algorithm for computing extinction probabilities of BPs, and for computing the probability of generating a given string for *arbitrary* SCFGs (see [19]). See also the recent paper [20], where it is has been further shown that for a very broad class of SCFGs, excluding only some degenerate "deeply critical" SCFGs, Newton's method yields a P-time algorithm for computing within desired precision the probability that the SCFG generates a string in a given regular language, given by a DFA. In particular, [20] shows that this runs in P-time for any SCFG whose parameters are estimated using the standard EM ("inside-outside") method.

In the case of 1-box RMCs, which are essentially equivalent to discrete-time *quasi-birth-death processes* (QBDs), and to *probabilistic one-counter automata*, it was shown in [21] that decomposed Newton's method requires only polynomially many iterations, as a function of the encoding size of $x = R(x)$, and of j, to compute q^* to within additive error 2^{-j}. The vector q^* corresponds to the so called G matrix of a QBD, which is a key to many other analyses of QBDs (see, e.g., [35, 3]), and this thus yields the first P-time algorithm, in the unit-cost arithmetic RAM model of computation, for computing the G matrix of an arbitrary QBD. More recently, in [40], it was shown that with suitable rounding of Newton's method the G matrix can be computed in P-time in the standard Turing model of computation.

Model checking (MoCh): model checking of RMCs was studied in [29], where it was shown how to use TP analysis toward both qualitative and quantitative model checking of RMCs. The algorithms are involved: in brief, given a labeled RMC, A, and a ω-regular

property, say given by a Büchi automaton $\mathcal{B}$, it is possible to use termination probabilities q^* to first define a finite-state MC, called the *conditioned summary chain*, $\mathcal{M}'$, of the "product" of the RMC and the naive determinization $\mathcal{B}'$ of $\mathcal{B}$, and then to boil down the probability of $L(\mathcal{B})$ in the original RMC to the probability of hitting a subset $\mathcal{T}$ of states in $\mathcal{M}'$, where $\mathcal{T}$ can be computed using suitable modifications to the notion of *special pairs*, used earlier for solving MoCh by [13] for finite-state MCs. Furthermore, a different algorithm can be used for properties specified by LTL formulas. For the resulting complexity bounds for the various cases of qualitative and quantitative analysis, see [29], whose results also yield the best available complexity bounds (improving by more than one exponential the prior bounds) for model checking ω-regular and LTL properties of probabilistic pushdown systems, a problem which was first studied in [17].

For model checking 1-box RMCs (equivalently, probabilistic one-counter automata (pOCAs)), a recent paper [8] shows how to use the polynomial time algorithm obtained in [23, 40] for computing (to within any desired precision) the termination probabilities q^* for 1-box RMCs and pOCAs, in order to obtain an algorithm for computing (to within desired precision) the probability of an ω-regular property for pOCAs which, for a fixed ω-regular property, also runs in polynomial time (see also [40]).

Recursive Markov Decision Processes (RMDPs): It is not difficult to generalize the definition of RMCs to define RMDPs, by allowing some nodes of the RMC to be *controlled*. RMDPs were first studied in [24, 30], where it was shown that, unfortunately, even very basic computational problems, such as computing *any non-trivial approximation* of the *optimal (supremum or infimum)* termination probabilities of finitely presented RMDPs is not computable. Furthermore, [24, 30] showed that even qualitative model checking (MoCh) analyses are undecidable already for 1-exit RMDPs.

Fortunately it was also shown in [24, 30] that for *1-exit* RMDPs (1-RMDPs), which correspond also to controlled versions of BPs and SCFGs, it is possible to set up a monotone *max/min probabilistic polynomial system of equations* (max/minPPS), $x = R(x)$, whose LFP, q^*, corresponds precisely to the vector of optimal termination probabilities. A maxPPS (respectively, minPPS), $x = R(x)$ consists of equations $x_i = R_i(x)$, where each $R_i(x)$ has the form $\max\{Q_1(x), \ldots, Q_{k_i}(x)\}$, where each $Q_j(x)$ is a probabilistic polynomial in the variables x. It was furthermore shown in [24, 30] that the controller always has optimal deterministic stackless and memoryless optimal strategies for optimizing termination probability in 1-RMDP. Already for 2-exit RMDPs, it is not even the case that there necessarily exists *any* optimal strategy for maximizing the probability of termination (see [24]). It was subsequently shown in [25, 30] that *qualitative* optimal termination problems for 1-RMDPs can be decided in P-time using a spectral optimization method that requires use of linear programming. The algorithms from [25] for deciding whether optimal termination probability for 1-RMDPs is exactly 1 were later used in [7] in order to show that there is a P-time algorithm for detecting whether there exists a strategy which achieves optimal termination probability 1 of reaching a given vertex of a 1-exit RMC in *any* calling context (any call stack). However, there need not exist any optimal strategy for reaching a vertex in any calling context, even when the supremum probability of doing so is 1, and even the decidability of determining whether the supremum probability is 1 for this problem remains open. Finally, in a recent advance it was shown in [18] that for 1-RMDPs the vector q^* of optimal termination probabilities can be approximated in P-time to within arbitrary desired precision, by using a generaliza-

tion of Newton's method applied to the corresponding max/minPPS equations $x = R(x)$, which converges monotonically to their LFP. The generalized Newton method requires solving an LP in each iteration (in both the maximizing and minimizing cases, which are different).

For 1-box RMDPs, corresponding to controlled QBDs and to MDP extensions of probabilistic one-counter automata, we do not have a corresponding equation system $x = R(x)$ which captures their termination probabilities. Nevertheless, it was shown in [6] and [4] that for both maximizing and minimizing the termination probability in 1-box RMDPs, the qualitative problem of deciding whether the optimal probability is 1 for termination, i.e. for hitting counter value 0 in *any* state, can be decided in P-time using, among other things, linear programming. Subsequently, it was shown in [5] that for a 1-box RMDP one can *approximate* the optimal probability of termination in *any* state in exponential time. Optimal strategies need not exist for maximizing termination probability in 1-box RMDPs [5]. It remains open whether this exponential time upper bound can be improved. Deciding whether the (optimal) termination probability is, say, $\geqslant 1/2$, is already square-root-sum-hard, even for 1-box RMCs ([23]). Apparently harder *selective* termination problems for 1-box RMDPs were also studied in [6], such as whether there is some strategy with which we hit counter value 0 *in a desired control state* with probability 1. It was shown in [6] that this problem is already PSPACE-hard, and that this particular qualitative selective termination problem is decidable. However, the decidability of limit-sure (and quantitative) "selective" termination for 1-box RMDPs remains open.

Recursive Stochastic Games: although we have not discussed *stochastic games* (see, e.g., [39, 10, 31]), we mention that a number of results, in particular about 1-RMDPs, extend naturally to two-player zero-sum *1-exit Recursive Simple Stochastic Games* (1-RSSGs) ([24, 30]) and to *1-exit Recursive Concurrent Stochastic Games* (1-RCSGs) ([26, 27]). In particular, corresponding to 1-RSSGs with the objective (and counter-objective) of maximizing (and minimizing) termination probability, there are monotone min-max-polynomial equations $x = R(x)$ whose LFP yields the vector of termination values starting at each vertex ([24, 30]). Corresponding to 1-RCSGs as shown in [27], there are monotone minimax-polynomial equations, where the value operator, $\mathsf{Val}(M)$, for a 1-shot 2-player zero-sum matrix game M is used in the equations, the LFP of which yields the value vector of the 1-RCSG. It was shown in [25, 30] that deciding whether the value of a 1-RSSG termination game is exactly 1 is in NP $\cap$ co-NP, and that this problem is already at least as hard as Condon's *quantitative* decision problem for finite-state SSGs [10], whereas for finite-state SSGs the qualitative decision problem of deciding whether the value is 1 is known to be in P-time. For 1-RCSG termination games it was shown in [27] that quantitative decision and approximation problems for the game value are solvable in PSPACE using the associated monotone system of equations $x = R(x)$, and it was shown that even the *qualitative* problem deciding whether the game value is 1 is at least as hard as the *square-root sum problem*, which as discussed already is not even known to be in NP. The complexity of analyzing 1-box RSSGs (equivalently, one-counter SSGs) was studied in [6, 4, 5] where some upper and lower bounds were established, but the precise complexity of a number of analysis problems for one-counter SSGs (and one-counter MDPs) remains open.

References

[1] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. B. Miltersen. On the complexity of numerical analysis. *SIAM J. Comput.*, 38(5):1987–2006, 2009. Earlier version appeared in *Proc. 21st IEEE Conference on Computational Complexity*, 2006. 30

[2] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008. 5

[3] D. Bini, G. Latouche, and B. Meini. *Numerical methods for Structured Makov Chains*. Oxford University Press, 2005. 31

[4] T. Brázdil, V. Brozek, and K. Etessami. One-counter stochastic games. In *FSTTCS*, pages 108–119, 2010. 33

[5] T. Brázdil, V. Brozek, K. Etessami, and A. Kucera. Approximating the termination value of one-counter mdps and stochastic games. In *ICALP (2)*, pages 332–343, 2011. 33

[6] T. Brazdil, V. Brozek, K. Etessami, T. Kucera, and D. Wojtczak. One-counter Markov decision processes. In *ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*, 2010. 33

[7] T. Brázdil, V. Brozek, V. Forejt, and A. Kucera. Reachability in recursive Markov decision processes. *Inf. Comput.*, 206(5):520–537, 2008. 32

[8] T. Brázdil, S. Kiefer, and A. Kucera. Efficient analysis of probabilistic programs with an unbounded counter. In *CAV*, pages 208–224, 2011. 32

[9] K. L. Chung. *A Course in Probability Theory*. Academic Press, 3rd edition, 2001. 5, 6

[10] A. Condon. The complexity of stochastic games. *Inf. & Comp.*, 96(2):203–224, 1992. 4, 33

[11] C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In *Proc. of 29th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 338–345, 1988. 12, 19, 23, 24, 25

[12] C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events (extended abstract). In *Proc. of 17th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 443 of *LNCS*, pages 336–349. Springer, 1990. 12, 25

[13] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995. 12, 19, 23, 24, 25, 27, 32

[14] C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. *IEEE Trans. on Automatic Control*, 43(10):1399–1418, 1998. 12, 25, 27

[15] L. de Alfaro, T. A. Henzinger, and O. Kupferman. Concurrent reachability games. In *Proc. 39th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 564–575, 1998. 15

[16] J. Esparza, S. Kiefer, and M. Luttenberger. Computing the least fixed point of positive polynomial systems. *SIAM Journal on Computing*, 39(6):2282–2355, 2010. 31

[17] J. Esparza, A. Kucera, and R. Mayr. Model checking probabilistic pushdown automata. *Logical Methods in Computer Science*, 2(1), 2006. 3, 32

[18] K. Etessami, A. Stewart, and M. Yannakakis. Polynomial time algorithms for branching Markov decision processes and probabilistic min(max) polynomial equations. In *Proc. 39rd Int. Coll. on Automata, Languages, and Programming (ICALP)*, 2012. 5, 32

[19] K. Etessami, A. Stewart, and M. Yannakakis. Polynomial time algorithms for multi-type branching processes and stochastic context-free grammars. In *Proc. of 44th Symp. on Theory of Computing (STOC)*, pages 579–588, 2012. 31

[20] K. Etessami, A. Stewart, and M. Yannakakis. Stochastic context-free grammars, regular languages, and Newton's method. In *Proc. 40th Int. Coll. on Automata, Languages, and Programming (ICALP)*, 2013. 31

[21] K. Etessami, D. Wojtczak, and M. Yannakakis. Quasi-birth-death processes, tree-like QBDs, probabilistic 1-counter automata, and pushdown systems. In *Proc. 5th Int. Symp. on Quantitative Evaluation of Systems (QEST)*, pages 243–253, 2008. 3, 31

[22] K. Etessami, D. Wojtczak, and M. Yannakakis. Recursive stochastic games with positive rewards. In *Proc. 35th ICALP*, pages 71–723, 2008. 5, 14

[23] K. Etessami, D. Wojtczak, and M. Yannakakis. Quasi-birth-death processes, tree-like QBDs, probabilistic 1-counter automata, and pushdown systems. *Perform. Eval.*, 67(9):837–857, 2010. 29, 32, 33

[24] K. Etessami and M. Yannakakis. Recursive Markov decision processes and recursive stochastic games. In *Proc. 32nd ICALP*, pages 891–903, 2005. 32, 33, 35

[25] K. Etessami and M. Yannakakis. Efficient analysis of classes of recursive Markov decision processes and stochastic games. In *Proc. 23rd Symp. on Theoretical Aspects of Comp. Sci. (STACS)*, pages 634–645, 2006. 5, 32, 33, 35

[26] K. Etessami and M. Yannakakis. Recursive concurrent stochastic games. In *33rd Int. Coll. on Automata, Languages, and Programming (ICALP)*, pages 324–335, 2006. 33, 35

[27] K. Etessami and M. Yannakakis. Recursive concurrent stochastic games. *Logical Methods in Computer Science*, 4(4), 2008. (Journal version of [26].). 5, 28, 33

[28] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *J. ACM*, 56(1), 2009. 2, 3, 28, 29, 30, 31

[29] K. Etessami and M. Yannakakis. Model checking of recursive probabilistic systems. *ACM Trans. Comput. Log.*, 13(2), 2012. 23, 28, 29, 31, 32

[30] K. Etessami and M. Yannakakis. Recursive Markov decision processes and recursive stochastic games. *Journal of the ACM (to appear)*, 2015. Combines results in *ICALP'05* & *STACS'06* papers [24, 25]. Manuscript available at: `http://homepages.inf.ed.ac.uk/kousha/j_sub_rmdp_rssg.pdf`. 5, 28, 32, 33

[31] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1997. 33

[32] T. D. Hansen, P. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. In *Innovations in Computer Science (ICS)*, pages 253–263, 2011. 25

[33] T. E. Harris. *The Theory of Branching Processes*. Springer-Verlag, 1963. 2

[34] M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *CAV*, pages 585–591, 2011. `http://www.prismmodelchecker.org`. 5

[35] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. ASA-SIAM series on statistics and applied probability, 1999. 31

[36] J. R. Norris. *Markov Chains*. Cambridge University Press, 1997. 5

[37] A. Pnueli. The temporal logic of programs. In *Proc. 18th Symp. on Foundations of Computer Science*, pages 46–57, 1977. 12

[38] M. L. Puterman. *Markov Decision Processes*. Wiley, 1994. 5, 19, 25, 26

[39] L. Shapley. Stochastic games. *Proc. Nat. Acad. Sci.*, 39:1095–1100, 1953. 5, 33

[40] A. Stewart, K. Etessami, and M. Yannakakis. Upper bounds for Newton's method on monotone polynomial systems, and P-time model checking of probabilistic one-counter automata. In *Proc. of 25th Int. Conf. on Computer Aided Verification (CAV)*, volume 8044 of *LNCS*, pages 495–510. Springer, 2013. 31, 32

[41] M. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. of 26th IEEE FOCS*, pages 327–338, 1985. 15, 24, 27

[42] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st Symp. on Logic in Comp. Sci. (LICS)*, pages 322–331, 1986. 12

[43] T. Wilke. Chapter 6: ω-automata. In J.-E. Pin, editor, *Handbook of Automata Theory*. European Math Society, 2013? 11, 12

[44] D. Wojtczak and K. Etessami. Premo : An analyzer for probabilistic recursive models. In *TACAS*, pages 66–71, 2007. 5

[45] Y. Ye. A new complexity result on solving the Markov decision problem. *Math. Oper. Res.*, 30(3):733–749, 2005. 25

[46] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1-2):343–359, 1996. 4