Model Checking Discounted Temporal Properties *

Luca de Alfaro [a] Marco Faella [a] Thomas A. Henzinger [b,c]
Rupak Majumdar [d] Mariëlle Stoelinga [e]

[a] *CE, University of California, Santa Cruz, USA*

[b] *EECS, University of California, Berkeley, USA*

[c] *Computer and Communication Sciences, EPFL, Switzerland*

[d] *CS, University of California, Los Angeles, USA*

[e] *EWI, University of Twente, The Netherlands*

Abstract

Temporal logic is two-valued: formulas are interpreted as either true or false. When applied to the analysis of stochastic systems, or systems with imprecise formal models, temporal logic is therefore fragile: even small changes in the model can lead to opposite truth values for a specification. We present a generalization of the branching-time logic CTL which achieves robustness with respect to model perturbations by giving a quantitative interpretation to predicates and logical operators, and by discounting the importance of events according to how late they occur. In every state, the value of a formula is a real number in the interval [0,1], where 1 corresponds to truth and 0 to falsehood. The boolean operators *and* and *or* are replaced by min and max, the path quantifiers $\exists$ and $\forall$ determine sup and inf over all paths from a given state, and the temporal operators $\diamond$ and $\square$ specify sup and inf over a given path; a new operator averages all values along a path. Furthermore, all path operators are discounted by a parameter that can be chosen to give more weight to states that are closer to the beginning of the path.

We interpret the resulting logic DCTL over transition systems, Markov chains, and Markov decision processes. We present two semantics for DCTL: a *path* semantics, inspired by the standard interpretation of state and path formulas in CTL, and a *fixpoint* semantics, inspired by the μ-calculus evaluation of CTL formulas. We show that, while these semantics coincide for CTL, they differ for DCTL, and we provide model-checking algorithms for both semantics.

* This research was supported in part by the AFOSR MURI grant F49620-00-1-0327, the ONR grant N00014-02-1-0671, the EU grant IST-004527 (ARTIST2), the DFG/NWO bilateral cooperation project Validation of Stochastic Systems

1 Introduction

Boolean state-transition models are useful for the representation and verification of computational systems, such as hardware and software systems. A boolean state-transition model is a labeled directed graph, whose vertices represent system states, whose edges represent state changes, and whose labels represent boolean observations about the system, such as the truth values of state predicates. Behavioral properties of boolean state-transition systems can be specified in temporal logic [19,4] and verified using model-checking algorithms [4].

For representing systems that are not purely computational but partly physical, such as hardware and software that interact with a physical environment, boolean state-transition models are often inadequate. Many quantitative extensions of state-transition models have been proposed for this purpose, such as models that embed state changes into the real time line, and models that assign probabilities to state changes. These models typically contain real numbers, e.g., for representing time or probabilities. Yet previous research has focused mostly on purely boolean frameworks for the specification and verification of quantitative state-transition models, where observations are truth values of state predicates, and behavioral properties are based on such boolean observations [13,3,1,17]. These boolean specification frameworks are *fragile* with respect to imprecisions in the model: even arbitrarily small changes in a quantitative model can cause different truth values for the specification.

We submit that a proper framework for the specification and verification of quantitative state-transition models should itself be quantitative. To start with, we consider observations that do not have boolean truth values, but real values [16]. Using these quantitative observations, we build a temporal logic for specifying quantitative temporal properties. A Ctl-like temporal logic has three kinds of operators. The first kind are boolean operators such as "and" and "or" for locally combining the truth values of boolean observations. These are replaced by "min" and "max" operators for combining the real values of quantitative observations. In addition, a "weighted average" ($\oplus_c$) operator computes a convex combination of two quantitative observations. The second kind of construct are modal operators such as "always" ($\Box$) and "eventually" ($\Diamond$) for temporally combining the truth values of all boolean observations along an infinite path. These are replaced by "inf" ("lim min") and "sup" ("lim max") operators over infinite sequences of real values. We introduce a "lim avg" ($\triangle$) operator that captures the long-run average value of a quantitative observation. For nondeterministic models, where there is a choice of future behaviors, there is a third kind of construct: the

(VOSS2), and the NSF grants CCR-0132780, CCR-9988172, CCR-0225610, CCR-0234690, and CCF-0427202.

path quantifiers "for-all-possible-futures" ($\forall$) and "for-some-possible-future" ($\exists$) turn path properties into state properties by quantifying over the paths from a given state. These are replaced by "inf-over-all-possible-futures" and "sup-over-all-possible-futures." Once boolean specifications are replaced by quantitative specifications, it becomes possible to discount the future, that is, to give more weight to the near future than to the far away future. This principle is well-understood in economics and in the theory of optimal control [2], but is equally natural in studying quantitative temporal properties of systems [10]. We call the resulting logic DCTL ("Discounted CTL"). While quantitative versions of dynamic logics [16], μ-calculi [14,20,21,10], and Hennessy-Milner logics [11] exist, DCTL is the first temporal logic in which the non-local temporal operators $\diamond$ and $\square$, along with the new temporal operator $\triangle$ and the path quantifiers $\forall$ and $\exists$, are given a quantitative interpretation.

We propose two semantics for DCTL: a *path* semantics and a *fixpoint* semantics. The path semantics is defined as follows. For a discount factor $\alpha < 1$, the $\diamond_\alpha$ (resp. $\square_\alpha$) operator computes the sup (resp. inf) over a path, weighing the value of a state that occurs k steps in the future by a factor α^k. As usual, the operators $\diamond_\alpha$ and $\square_\alpha$ are one the dual of the other. The $\triangle_\alpha$ operator computes the discounted long-run average of the values along a path (see, e.g., [2]), where the value of a state that occurs k steps in the future is again multiplied by a factor α^k; the $\triangle_\alpha$ operator is self-dual. The $\forall$ and $\exists$ operators then combine these values over the paths: in transition systems, $\forall$ and $\exists$ associate with each state the inf and sup of the values for the paths that leave the state; in probabilistic systems, $\forall$ and $\exists$ associate with each state the least and greatest expectation of the value for those paths (for Markov chains, there is a single expected value at each state, but for Markov decision processes, the least and greatest expected value are generally different). Thus, the path semantics of DCTL is obtained by lifting to a quantitative setting the classical interpretation of path and state formulas in CTL.

The fixpoint semantics is obtained by lifting to a quantitative setting the connection between CTL and the μ-calculus [4]. In a transition system, given a set r of states, denote by $\exists\mathrm{Pre}(r)$ the set of all states that have a one-step transition to r. Then, the semantics of $\exists\diamond r$ for a set r of states can be defined as the least fixpoint of the equation $x = r \cup \exists\mathrm{Pre}(x)$, denoted $\mu x.(r \cup \exists\mathrm{Pre}(x))$. We lift this definition to a quantitative, discounted setting by interpreting $\cup$ as pointwise maximum, and $\exists\mathrm{Pre}(x)$ as the maximal expected value of x achievable in one step [10]. For a discount factor $\alpha < 1$, the semantics $\exists\diamond_\alpha r$ is obtained by multiplying the next-step expectation with α, i.e., $\mu x.(r \cup \alpha \cdot \exists\mathrm{Pre}(x))$.

The path and fixpoint semantics coincide on transition systems, but differ on Markov chains (and consequently on Markov decision processes). This is illustrated by the Markov chain in Figure 1. Consider the DCTL formula ϕ:

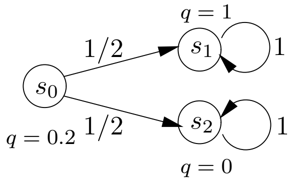

Fig. 1. A Markov chain illustrating the difference between path and fixpoint semantics. The states are labeled with the values taken by observation q.

$\exists\Diamond_\alpha q$, for $\alpha = 0.8$. According to the path semantics, there are two paths from s_0, each followed with probability 1/2: the first path has the discounted sup equal to 0.8, and the second has the discounted sup equal to 0.2; hence, ϕ has the value $(0.8 + 0.2)/2 = 0.5$ at s_0. According to the fixpoint semantics, $q \cup 0.8 \cdot \exists\mathrm{Pre}(q)$ has the value $\max\{0.2, 0.8 \cdot (1 + 0)/2)\} = 0.4$ at s_0, and this is also the value of ϕ at s_0.

To highlight the different perspective taken by the two semantics, consider a water tank, and assume that q represents the daily level of water in the tank (0 is empty, 1 is full). Consider the formula $\exists\Diamond q$.

Setting aside the discounting aspect, in the fixpoint semantics $\exists\Diamond q$ is the expected value of the amount of money we can realize by selling the tank (where a tank with level q has value q), provided each day we choose optimally whether to sell. In the fixpoint semantics we must decide when to stop: the choice of selling the tank, or of waiting for one more day, corresponds to the choice between the two sides q and $\exists\mathrm{Pre}(x)$ of the $\cup$ operator (interpreted as pointwise maximum) in the fixpoint. Hence the fixpoint semantics is suited for system control, since the decision of which side of $\cup$ to take corresponds to a control decision for the system.

In contrast, again setting aside discounting, in the path semantics $\exists\Diamond q$ is the expected value of the maximum level that occurs along a system behavior (discounting accounts for the fact that immediate emergencies are more serious than ones that are farther in the future). In the path semantics, we have no control over stopping: we can only observe the value of q over infinite runs, and compute the expected maximum value it reaches. Such a semantics is well-suited for system specification.

In DCTL, discounting serves two purposes. First, it leads to a notion of "quality" with which a specification is satisfied. For example, assume that we wish to reach a state with a high value of q. Without discounting, the formula $\exists\Diamond q$ has the same value, regardless of the time required to reach q; on the other hand, the formula $\exists\Diamond_\alpha q$, for $\alpha < 1$, has a higher value if the high q value is reached earlier. In other words, discounted reachability properties account not only for *how well* the goal is eventually satisfied (the value of q that is reached), but also for *how soon* is it satisfied. Likewise, if q represents the "level of functionality" of a system, then the specification $\forall\Box_\alpha q$ will have a value that is higher the longer the system functions well, even if the system will eventually always break. Second, discounting is instrumental in achieving robustness with respect to system perturbations. Indeed, we will show that for discount factors smaller than 1, the value of DCTL formulas in both semantics is a continuous function of the values of the numerical quantities (observations, transition probabilities) of the model.

We present algorithms for model checking both semantics of DCTL over transition systems, Markov chains, and Markov decision processes (MDPs). In all cases but one (the $\forall\Diamond$ operator in the fixpoint semantics of MDPs), the algorithms achieve polynomial time-complexity in the size of the system. For transition systems, we present algorithms for $\Box$ and $\Diamond$ that achieve linear-logarithmic running time, improving on the results presented in the preliminary version of this paper [9]. For Markov chains and MDPs, the fixpoint and path semantics are different; while the algorithms for the fixpoint semantics follow the approach of dynamic programming [2], the algorithms for the path semantics are novel (and coincide with those in [9]). Note that, due to the discounting, DCTL is a quantitative logic even when interpreted over purely boolean state-transition systems. As for CTL, the algorithms work recursively on the subformulas of a given formula. Due to the duality among the operators, we need to consider only the cases for $\exists\Diamond$, $\forall\Diamond$, and $\exists\triangle$.

In transition systems, the path and fixpoint semantics coincide. In [9] we presented algorithms for $\exists\Diamond$ and $\forall\Diamond$ that are based on iterating quantitative fixpoint expressions; the resulting time-complexity was quadratic. Here, we present improved algorithms of linear-logarithmic (i.e. $n \log n$) time complexity. The algorithm for $\exists\triangle$ (discounted long-run average along a path) builds on both Karp's algorithm for computing minimum mean-weight cycles and a discounted version of Bellman-Ford for computing shortest paths; the resulting time complexity is cubic in the size of the transition system. In all cases, the time complexity is linear in the size of the formula.

For Markov chains, the fixpoint and path semantics differ. The model-checking algorithms for the fixpoint semantics rely on reductions to linear programming, following a common approach in optimal control [2]. The algorithms for the path semantics are based on a detailed analysis of the behavior of the paths outgoing from each state. In both cases, the time complexity is polynomial in the size of the system. However, the time complexity is exponential in the size of the DCTL formula, due to the fact that the bit-wise encodings of the valuations grows exponentially with respect to the number of nestings of temporal operators (in practice, of course, one would be unlikely to implement arbitrary-precision arithmetic).

In MDPs, the path semantics can be model-checked via reductions to linear programming. The main difficulty in the reduction is that the optimal policy with respect to the property is not necessarily memoryless; thus, we must phrase the linear programming problem in terms of quantities that preserve this past dependency. As in Markov chains, the resulting algorithms are polynomial in the size of the system, and exponential in the size of the formula. Lastly, we consider the model-checking of the fixpoint semantics of DCTL over MDPs. For the operators $\exists\Diamond$ and $\exists\triangle$ (as well as for their duals $\forall\Box$ and $\forall\triangle$), we show that we can compute the required fixpoints via reductions to linear programming, achieving polynomial time-complexity in the size of the system. On the other hand, for the $\forall\Diamond$ operator we present an algorithm of nondeterministic polynomial-time complexity with respect to the size of the system. The difficulty is due to the fact that the $\forall\Diamond$ operator combines a min over nondeterminism (the $\forall$ part) with a max over valuations (the $\Diamond$ part), precluding known avenues of reduction to linear programming, contrary to what was claimed in [9]. It is an open problem whether this algorithm can be improved, making all model-checking algorithms fall into polynomial time with respect to the size of the system.

A related approach to the specification and verification of quantitative systems has been proposed in [18]. There, the authors define an abstract quantitative μ-calculus, based on constraint semirings, which provides a general framework for expressing properties of quantitative transition systems. They provide model-checking algorithms, based on the iterative evaluation of fixpoints, for a restricted class of formulas (c-CTL). They also point out the difference between the path semantics and the fixpoint semantics of the proposed language, in a similar fashion to what was done in the preliminary version of this paper [9] and is restated in the present work. The quantitative μ-calculus defined there subsumes the languages presented in this work, when our definition is restricted to transition systems. However, even for the case of transition systems, our model-checking algorithms improve over the fixpoint iteration in the case of the $\Diamond$ and $\Box$ operators, and they allow for the efficient computation of the $\triangle$ operator, for which the standard fixpoint evaluation need not terminate within a finite number of steps.

2 Discounted CTL

2.1 Syntax

Let Σ be a set of propositions and let A be a set of parameters. The DCTL formulas over (Σ, A) are generated by the grammar

$$\phi ::= r \mid \mathrm{T} \mid \mathrm{F} \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \phi \oplus_c \phi \mid \exists\psi \mid \forall\psi$$
$$\psi ::= \Diamond_c\phi \mid \Box_c\phi \mid \triangle_c\phi$$

where $r \in \Sigma$ is a proposition and $c \in A$ is a parameter. The formulas generated by ϕ are *state formulas*; the formulas generated by ψ are *path formulas*. The DCTL formulas are the state formulas. The $\triangle$-free fragment of DCTL is the set of DCTL formulas with no $\triangle_c \phi$ subformula.

2.2 Semantics for Labeled Transition Systems

We define two semantics for DCTL: the path semantics, and the fixpoint semantics. In the path semantics, the path operators $\Diamond$ and $\Box$ determine the discounted sup and inf values over a path, and the $\exists$ and $\forall$ operators determine the minimum and maximum values of the path formula over all paths from a given state. The fixpoint semantics is defined by lifting to a quantitative setting the usual connection between CTL and μ-calculus.

Discount factors. Let A be a set of parameters. A *parameter interpretation* of A is a function $\langle\cdot\rangle\colon A \to [0,1)$ that assigns to each parameter a real number between 0 and 1, called a *discount factor*. We write $\mathcal{I}_A$ for the set of parameter interpretations of A. We denote by $|q|_b$ the length of the binary encoding of a number $q \in \mathbb{Q}$, and we denote by $|\langle\cdot\rangle|_b = \sum_{c \in A} |\langle c \rangle|_b$ the size of the interpretation $\langle\cdot\rangle$ of A.

Valuations. Let S be a set of states. A *valuation* on S is a function $v\colon S \to [0,1]$ that assigns to each state a real between 0 and 1. The valuation v is *boolean* if $v(s) \in \{0,1\}$ for all $s \in S$. We write $\mathcal{V}_S$ for the set of valuations on S. We write $\mathbf{0}$ for the valuation that maps all states to 0, and $\mathbf{1}$ for the valuation that maps all states to 1. For two real numbers u_1, u_2 and a discount factor $\alpha \in [0,1)$, we write $u_1 \sqcup u_2$ for $\max\{u_1,u_2\}$, $u_1 \sqcap u_2$ for $\min\{u_1,u_2\}$, and $u_1 +_\alpha u_2$ for $(1-\alpha)\cdot u_1 + \alpha \cdot u_2$. We lift operations on reals to operations on valuations in a pointwise fashion; for example, for two valuations $v_1, v_2 \in \mathcal{V}_S$, by $v_1 \sqcup v_2$ we denote the valuation that maps each state $s \in S$ to $v_1(s) \sqcup v_2(s)$.

Labeled transition systems. A (finite-state) *labeled transition system* (LTS) $\mathcal{S} = (S, \delta, \Sigma, [\cdot])$ consists of a finite set S of states, a transition relation $\delta\colon S \to 2^S \backslash \{\emptyset\}$ that assigns to each state a finite nonempty set of successor states, a set Σ of propositions, and a function $[\cdot]\colon \Sigma \to \mathcal{V}_S$ that assigns to each proposition a valuation. We denote by $|\delta|$ the value $\sum_{s \in S} |\delta(s)|$. The LTS $\mathcal{S}$ is *boolean* if for all propositions $r \in \Sigma$, the valuation $[r]$ is boolean. A *path* of $\mathcal{S}$ is an infinite sequence $s_0 s_1 s_2 \ldots$ of states such that $s_{i+1} \in \delta(s_i)$ for all $i \geq 0$. Given a state $s \in S$, we write Traj_s for the set of paths that start in s.

The path semantics. The DCTL formulas over (Σ, A) are evaluated w.r.t. an LTS $\mathcal{S} = (S, \delta, \Sigma, [\cdot])$ whose propositions are Σ, and w.r.t. a parameter interpretation $\langle\cdot\rangle \in \mathcal{I}_A$. Every state formula ϕ defines a valuation $[\![\phi]\!]^{\mathrm{p}} \in \mathcal{V}_S$:

$$
\begin{aligned}
[\![r]\!]^{\mathrm{p}} &= [r] & [\![\phi_1 \vee \phi_2]\!]^{\mathrm{p}} &= [\![\phi_1]\!]^{\mathrm{p}} \sqcup [\![\phi_2]\!]^{\mathrm{p}} \\
[\![\mathrm{T}]\!]^{\mathrm{p}} &= \mathbf{1} & [\![\phi_1 \wedge \phi_2]\!]^{\mathrm{p}} &= [\![\phi_1]\!]^{\mathrm{p}} \sqcap [\![\phi_2]\!]^{\mathrm{p}} \\
[\![\mathrm{F}]\!]^{\mathrm{p}} &= \mathbf{0} & [\![\phi_1 \oplus_c \phi_2]\!]^{\mathrm{p}} &= [\![\phi_1]\!]^{\mathrm{p}} +_{\langle c\rangle} [\![\phi_2]\!]^{\mathrm{p}} \\
[\![\neg\phi]\!]^{\mathrm{p}} &= \mathbf{1} - [\![\phi]\!]^{\mathrm{p}} & [\![\exists\psi]\!]^{\mathrm{p}}(s) &= \sup\{[\![\psi]\!]^{\mathrm{p}}(\rho) \mid \rho \in \mathit{Traj}_s\} \\
& & [\![\forall\psi]\!]^{\mathrm{p}}(s) &= \inf\{[\![\psi]\!]^{\mathrm{p}}(\rho) \mid \rho \in \mathit{Traj}_s\}
\end{aligned}
$$

A path formula ψ assigns a real $[\![\psi]\!]^{\mathrm{p}}(\rho) \in [0,1]$ to each path ρ of $\mathcal{S}$:

$$
\begin{aligned}
[\![\Diamond_c\phi]\!]^{\mathrm{p}}(s_0 s_1 \ldots) &= \sup\{\langle c\rangle^i \cdot [\![\phi]\!]^{\mathrm{p}}(s_i) \mid i \geq 0\} \\
[\![\Box_c\phi]\!]^{\mathrm{p}}(s_0 s_1 \ldots) &= \inf\{1 - \langle c\rangle^i \cdot (1 - [\![\phi]\!]^{\mathrm{p}}(s_i)) \mid i \geq 0\} \\
[\![\triangle_c\phi]\!]^{\mathrm{p}}(s_0 s_1 \ldots) &= (1 - \langle c\rangle) \cdot \textstyle\sum\{\langle c\rangle^i \cdot [\![\phi]\!]^{\mathrm{p}}(s_i) \mid i \geq 0\}.
\end{aligned}
$$

The term $(1 - \langle c\rangle)$ in the definition of $[\![\triangle_c\phi]\!]^{\mathrm{p}}$ is a normalizing factor ensuring that $[\![\triangle_c\phi]\!]^{\mathrm{p}}(\rho) \in [0,1]$ for all paths ρ.

The fixpoint semantics. In this semantics, the DCTL formulas are evaluated with respect to an LTS $\mathcal{S}$ and a parameter interpretation $\langle\cdot\rangle \in \mathcal{I}_A$. Given a valuation $x \in \mathcal{V}_S$, we denote by $\exists\mathrm{Pre}(x) \in \mathcal{V}_S$ the valuation defined by $\exists\mathrm{Pre}(x)(s) = \max\{x(t) \mid t \in \delta(s)\}$, and we denote by $\forall\mathrm{Pre}(x) \in \mathcal{V}_S$ the valuation defined by $\forall\mathrm{Pre}(x)(s) = \min\{x(t) \mid t \in \delta(s)\}$. The fixpoint semantics $[\![\cdot]\!]^{\mathrm{f}}$ for the propositions, the boolean operators, and $\oplus_c$ is similar to the path semantics, only that $[\![\cdot]\!]^{\mathrm{p}}$ is replaced by $[\![\cdot]\!]^{\mathrm{f}}$. The other operators are defined as follows:

$$
\begin{aligned}
[\![\exists\Diamond_c\phi]\!]^{\mathrm{f}} &= \mu x.([\![\phi]\!]^{\mathrm{f}} \sqcup (\mathbf{0} +_{\langle c\rangle} \exists\mathrm{Pre}(x))) && (1) \\
[\![\forall\Diamond_c\phi]\!]^{\mathrm{f}} &= \mu x.([\![\phi]\!]^{\mathrm{f}} \sqcup (\mathbf{0} +_{\langle c\rangle} \forall\mathrm{Pre}(x))) && (2) \\
[\![\exists\Box_c\phi]\!]^{\mathrm{f}} &= \mu x.([\![\phi]\!]^{\mathrm{f}} \sqcap (\mathbf{1} +_{\langle c\rangle} \exists\mathrm{Pre}(x))) && (3) \\
[\![\forall\Box_c\phi]\!]^{\mathrm{f}} &= \mu x.([\![\phi]\!]^{\mathrm{f}} \sqcap (\mathbf{1} +_{\langle c\rangle} \forall\mathrm{Pre}(x))) && (4) \\
[\![\exists\triangle_c\phi]\!]^{\mathrm{f}} &= \mu x.([\![\phi]\!]^{\mathrm{f}} +_{\langle c\rangle} \exists\mathrm{Pre}(x)) && (5) \\
[\![\forall\triangle_c\phi]\!]^{\mathrm{f}} &= \mu x.([\![\phi]\!]^{\mathrm{f}} +_{\langle c\rangle} \forall\mathrm{Pre}(x)) && (6)
\end{aligned}
$$

Above, for a function $F\colon \mathcal{V}_S \to \mathcal{V}_S$, the notation $\mu x.F(x)$ indicates the unique valuation x_* such that $x_* = F(x_*)$. Uniqueness of the fixpoints is proved in Theorem 1 for the more general case of Markov decision processes.

2.3 Semantics for Markov Processes

Given a finite set S, let $\mathrm{Distr}(S)$ be the set of probability distributions over S; for $a \in \mathrm{Distr}(S)$, we denote by $\mathrm{Supp}(a) = \{s \in S \mid a(s) > 0\}$ the support of a. A probability distribution a over S is *deterministic* if $a(s) \in \{0,1\}$ for all $s \in S$.

Markov decision processes. A (finite-state) *Markov decision process* (MDP) $\mathcal{S} = (S, \tau, \Sigma, [\cdot])$ consists of a finite set S of states, a probabilistic transition relation $\tau\colon S \to 2^{\mathrm{Distr}(S)} \setminus \{\emptyset\}$, which assigns to each state a finite nonempty set of probability distributions over the successor states, a set Σ of propositions, and a function $[\cdot]\colon \Sigma \to \mathcal{V}_S$ that assigns to each proposition a valuation. The MDP $\mathcal{S}$ is *boolean* if for all propositions $r \in \Sigma$, the valuation $[r]$ is boolean. We denote by $|\tau|_b$ the length of the binary encoding of τ, defined by $\sum_{s\in S} \sum_{a\in\tau(s)} \sum_{t\in\mathrm{Supp}(a)} |a(t)|_b$, and we denote by $|[\cdot]|_b = \sum_{q\in\Sigma} \sum_{s\in S} |[q](s)|_b$ the size of the binary encoding of $[\cdot]$. Then, the binary size of $\mathcal{S}$ is given by $|\mathcal{S}|_b = |\tau|_b + |[\cdot]|_b$.

A finite (resp. infinite) *path* of $\mathcal{S}$ is a finite (resp. infinite) sequence $s_0 s_1 s_2 \ldots s_m$ (resp. $s_0 s_1 s_2 \ldots$) of states such that for all $i < m$ (resp. $i \in \mathbb{N}$) there is $a_i \in \tau(s_i)$ with $s_{i+1} \in \mathrm{Supp}(a_i)$. We denote by *FTraj* and *Traj* the sets of finite and infinite paths of $\mathcal{S}$ respectively; for $s \in S$, we denote by Traj_s the infinite paths starting from s. A *strategy* π for $\mathcal{S}$ is a mapping from *FTraj* to $\mathrm{Distr}(\bigcup_{s\in S} \tau(s))$: once the MDP has followed the path $s_0 s_1 \ldots s_m \in \mathit{FTraj}$, the strategy π prescribes the probability $\pi(s_0 s_1 \ldots s_m)(a)$ of using a next-state distribution $a \in \tau(s_m)$. For all $s_0 s_1 \ldots s_m \in \mathit{FTraj}$, we require that $\mathrm{Supp}(\pi(s_0 s_1 \ldots s_m)) \subseteq \tau(s_m)$. Thus, under strategy π, after following a finite path $s_0 s_1 \ldots s_m$ the MDP takes a transition to state s_{m+1} with probability $\sum_{a\in\tau(s_m)} a(s_{m+1}) \cdot \pi(s_0 s_1 \ldots s_m)(a)$. We denote by Π the set of all strategies for $\mathcal{S}$. The transition probabilities corresponding to a strategy π, together with an initial state s, give rise to a probability space $(\mathit{Traj}_s, \mathcal{B}_s, \mathrm{Pr}_s^\pi)$, where $\mathcal{B}_s$ is the set of measurable subsets of 2^{Traj_s}, and Pr_s^π is the probability measure over $\mathcal{B}_s$ induced by the next-state transition probabilities described above [15,22]. Given a random variable X over this probability space, we denote its expected value by $\mathrm{E}_s^\pi[X]$. For $l \in \mathbb{N}$, the random variable $Z_l\colon \mathit{Traj}_{s_i} \to S$ defined by $Z_l(s_0 s_1 \ldots) = s_l$ yields the state of the stochastic process after l steps.

Special cases of MDPs: Markov chains and transition systems. Markov chains and LTSs can be defined as special cases of MDPs. An MDP $\mathcal{S} = (S, \tau, \Sigma, [\cdot])$ is a *Markov chain* if $|\tau(s)| = 1$ for all $s \in S$. It is customary to specify the probabilistic structure of a Markov chain via its *probability transition matrix* $P = [p_{s,t}]_{s,t\in S}$, defined for all $s, t \in S$ by $p_{s,t} = a(t)$, where a is the unique distribution $a \in \tau(s)$. An initial state $s \in S$ completely determines a probability space $(\mathit{Traj}_s, \mathcal{B}_s, \mathrm{Pr}_s)$, and for a random variable X over this probability space, we let $\mathrm{E}_s[X]$ denote its expectation. An MDP $\mathcal{S} = (S, \tau, \Sigma, [\cdot])$ is an LTS if, for all $s \in S$ and all $a \in \tau(s)$, the distribution a is deterministic; in that case, we define $\delta\colon S \to 2^S$ by $\delta(s) = \{t \in S \mid \exists a \in \tau(s).\, a(t) = 1\}$.

The path semantics. The DCTL formulas over (Σ, A) are evaluated with respect to a MDP $\mathcal{S} = (S, \tau, \Sigma, [\cdot])$ and with respect to a parameter interpretation $\langle\cdot\rangle \in \mathcal{I}_A$. The semantics $[\![\psi]\!]^{\mathrm{p}}$ of a path formula ψ is defined as for LTSs; we note that $[\![\psi]\!]^{\mathrm{p}}$ is a random variable over the probability space $(\mathit{Traj}_s, \mathcal{B}_s, \Pr_s)$. Every state formula ϕ defines a valuation $[\![\phi]\!]^{\mathrm{p}} \in \mathcal{V}_S$: the clauses for propositions, boolean operators, and $\oplus_c$ are as for LTSs; the clauses for $\exists$ and $\forall$ are as follows:

$$\begin{aligned}
[\![\exists\psi]\!]^{\mathrm{p}}(s) &= \sup\{\mathrm{E}_s^\pi([\![\psi]\!]^{\mathrm{p}}) \mid \pi \in \Pi\}, \\
[\![\forall\psi]\!]^{\mathrm{p}}(s) &= \inf\{\mathrm{E}_s^\pi([\![\psi]\!]^{\mathrm{p}}) \mid \pi \in \Pi\}.
\end{aligned}$$

The fixpoint semantics. Given a valuation x: $S \to [0,1]$, we denote by $\exists\mathrm{Pre}(x)$: $S \to [0,1]$ the valuation defined by $\exists\mathrm{Pre}(x)(s) = \max_{a\in\tau(s)} \sum_{t\in S} x(t)\cdot a(t)$, and we denote by $\forall\mathrm{Pre}(x)$: $S \to [0,1]$ the valuation defined by $\forall\mathrm{Pre}(x)(s) = \min_{a\in\tau(s)} \sum_{t\in S} x(t)a(t)$. With this notation, the fixpoint semantics $[\![\cdot]\!]^{\mathrm{f}}$ is defined by the same clauses as for LTSs.
For a valuation $x \in \mathcal{V}_S$, we denote by $\|x\|^\infty$ the infinity norm of x; namely $\|x\|^\infty = \max\{x(s) \mid s \in S\}$. For the fixpoint semantics to be well–defined, we have to show that the expressions on the right hand side of (1)–(6) have a unique fixpoint. Given an operator $F : \mathcal{V}_S \to \mathcal{V}_S$, and a constant $\beta \in [0,1]$, we say that F is a β–contraction if and only if, for all $x, y \in \mathcal{V}_S$, $\|F(x) - F(y)\|^\infty \leq \beta \cdot \|x - y\|^\infty$.

Lemma 1 *The operators occurring in the right hand side of (1)–(6) are $\langle c\rangle$–contractions.*

Proof. Let $\alpha = \langle c\rangle$ and $q = [\![\phi]\!]^{\mathrm{f}}$. Given two valuations $x, y \in \mathcal{V}_S$, let $\|x - y\|^\infty = \xi$. Let F be the operator used in (1), namely $F(x) = q \sqcup \alpha\exists\mathrm{Pre}(x)$. For all $s \in S$, there are $a, b \in \tau(s)$ such that:

$$\begin{aligned}
F(x)(s) = q(s) \sqcup \alpha \sum_{t\in S} x(t)a(t), \\
F(y)(s) = q(s) \sqcup \alpha \sum_{t\in S} y(t)b(t).
\end{aligned}$$

We prove that $|F(x)(s) - F(y)(s)| \leq \alpha \cdot \xi$. The result is trivial if $F(x)(s) = F(y)(s) = q(s)$. If $F(x)(s) = q(s)$ and $F(y)(s) > q(s)$ we have

$$|F(x)(s) - F(y)(s)| = F(y)(s) - q(s) \leq \alpha \left(\sum_{t\in S} y(t)b(t) - \sum_{t\in S} x(t)a(t)\right).$$

Symmetrically for the case when $F(y)(s) = q(s)$ and $F(x)(s) > q(s)$. Thus, in all cases it is sufficient to prove that $|\sum_{t\in S} x(t)a(t) - \sum_{t\in S} y(t)b(t)| \leq \xi$. Assume by contradiction that $\sum_{t\in S} x(t)a(t) - \sum_{t\in S} y(t)b(t) > \xi$. Then, the value $\exists\mathrm{Pre}(y)(s)$ can be increased by taking action a instead of b, in contradiction with the hypothesis that b is the best action from s. Formally, we have

$$|\sum_{t\in S} y(t)a(t) - \sum_{t\in S} x(t)a(t)| = |\sum_{t\in S}(y(t) - x(t))a(t)| \leq |\sum_{t\in S} \xi a(t)| = \xi,$$

and thus $\sum_{t \in S} y(t)a(t) \geq \sum_{t \in S} x(t)a(t) - \xi > \sum_{t \in S} y(t)b(t)$. Similar arguments hold for the remaining operators. $\square$

As a consequence of this lemma, and by the contraction mapping theorem, we have immediately that the r.h.s. of (1)–(6) have a unique fixpoint, showing that the fixpoint semantics is well-defined.

Theorem 1 *The right hand side of (1)–(6) have a unique fixpoint.*

2.4 Properties of DCTL

Throughout the rest of the paper, unless differently specified, we fix a parameter interpretation $\langle\cdot\rangle$, a set of propositions Σ, a proposition $r \in \Sigma$, and a parameter c and write $[r] = q$ and $\langle c \rangle = \alpha$. We omit the superscripts p and f and just write $[\![\phi]\!]$ if the path and fixpoint semantics of ϕ coincide.

2.4.1 Duality laws. For all state formulas ϕ over (Σ, A), all MDPs with propositions Σ, and all parameter interpretations of A and $* \in \{\mathrm{p}, \mathrm{f}\}$, we have the following equivalences:

$$[\![\neg\exists\Diamond_c\phi]\!]^* = [\![\forall\Box_c\neg\phi]\!]^* \qquad [\![\neg\exists\Box_c\phi]\!]^* = [\![\forall\Diamond_c\neg\phi]\!]^*, \qquad [\![\neg\exists\triangle_c\phi]\!]^* = [\![\forall\triangle_c\neg\phi]\!]^*.$$

In particular, we see that $\triangle_c$ is self-dual and that a minimalist definition of DCTL will omit one of $\{\mathrm{T}, \mathrm{F}\}$, one of $\{\vee, \wedge\}$, and one of $\{\exists, \forall, \Diamond, \Box\}$ [1].

2.4.2 Comparing the two semantics. We show that the path and fixpoint semantics coincide over transition systems, and over Markov systems with boolean propositions (for non-nested formulas), but do not coincide in general over (non-boolean) Markov chains. This result indicates that the standard connection between CTL and μ-calculus breaks down as soon as we consider *both* probabilistic systems and quantitative valuations. The reason, essentially, is that in probabilistic systems with quantitative evaluations, the operator $\sqcup$ does not commute with the expectation operator E. We start by proving that the two semantics always coincide for the $\triangle_c$ operator.

Theorem 2 *For all MDPs with propositions Σ, all parameter interpretations of c, and all $r \in \Sigma$, we have $[\![\exists\triangle_c r]\!]^{\mathrm{p}} = [\![\exists\triangle_c r]\!]^{\mathrm{f}}$ and $[\![\forall\triangle_c r]\!]^{\mathrm{p}} = [\![\forall\triangle_c r]\!]^{\mathrm{f}}$.*

Proof. We prove the result for $\exists$, as the case for $\forall$ is analogous. Let $v = [\![\exists\triangle_c r]\!]^{\mathrm{f}}$ and $u = [\![\exists\triangle_c r]\!]^{\mathrm{p}}$. Writing out the definitions of the fixpoint and path

[1] One cannot remove, say, both $\exists$ and $\Diamond$ because the negation is not allowed in front of path formulas.

semantics, we have, for all $s \in S$:

$$v(s) = (1-\alpha)\cdot q(s) + \alpha \cdot \max_{a \in \tau(s)} \sum_{t \in S} v(t)\cdot a(t) \tag{7}$$

$$u(s) = (1-\alpha) \sup_{\pi \in \Pi} \mathrm{E}_s^\pi \Big[\sum_{i \geq 0} \alpha^i q(Z_i) \Big]. \tag{8}$$

To prove that $u = v$, we prove that u is a fixpoint of (7). For all $s \in S$, we have:

$$\begin{aligned}
&(1-\alpha)\cdot q(s) + \alpha \cdot \max_{a \in \tau(s)} \sum_{t \in S} a(t)\cdot u(t) \\
&= (1-\alpha)\cdot q(s) + \alpha \cdot \max_{a \in \tau(s)} \sum_{t \in S} a(t)\cdot (1-\alpha)\cdot \sup_{\pi \in \Pi} \mathrm{E}_t^\pi \Big[\sum_{i \geq 0} \alpha^i q(Z_i) \Big] \\
&= (1-\alpha)\cdot \Big[q(s) + \alpha \cdot \max_{a \in \tau(s)} \sum_{t \in S} a(t)\cdot \sup_{\pi \in \Pi} \mathrm{E}_t^\pi \Big[\sum_{i \geq 0} \alpha^i q(Z_i) \Big] \Big] \\
&= (1-\alpha)\cdot \Big[q(s) + \sup_{\pi \in \Pi} \mathrm{E}_s^\pi \Big[\sum_{i \geq 1} \alpha^i q(Z_i) \Big] \Big] \\
&= (1-\alpha)\cdot \sup_{\pi \in \Pi} \mathrm{E}_s^\pi \Big[\sum_{i \geq 0} \alpha^i q(Z_i) \Big] = u(s).
\end{aligned}$$

□

In the following result, we prove that on LTSs the two semantics coincide for $\diamond_c r$ and $\square_c r$ formulas. We first introduce some notation. It is a classical result from fixpoint theory that $[\![\forall \diamond_c r]\!]^{\mathrm{f}} = \lim_{n\to\infty} v_n$, where v_n is defined as follows.

$$v_0(s) = q(s) \qquad v_{n+1}(s) = q(s) \sqcup \alpha \cdot \min\{v_n(s') \mid s' \in \delta(s)\}.$$

Let $[\![\forall \diamond_c^k r]\!]^{\mathrm{p}}$ denote the path semantics of the formula $\forall \diamond_c r$ when only the first $k+1$ states of each trajectory are considered, that is,

$$[\![\forall \diamond_c^k r]\!]^{\mathrm{p}}(s) = \inf_{s_0 s_1 \ldots \in \mathit{Traj}(s)} \ \sup_{0 \leq i \leq k} \alpha^i \cdot q(s_i).$$

Then, the following holds.

Lemma 2 *For each step $k \geq 0$, we have $v_k = [\![\forall \diamond_c^k r]\!]^{\mathrm{p}}$.*

Proof. The case $k=0$ is trivial, since $v_0(s) = q(s) = \inf_{s_0 s_1 \ldots \in \mathit{Traj}(s)} \min\{\alpha^0 \cdot q(s_0)\}$. For the inductive step, assume the thesis holds for some k. Then,

$$\begin{aligned}
[\![\forall \Diamond_c^{k+1} r]\!]^{\mathrm{p}}(s) &= \inf_{\rho \in \mathit{Traj}(s)} \sup_{0 \le i \le k+1} \alpha^i \cdot q(s_i) \\
&= \inf_{\rho \in \mathit{Traj}(s)} \max\{q(s), \sup_{0 < i \le k+1} \alpha^i \cdot q(s_i)\} \\
&= \max\{q(s), \inf_{\rho \in \mathit{Traj}(s)} \sup_{0 < i \le k+1} \alpha^i \cdot q(s_i)\} \\
&= \max\{q(s), \alpha \cdot \min_{s' \in \delta(s)} \inf_{\rho \in \mathit{Traj}(s')} \sup_{0 \le i \le k+1} \alpha^i \cdot q(s_i)\} \\
&= v_{k+1}(s).
\end{aligned}$$

□

The following theorem summarizes the relations between the semantics.

Theorem 3 *The following assertions hold:*

(1) For all LTSs with propositions Σ, all parameter interpretations of A, and all DCTL *formulas ϕ over (Σ, A), we have $[\![\phi]\!]^{\mathrm{p}} = [\![\phi]\!]^{\mathrm{f}}$.*
(2) For all boolean MDPs with propositions Σ, all parameter interpretations of A, and all DCTL *formulas ϕ over (Σ, A) that contain no nesting of path quantifiers, we have $[\![\phi]\!]^{\mathrm{p}} = [\![\phi]\!]^{\mathrm{f}}$.*
(3) There is a Markov chain $\mathcal{S}$ with propositions Σ, a parameter interpretation A, and a DCTL *formula ϕ over (Σ, A) such that $[\![\phi]\!]^{\mathrm{p}} \neq [\![\phi]\!]^{\mathrm{f}}$.*

Proof. Part 1 is proved by structural induction on ϕ. The cases $\Box$ and $\Diamond$ are a consequence of Lemma 2. The case $\triangle$ is a consequence of Theorem 2.

Part 2 follows from the equivalence between the linear and the branching semantics of μ-calculus in the case of strongly guarded formulas, as detailed in Theorem 6 of [7] and Theorem 3 of [10].

Part 3 is witnessed by the Markov chain in Figure 1. Formally, $\mathcal{S} = (S, \tau, \Sigma, [\cdot])$ with $S = \{s_0, s_1, s_2\}$, $\Sigma = \{r\}$, and $[s_0] = 0.2$, $[s_1] = 1$, and $[s_2] = 0$. From state s_0, there are two transitions, to s_1 and s_2, having probability 1/2 each; states s_1 and s_2 are sinks. We consider the DCTL formula $\exists \Diamond_c r$, along with a discount factor interpretation such that $\alpha = 0.8$. According to the path semantics, there are two paths from s_0, each followed with probability 1/2: the first path has discounted sup equal to 0.8, and the second has discounted sup equal to 0.2; hence, $[\![\exists \Diamond_c r]\!]^{\mathrm{p}}(s_0) = (0.8 + 0.2)/2 = 0.5$. According to the fixpoint semantics, $\alpha \cdot \exists \mathrm{Pre}(q)$ at s_0 is $0.8(1+0)/2 = 0.4$, and $\max\{0.2, 0.4\} = 0.4$; thus, $[\![\exists \Diamond_c r]\!]^{\mathrm{f}}(s_0) = 0.4$. □

2.4.3 Robustness. Consider two MDPs $\mathcal{S} = (S, \tau, \Sigma, [\cdot])$ and $\mathcal{S}' = (S, \tau', \Sigma, [\cdot]')$ with the same state space S and the same set Σ of propositions. We define

$$\|\mathcal{S}, \mathcal{S}'\| = \max_{s \in S} \Big\{ \max_{r \in \Sigma} |[r](s) - [r]'(s)|, \\ \max_{a \in \tau(s)} \min_{b \in \tau'(s)} \sum_{s' \in S} |a(s') - b(s')|, \\ \max_{b \in \tau'(s)} \min_{a \in \tau(s)} \sum_{s' \in S} |a(s') - b(s')| \Big\}.$$

It is not difficult to see that $\|\cdot, \cdot\|$ is a metric on the MDPs with state space S. Such metric first considers one state at a time. For each state, the local distance is given by the maximum difference between the value of a proposition in said state in the two systems. The one-step distance, instead, considers the best way a transition from the first system can be matched by a transition in the second, and viceversa. For each state, the maximum of the local distance and the one-step distance is taken. Finally, the maximum over all states is taken.

For an MDP $\mathcal{S}$ and a parameter interpretation $\langle\cdot\rangle$, we write $[\![\cdot]\!]^{\mathrm{f}}_{\mathcal{S},\langle\cdot\rangle}$ and $[\![\cdot]\!]^{\mathrm{p}}_{\mathcal{S},\langle\cdot\rangle}$ to denote the two semantics functions defined on $\mathcal{S}$ with respect to $\langle\cdot\rangle$. The following theorem characterizes the continuity of the fixpoint and path semantics.

Theorem 4 *Let $\mathcal{S}$ and $\mathcal{S}'$ be two MDPs with state space S, and let $\langle\cdot\rangle$ be a parameter interpretation.*

(1) For all $\epsilon > 0$, for all DCTL *formulas ϕ and for all states $s \in S$, if $\|\mathcal{S}, \mathcal{S}'\| \leq \epsilon$, then $|[\![\phi]\!]^{\mathrm{f}}_{\mathcal{S},\langle\cdot\rangle}(s) - [\![\phi]\!]^{\mathrm{f}}_{\mathcal{S}',\langle\cdot\rangle}(s)| \leq \epsilon$.*

(2) For all DCTL *formulas ϕ, for all $\epsilon > 0$, there is a $\delta > 0$ such that for all states $s \in S$, if $\|\mathcal{S}, \mathcal{S}'\| \leq \delta$, then $|[\![\phi]\!]^{\mathrm{p}}_{\mathcal{S},\langle\cdot\rangle}(s) - [\![\phi]\!]^{\mathrm{p}}_{\mathcal{S}',\langle\cdot\rangle}(s)| \leq \epsilon$.*

Proof. We first prove statement (1), by induction on the structure of formulas. Fix $\epsilon > 0$. If ϕ is a proposition r, then $|r(s) - r(s)| \leq \|\mathcal{S}, \mathcal{S}'\| \leq \epsilon$. The cases T, F are obvious. If $\phi = \neg\phi'$, then

$$|[\![\neg\phi']\!]^{\mathrm{f}}_{\mathcal{S}}(s) - [\![\neg\phi']\!]^{\mathrm{f}}_{\mathcal{S}'}(s)| = |1 - [\![\phi']\!]^{\mathrm{f}}_{\mathcal{S}}(s) - 1 + [\![\phi']\!]^{\mathrm{f}}_{\mathcal{S}'}(s)|$$

and the result follows by induction. Similarly, Boolean operations are trivial by induction.

Now consider the path formulas. The technical result we need is that $|\exists\mathrm{Pre}(f)_{\mathcal{S}} - \exists\mathrm{Pre}(f)_{\mathcal{S}'}| \leq \|\mathcal{S}, \mathcal{S}'\|$ for all valuations f, where $\exists\mathrm{Pre}(f)_{\mathcal{S}}$ is the predecessor operator in MDP $\mathcal{S}$, and similarly, $\exists\mathrm{Pre}(f)_{\mathcal{S}'}$ is the predecessor operator in $\mathcal{S}'$.

In order to prove the result for path formulas, we use the fixpoint definitions of the semantics of the path formulas. Also, we only look at formulas of the form $\exists\psi$, since $[\![\forall\psi]\!]^{\mathrm{f}} = 1 - [\![\exists\neg\psi]\!]^{\mathrm{f}}$ will follow from the induction hypothesis. We show the proof for $\phi = \exists\Diamond_c\phi'$. By induction on the structure of formulas, $|[\![\phi']\!]^{\mathrm{f}}_{\mathcal{S}}(s) - [\![\phi']\!]^{\mathrm{f}}_{\mathcal{S}'}(s)| \leq \epsilon$ whenever $\|\mathcal{S},\mathcal{S}'\| \leq \epsilon$. The sequence $x_0 = \mathbf{0}$, $x_{n+1} = [\![\phi']\!]^{\mathrm{f}}_{\mathcal{S}} \vee \alpha\exists\mathrm{Pre}(x_n)_{\mathcal{S}}$ converges to $[\![\phi]\!]^{\mathrm{f}}_{\mathcal{S}}$. Similarly, the sequence $y_0 = \mathbf{0}$, $y_{n+1} = [\![\phi']\!]^{\mathrm{f}}_{\mathcal{S}'} \vee \alpha\exists\mathrm{Pre}(y_n)_{\mathcal{S}'}$ converges to $[\![\phi]\!]^{\mathrm{f}}_{\mathcal{S}'}$. We shall use induction on n to show that these two sequences are close. For the base case, we have $|x_0(s) - y_0(s)| = 0 \leq \epsilon$. Assume by induction on n that $|x_n(s) - y_n(s)| \leq \epsilon$ whenever $\|\mathcal{S},\mathcal{S}'\| \leq \epsilon$. Then (the induction case)

$$|x_{n+1}(s) - y_{n+1}(s)| = |([\![\phi']\!]^{\mathrm{f}}_{\mathcal{S}} \vee \alpha\exists\mathrm{Pre}(x_n))(s) - ([\![\phi']\!]^{\mathrm{f}}_{\mathcal{S}'} \vee \alpha\exists\mathrm{Pre}(y_n))(s)| \leq \epsilon.$$

Thus, $|[\![\exists\Diamond_c\phi']\!]^{\mathrm{f}}_{\mathcal{S}}(s) - [\![\exists\Diamond_c\phi']\!]^{\mathrm{f}}_{\mathcal{S}'}(s)| \leq \epsilon$ whenever $\|\mathcal{S},\mathcal{S}'\| \leq \epsilon$. The cases for $\exists\Box$ and $\exists\triangle$ are similar.

We now proceed to statement (2). Notice that unlike statement (1), we first fix a formula, and let δ depend both on ϵ and on the formula. We shall prove the theorem by induction on the structure of the formula. The cases for propositions and boolean operations are as before, we focus on path formulas. Again, we only consider existential formulas.

Consider the formula $\phi \equiv \exists\Diamond_c\phi'$. By induction, we have proved for ϕ' that for every $\epsilon > 0$ there is $\delta > 0$ such that $|[\![\phi']\!]^{\mathrm{p}}_{\mathcal{S}}(s) - [\![\phi']\!]^{\mathrm{p}}_{\mathcal{S}'}(s)| \leq \epsilon$ whenever $\|\mathcal{S},\mathcal{S}'\| \leq \delta$. Now fix an $\epsilon > 0$. We give a simple bound on δ such that $|[\![\phi]\!]^{\mathrm{p}}_{\mathcal{S}}(s) - [\![\phi]\!]^{\mathrm{p}}_{\mathcal{S}'}(s)| \leq \epsilon$ whenever $\|\mathcal{S},\mathcal{S}'\| \leq \delta$. First, we need only look at finite paths: choose N such that $c^N < \frac{\epsilon}{2}$, where c is the discount factor. Then, the contribution of any term occurring more than N steps in the future the path is bounded by $\epsilon/2$. The difference $|[\![\exists\Diamond_c\phi']\!]^{\mathrm{p}}_{\mathcal{S}} - [\![\exists\Diamond_c\phi']\!]^{\mathrm{p}}_{\mathcal{S}'}|$ is certainly bounded by $|S|^N \cdot (N\epsilon') \cdot \epsilon'$, where the first term gives the number of sequences of length N, the second gives the difference in probabilities in the two MDPs along any behavior of length N, and the third term gives the difference in the valuations of ϕ' in the two MDPs. The value ϵ' will be chosen judiciously as follows. For the above bound to satisfy $|S|^N N\epsilon'^2 \leq \epsilon/2$, we must set $\epsilon' < (\frac{\epsilon}{2|S|^N N})^{\frac{1}{2}}$. Now by induction on ϕ', we construct a δ' such that $|[\![\phi']\!]^{\mathrm{p}}_{\mathcal{S}}(s) - [\![\phi']\!]^{\mathrm{p}}_{\mathcal{S}'}(s)| \leq \epsilon'$ whenever $\|\mathcal{S},\mathcal{S}'\| \leq \delta'$, where we further ensure that $\delta' \leq \epsilon'$ (by taking the minimum of the constructed δ and ϵ'). Then, this δ' has the property that whenever $\|\mathcal{S},\mathcal{S}'\| \leq \delta'$, we have $|[\![\phi]\!]^{\mathrm{p}}_{\mathcal{S}}(s) - [\![\phi]\!]^{\mathrm{p}}_{\mathcal{S}'}(s)| \leq \epsilon$. The case $\exists\Box_c\phi'$ is similar. Finally, the case $\exists\triangle_c\phi'$ follows from part (1) because the path semantics is the same as the fixpoint semantics for $\triangle_c$. $\Box$

Notice that in the continuity statement for the path semantics, δ depends on the formula in addition to ϵ. In general, an iterated application of the $\exists\Diamond$ operator can amplify an arbitrary small difference in probabilities, as shown by the following example.

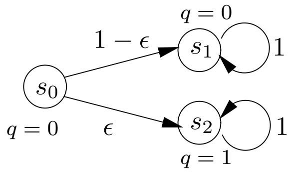

Fig. 2. A Markov chain illustrating Example 1.

Example 1 Consider the three-state Markov chain $\mathcal{S} = (\{s_0, s_1, s_2\}, \tau, \{r\}, [\cdot])$ in Figure 2. As shown in the picture, $\tau(s_0)$ is the distribution that chooses s_1 with probability $1 - \epsilon$ and s_2 with probability ϵ, and $\tau(s_i)$ chooses s_i with probability 1 for $i = 1, 2$. We then have $q(s_0) = q(s_1) = 0$ and $q(s_2) = 1$. Consider the Markov chain $\mathcal{S}'$ that differs from $\mathcal{S}$ in that $\tau(s_0)$ chooses s_1 with probability 1. Then $\|\mathcal{S}, \mathcal{S}'\| = \epsilon$. Now consider the formulas $(\exists\Diamond_c)^n r$, for $n \geq 1$. Let $x_n = [\![(\exists\Diamond_c)^n r]\!]^{\mathrm{p}}_{\mathcal{S},\langle\cdot\rangle}(s_0)$. Then $x_{n+1} = (1 - \epsilon) \cdot x_n + \alpha \cdot \epsilon$, and the limit as n goes to ∞ is α. On the other hand, $[\![(\exists\Diamond_c)^n r]\!]^{\mathrm{p}}_{\mathcal{S}',\langle\cdot\rangle}(s_0) = 0$ for all n.

3 Model Checking DCTL

The model-checking problem of a DCTL formula ϕ over a system with respect to one of the two semantics $* \in \{\mathrm{p}, \mathrm{f}\}$ consists in computing the value $[\![\phi]\!]^*(s)$ for all states s of the system under consideration. Similarly to CTL model checking [4], we recursively consider one of the subformulas ψ of ϕ and compute the valuation $[\![\psi]\!]^*$. Then we replace ψ in ϕ by a new proposition p_ψ with $[p_\psi] = [\![\psi]\!]^*$. Because of the duality laws stated in Section 2.4.1, it suffices to focus on model checking formulas of the forms $\exists\Diamond_c r$, $\forall\Diamond_c r$, and $\forall\triangle_c r$, for a proposition $r \in \Sigma$. We will present the algorithms, for both semantics, over transition systems in Section 3.1, over Markov chains in Section 3.2, and over MDPs in Section 3.3.

For complexity analyses, we assume that operations on reals (comparison, addition, and multiplication) can be performed in constant time: in other words, we provide the asymptotic complexity of each algorithm in terms of the number of arithmetic operations.

3.1 Model Checking DCTL over Transition Systems

We fix a finite LTS $\mathcal{S} = (S, \delta, \Sigma, [\cdot])$. As stated in Theorem 3, the two semantics of DCTL coincide over LTSs. Hence, only one algorithm is needed to model check a formula in either semantics.

The fixpoint semantics of $\exists\Diamond_c r$ and $\forall\Diamond_c r$ (equations (1) and (2)) suggest approximation algorithms for evaluating the corresponding formulas over LTSs by Picard iteration. For instance, $[\![\exists\Diamond_c r]\!]^{\mathrm{f}} = \lim_{n\to\infty} v_n$, where $v_0(s) = q(s)$, and $v_{n+1}(s) = q(s) \sqcup \alpha \cdot \max\{v_n(s') \mid s' \in \delta(s)\}$ for all $n \geq 0$. Lemmas 4 and 10 show that these algorithms reach their fixpoints within $|S|$ steps, thus yielding exact algorithms rather than approximations. As stated in [9], this leads to algorithms for model checking $\exists\Diamond_c r$ and $\forall\Diamond_c r$ in time $O(|S| \cdot |\delta|)$. Here, we provide improved algorithms of complexity $O(|\delta| + |S| \log |S|)$.

The algorithms use a priority queue data structure, that provides the following functionalities.

- *insert*(Q, t, x): Inserts element t in the queue Q and it assigns priority x to t.
- *empty*(Q): Returns **true** if the queue Q is empty, and **false** otherwise.
- *extract_max*(Q): Returns a pair (t, x), where t is an element with the highest priority in Q and x is its priority, and it removes t from the queue.
- *increase_key*(Q, t, x): If t belongs to the queue Q and its current priority is smaller than x, then increases its priority to x, otherwise it leaves Q unchanged.

By using heaps, we can implement these procedures such that computing the function *empty* takes time $O(1)$, while the other functions require time $O(\log n)$, where n is the number of elements in the queue, see for instance [5].

3.1.1 Model checking $\exists\Diamond$. Informally, the idea behind the improved algorithm for $\exists\Diamond r$ is as follows. First, we set $u(s) = q(s)$ for all states s, and we mark the states "not-done". Then, we iteratively pick the not-done state s having the largest value of $u(s)$, and we mark it "done"; we also propagate to all its predecessors t the value $u(t) := u(t) \sqcup \alpha \cdot u(s)$. We show that when a state is marked "done", it holds $u(s) = [\![\exists\Diamond_c r]\!](s)$. This algorithm can be implemented using a priority queue that contains all the "not-done" states, as follows.

Algorithm 1
function ExistsDiamond$(\mathcal{S}, \alpha, q)$
vars:
    val : state array of rationals
    Q : priority queue
init:
    for each $t \in S$ **do**
        insert$(Q, t, q(t))$
    done
main:
    while not *empty*(Q) **do**
        $(t, val[t])$:= *extract_max*(Q)
        for each s such that $t \in \delta(s)$ **do**
            increase_key$(Q, s, \alpha \cdot val[t])$
        done

done
return *val*

To show the correctness of the algorithm, we first observe that the value $[\![\diamond_c r]\!](\rho)$ is attained at the first occurrence of a certain state in the path ρ and that the value $[\![\exists\diamond_c r]\!](s)$ is attained on an acyclic, finite path.

Lemma 3 *Given a path $\rho = s_0 s_1 \ldots \in Traj(s)$, there exists $k \geq 0$ such that:* (i) *for all $0 \leq i < k$: $s_i \neq s_k$, and* (ii) $[\![\diamond_c r]\!](\rho) = \alpha^k q(s_k)$.

Proof. Consider a path $\rho = s_0 s_1 \ldots$ and let $X = \{\alpha^i \cdot q(s_i) \mid i \geq 0\}$ and $u = \sup X = [\![\diamond_c r]\!](\rho)$. Consider two indices $0 < j < k$, such that $s_j = s_k$. Then, $\alpha^k \cdot q(s_k) = \alpha^k \cdot q(s_j) < \alpha^j \cdot q(s_j) < u$. In words, the discounted values of r at state s_k is smaller than every previous occurrence of the same state. $\square$

Lemma 4 *Given a state $s \in S$, there is a finite, acyclic path $s_0 s_1 \ldots s_k \in FTraj(s)$ such that:*

$$[\![\exists\diamond_c r]\!](s) = \alpha^k q(s_k).$$

Proof. From Lemma 3. $\square$

Lemma 5 *During the execution of Algorithm 1, it is always true that, if state s does not belong to the queue, then $val[s]$ is greater than or equal to the maximum priority in the queue.*

Proof. At the beginning, the property is trivially true. If the property is true at some moment, and *extract_max* is called, the property remains true. Moreover, if the property is true at some moment, and *increase_key*$(Q, t, \cdot)$ is called, the property also remains true. This follows from the fact that the priority of t can only be increased to $\alpha \cdot x$, where x is the priority of a state which has just been removed from the queue. $\square$

The following results use the abbreviation $v(s)$ for $[\![\exists\diamond_c r]\!](s)$.

Lemma 6 *During the execution of Algorithm 1, a state s is never assigned a priority greater than $v(s)$.*

Proof. The property holds after initialization because then $val[s] = q(s) \leq v(s)$. Assume that the property is true and that the function *increase_key* is called. Then there exists a pair $(t, val[t])$ with $t \in \delta(s)$ and which was just removed from the queue. Applying the definition of v and the assumption yields $v(s) \geq \alpha \cdot v(t) \geq \alpha \cdot val[t] = val[s]$. All other statements trivially preserve this property. $\square$

Lemma 7 *During the execution of Algorithm 1, when a state s is extracted from the queue, $val[s] = v(s)$.*

Proof. For all states $s \in S$, let

$$l_s = \min\{|\rho| \mid \rho \text{ is a finite, acyclic path between } s \text{ and } t \text{ with } v(s) = \alpha^{|\rho|}q(t)\}.$$

Note that Lemma 4 ensures that the above minimum is never taken over an empty set. We prove our statement by induction on l_s.

If $l_s = 0$, we have $v(s) = q(s)$. Then, Lemma 6 guarantees that the priority of s is never increased from its initial value of $q(s)$, and we obtain the result.

If $l_s > 0$, then we have $v(s) = \max_{t \in \delta(s)} \alpha v(t)$, say $v(s) = \alpha v(t_0)$. Moreover, $l_s > l_{t_0}$. Now, consider the moment where t_0 was extracted from the queue. By induction hypothesis, we have $val[t_0] = v(t_0)$. Lemma 6 yields that $val[s] \leq v(s) = \alpha v(t_0) = \alpha val[t_0]$. Then, in particular, Lemma 5 implies that s was still in the queue when t_0 was extracted. Since $t_0 \in \delta(s)$, the function *increase_key* is called and, since $val[s] \leq \alpha v(t_0)$, this call sets $val[s]$ to its final value $\alpha v(t_0)$. $\square$

The following is a direct consequence of Lemma 7.

Lemma 8 [Correctness] *Let $val =$ **ExistsDiamond**$(\mathcal{S}, \alpha, q)$. For all $s \in S$, $val[s] = [\![\exists\Diamond_c r]\!](s)$.*

Lemma 9 [Complexity] *Algorithm 1 runs in time $O(|\delta| + |S| \log |S|)$.*

Proof. The initialization phase alone takes time $O(|S| \log |S|)$. In each iteration of the main loop, a state t is extracted from the queue and *increase_key* is called on all predecessors of t. Thus, *increase_key* may be called several times on each state. However, the priority of a state s can only be increased once. To see this, assume that at some point the priority of s is increased to the value $\alpha val[t]$. It holds that all the states that are still in the queue have priority at most $val[t]$. Therefore, the priority of s cannot be further increased. Considering that *increase_key*$(\cdot, s, \cdot)$ runs in constant time unless it actually increases the value of s, the complexity of the main loop reduces to examining every edge in the LTS (time $O(|\delta|)$), plus increasing the value of each state at most once (time $O(|S| \log |S|)$). $\square$

Notice that if we want to compute $[\![\exists\Diamond_c r]\!]$ on a fixed state s, we can achieve a smaller complexity by exploiting the equation

$$[\![\exists\Diamond_c r]\!](s) = \max\{\alpha^{sp(s,t)} \cdot q(t) \mid t \in S\},$$

where $sp(s,t)$ is the length of an (unweighted) shortest path from s to t. The values $sp(s,t)$ can be computed by a breadth first search over the LTS, yielding time complexity $O(|S| + |\delta|)$ for the above algorithm.

3.1.2 Model checking $\forall\Diamond$. The algorithm for $\forall\Diamond$ is similar to the algorithm for $\exists\Diamond$, except that the valuation of a state is increased when *all* of its successors are marked "done", rather than each time a successor is marked "done". Again, the algorithm can be implemented using a priority queue, as follows.

Algorithm 2
function ForallDiamond$(\mathcal{S}, \alpha, q)$
vars:
    val : state array of rationals
    $count$: state array of integers
    Q : priority queue
init:
    for each $t \in S$ **do**
        $count[t] := |\delta(t)|$
        $insert(Q, t, q(t))$
    done
main:
    while not $empty(Q)$ **do**
        $(t, val[t]) := extract\_max(Q)$
        for each s such that $t \in \delta(s)$ **do**
            $count[s] := count[s] - 1$
            if $count[s] = 0$ **then** $increase\_key(Q, s, \alpha \cdot val[t])$
        done
    done
    return val

When proving the correctness of Algorithm 2, we use the short notation $v(s)$ for $[\![\forall\Diamond_c r]\!](s)$. Moreover, we denote by $prio(s)$ the priority of s, when s is a state belonging to the queue.

For all $s_0 \in S$, we define $STraj(s_0)$ (S stands for "simple") to be the set of all finite paths $\rho = s_0 \dots s_n$ such that: *(i)* ρ is acyclic (no state repetitions), and *(ii)* it can be extended in one step to a cyclic path, i.e. there is $i \leq n$ s.t. $s_i \in \delta(s_n)$. Notice that $STraj(s_0)$ is finite and every path in $STraj(s_0)$ contains at most $|S|-1$ steps. In the statement of the following lemma, we assume that the semantics of $\Diamond_c r$ is extended to finite paths in the obvious way.

Lemma 10 *In order to compute the value of* $[\![\forall\Diamond_c r]\!](s)$*, only paths in* $STraj(s)$ *need to be considered. Formally,*

$$[\![\forall\Diamond_c r]\!](s) = \min_{\rho\in STraj(s)} [\![\Diamond_c r]\!](\rho).$$

Moreover, there is a path $\rho \in Traj(s)$ *such that* $[\![\forall\Diamond_c r]\!](s) = [\![\Diamond_c r]\!](\rho)$*.*

Proof. We first prove that $\inf_{\rho\in Traj(s)}[\![\Diamond_c r]\!](\rho) \geq \min_{\rho\in STraj(s)}[\![\Diamond_c r]\!](\rho)$. Let $\rho^* \in STraj(s)$ be any path such that $[\![\Diamond_c r]\!](\rho^*) = \min_{\rho\in STraj(s)}[\![\Diamond_c r]\!](\rho)$. If we prove that every path $\rho \in Traj(s)$ gives a value for $\Diamond_c r$ greater than the one of ρ^*, we are done. Take any path $\rho = s_0 s_1 \dots$ in $Traj(s)$. Let ρ' be the longest prefix of ρ which is an acyclic path. Clearly, $\rho' \in STraj$. Since ρ' is a prefix of ρ, it holds that $[\![\Diamond_c r]\!](\rho) \geq [\![\Diamond_c r]\!](\rho') \geq [\![\Diamond_c r]\!](\rho^*)$.
Conversely, we prove that $\inf_{\rho\in Traj(s)}[\![\Diamond_c r]\!](\rho) \leq \min_{\rho\in STraj(s)}[\![\Diamond_c r]\!](\rho)$. We do this by showing that every element in $STraj(s)$ has a corresponding element in $Traj(s)$ which assigns the same value to $\Diamond_c r$, thus also proving the second statement of the Lemma.
Let $\rho = s_0 \dots s_n$ be an element of $STraj(s)$ and let ρs_{n+1} be an extension of ρ which is a cyclic path. Formally, $s_{n+1} = s_j$ for some $0 \leq j \leq n$. Let ρ' be the infinite path obtained by repeating forever the loop in ρs_{n+1}, i.e. $\rho' = s_0 \dots s_{j-1}(s_j \dots s_n)^\omega$. By Lemma 3, the value $[\![\Diamond_c r]\!](\rho') = \sup_{i\geq 0} \alpha^i \cdot [\![r]\!](\rho'(i))$ is attained at the first occurrence of some state. By construction of ρ' this state must occur in the first $n+1$ states of ρ', which are the original states of ρ. Therefore, $[\![\Diamond_c r]\!](\rho') = \sup_{0\leq i\leq n} \alpha^i \cdot [\![r]\!](s_i) = [\![\Diamond_c r]\!](\rho)$. $\square$

Lemma 11 *During the execution of Algorithm 2, it is always true that, if state* s *does not belong to the queue, then* $val[s]$ *is greater than or equal to the maximum priority in the queue.*

Proof. As for Lemma 5. $\square$

Lemma 12 *During the execution of Algorithm 2, a state* s *is never assigned a priority greater than* $v(s)$*.*

Proof. By contradiction, let s be the first state whose priority is modified to a value greater than $v(s)$, by means of a call to $increase\_key(Q, s, \alpha \cdot val[t])$. Notice that this can only happen if $count[s] = 1$ when t is extracted from the queue.
Since s is the first such node, the priority of t was never set to a value greater than $v(t)$. Thus, after the above call to *increase_key*, we have $v(s) < prio(s) = \alpha \cdot val[t] \leq \alpha \cdot v(t)$. Considering Lemma 10, let $\rho \in Traj(s)$ be a path such that $[\![\Diamond_c r]\!](\rho) = v(s)$. It must be $[\![\Diamond_c r]\!](\rho) \leq \alpha \cdot v(t)$. Let $\rho = (s\, s_1 s_2 \dots)$, we claim that the state s_1 is still in the queue when t is extracted, thus contradicting the assumption that $count[s] = 1$ when t is extracted.

If s_1 was extracted before t, by Lemma 11 we get $val[s_1] \geq val[t]$. Therefore, we get from the initial assumption that $v(s) < \alpha \cdot val[t] \leq \alpha \cdot val[s_1] \leq \alpha \cdot v(s_1) \leq \alpha \cdot [\![\Diamond_c r]\!](s_1 s_2 \ldots)$, and at the same time $v(s) = q(s) \sqcup \alpha \cdot [\![\Diamond_c r]\!](s_1 s_2 \ldots) \geq \alpha \cdot [\![\Diamond_c r]\!](s_1 s_2 \ldots)$, which is a contradiction. $\square$

Lemma 13 *During the execution of Algorithm 2, when a state s is extracted from the queue, $val[s] = v(s)$.*

Proof. For all states s, let ACF_s be the set of acyclic (and thus finite) paths ρ starting at s that satisfy:

- if t is the last state of ρ, then $v(t) = q(t)$, and
- if t is a state of ρ that is not the last, then $v(t) > q(t)$.

Note that this set is non-empty. To see this, let t_* be the state with highest v-value among those reachable from s; then, ACF_s must contain a prefix of each acyclic path from s to t_*. Let also $l_s = \max\{|\rho| \mid \rho \in ACF_s\}$. We prove our statement by induction on l_s.
If $l_s = 0$, we have $v(s) = q(s)$. Then, Lemma 12 guarantees that the priority of s is never increased from its initial value of $q(s)$, and we obtain the result.
If $l_s > 0$, we have $v(s) > q(s)$. Then, for all states $t \in \delta(s)$, we have $l_t \leq l_s - 1$ and $v(t) \geq \frac{v(s)}{\alpha} > v(s)$. By inductive hypothesis, when t is extracted, $val[t] = v(t)$. Now, if s is extracted before t, *count*$[s]$ never reached 0 and thus the priority of s was never modified from its initial value $q(s)$. Thus, Lemma 11 guarantees that, after s is extracted, all elements still in the queue have priority at most $q(s)$. We then obtain that, when t is finally extracted, $val[t] \leq q(s) < v(s) \leq v(t)$, which contradicts the inductive hypothesis. This proves that all successors of s are extracted before s itself. Notice that $v(s) = \alpha \min_{t \in \delta(s)} v(t)$. Then, when the last successor of s leaves the queue, it assigns the correct value $v(s)$ to the priority of s. $\square$

The following is a direct consequence of Lemma 13.

Lemma 14 [Correctness] *Let val = **ForallDiamond**$(\mathcal{S}, \alpha, q)$. For all $s \in S$, $val[s] = [\![\forall \Diamond_c r]\!](s)$.*

Lemma 15 [Complexity] *Algorithm 2 runs in time $O(|\delta| + |S| \log |S|)$.*

Proof. The initialization phase requires time $O(|S| \log |S|)$. In each iteration of the main loop, a different state is extracted from the queue and its incoming edges are considered. An optional call to *increase_key* is made. In total, every edge in the LTS is considered once (time $O(|\delta|)$) and *increase_key* is called at most once for every state (time $O(|S| \log |S|)$). $\square$

3.1.3 Model checking $\forall\triangle$. Computing $[\![\forall\triangle_c r]\!](s)$ consists in minimizing the (discounted) average $[\![\triangle_c r]\!]$ over the paths from s. As observed by [23] for the non-discounted case, the minimal discounted average is obtained on a path ρ' from s which, after some prefix ρ keeps repeating some simple cycle ℓ. Hence ℓ contains at most $|S|$ states. To find ρ', we use two steps. In the first phase, we find for each state s the simple cycle ℓ starting at s with the minimal discounted average. In the second phase, we find the best prefix-cycle combination $\rho\ell^\omega$.

Phase 1. We need to compute

$$L_\alpha(s) = \min\{[\![\triangle_c r]\!](\rho) \mid \rho \in \mathit{Traj}_s \text{ and } \rho = (s_0 s_1 s_2 \dots s_{n-1})^\omega \text{ and } n \leq |S|\},$$

where the value $[\![\triangle_c r]\!](\rho)$ is given by $\frac{1-\alpha}{1-\alpha^n} \cdot \sum_{i=0}^{n-1} \alpha^i \cdot q(s_i)$. Consider for $n \geq 0$ the recursion

$$v_0(s, s') = 0, \qquad v_{n+1}(s, s') = q(s) + \alpha \cdot \min\{v_n(t, s') \mid t \in \delta(s)\}.$$

Then $v_n(s, s')$ minimizes $\sum_{i=0}^{n-1} \alpha^i \cdot q(s_i)$ over all finite paths $s_0 s_1 \dots s_n$ with $s_0 = s$ and $s_n = s'$. Hence

$$L_\alpha(s) = (1-\alpha) \cdot \min\left\{\tfrac{v_1(s,s)}{1-\alpha^1}, \tfrac{v_2(s,s)}{1-\alpha^2}, \dots, \tfrac{v_{|S|-1}(s,s)}{1-\alpha^{|S|-1}}\right\}.$$

For a fixed state s', computing $\min\{v_n(t, s') \mid t \in \delta(s)\}$ for all $s \in S$ can be done in $O(|\delta|)$ time. Therefore, v_{n+1} is obtained from v_n in $O(|S|^2 + |S| \cdot |\delta|) = O(|S| \cdot |\delta|)$ time. Hence, the computation of $v_{|S|-1}$ and L_α requires $O(|S|^2 \cdot |\delta|)$ time. A possible implementation of this phase is sketched in Algorithm 3, where it holds that $L_\alpha = \mathbf{LoopCost}(\mathcal{S}, \alpha, q)$. To make the complexity of the algorithm more explicit, the transition function δ is treated as a relation $\delta \subseteq S \times S$.

Phase 2. After a prefix of length n, the cost $L_\alpha(s)$ of repeating a cycle at state s has to be discounted by α^n, which is exactly the factor by which we discount $q(s)$ after taking that prefix. Hence, we modify the original LTS $\mathcal{S}$ into an LTS $\mathcal{S}^+$, as follows. For every state $s \in S$, we add a copy $\hat{s}$ whose weight $w^+(\hat{s})$ we set to $L_\alpha(s)$; the weights $w^+(s)$ of states $s \in S$ remain $q(s)$. Moreover, for every $t \in S$ and $s \in \delta(t)$, we add $\hat{s}$ as a successor to t, that is, $\delta^+(t) = \delta(t) \cup \{\hat{s} \mid s \in \delta(t)\}$ and $\delta^+(\hat{s}) = \{\hat{s}\}$. Taking the transition from t to $\hat{s}$ corresponds to moving to s and repeating the optimal cycle from there. We find the value of the optimal prefix-cycle combination starting from s as the *discounted distance* from s to $\hat{S} = \{\hat{s} \mid s \in S\}$ in the modified graph $\mathcal{S}^+$ with weights w^+. Formally, given an LTS $\mathcal{S}$, a state s, a weight function $w\colon S \to \mathbb{R}^{\geq 0}$, a discount factor α, and a target set T, the minimal discounted distance from s to T is $d(s) = \min\{\sum_{i=0}^{n-1} \alpha^i \cdot w(s_i) \mid s_0 s_1 \dots s_{n-1} \in \mathit{FTraj}(s) \text{ and } s_{n-1} \in T\}$. The value of $d(s)$ for $s \in S$ is computed by the call $\mathbf{DiscountedDistance}(\mathcal{S}^+, w^+, \alpha, \hat{S})$ to the Algorithm 4, which is a discounted version of the Bellman-Ford algorithm for

Algorithm 3
function LoopCost$(\mathcal{S}, \alpha, q)$
vars:
    v_i, for $i \in \{0, \ldots, |S|-1\}$: (state * state) array of rationals
    L_α : state array of rationals
init:
    for each $s, t \in S$ **do**
        $v_0[s, t] := 0$
    done
    for each $s, t \in S$ and $i \in \{1, \ldots, |S|-1\}$ **do**
        $v_i[s, t] := \infty$
    done
main:
    for $i := 1$ **to** $|S| - 1$ **do**
        for each edge $(s, t) \in \delta$ **do**
            for each $s' \in S$ **do**
                if $v_i[s, s'] > v_{i-1}[t, s']$ **then** $v_i[s, s'] := v_{i-1}[t, s']$
            done
        done
    done
    for each $s, s' \in S$ **do**
        $v_i[s, s'] := q(s) + \alpha v_i[s, s']$
    done
    for each $s \in S$ **do**
        $L_\alpha[s] := (1-\alpha) \min\{\frac{v_1[s,s]}{1-\alpha}, \frac{v_2[s,s]}{1-\alpha^2}, \ldots, \frac{v_{|S|-1}[s,s]}{1-\alpha^{|S|-1}}\}$
    done
    return L_α

Algorithm 4
function DiscountedDistance$(\mathcal{S}, w, \alpha, T)$
vars:
    d : state array of rationals
init:
    for each $t \in S$ **do**
        if $t \in T$ **then** $d[t] := w(t)$ **else** $d[t] := \infty$
    done
main:
    for $i := 1$ **to** $|S| - 1$ **do**
        for each $s \in S$ and $s' \in \delta(s)$ **do**
            if $d[s] > w(s) + \alpha \cdot d[s']$ **then** $d[s] := w(s) + \alpha \cdot d[s']$
        done
    done
    return d

finding shortest paths. Our algorithm performs backward computation from the set T, because discounted shortest paths (i.e., paths whose discounted distance is minimal among all paths with the same first and last state) are closed under suffixes, but not under prefixes.

Like the standard version, discounted Bellman-Ford runs in $O(|S|\cdot|\delta|)$ time. Thus, the complexity of computing $[\![\forall\triangle_c r]\!]$ is dominated by the first phase.

Lemma 16 [Correctness] *Let* $d = \textbf{DiscountedDistance}(\mathcal{S}^+, w^+, \alpha, \hat{S})$*, where* $\mathcal{S}^+, w^+$ *and* $\hat{S}$ *are defined in the previous section. For all* $s \in S$*,* $d[s] = [\![\forall\triangle_c r]\!](s)$*.*

Lemma 17 [Complexity] *The value* $[\![\forall\triangle_c r]\!]$ *can be computed in time* $O(|S|^2 \cdot |\delta|)$*.*

3.1.4 Complexity of DCTL model checking over LTSs. The overall complexity of model checking a DCTL formula is polynomial in the size of the system and the size of the formula.

Theorem 5 *Consider a* DCTL *formula* ϕ*, an LTS* $\mathcal{S} = (S, \delta, P, [\cdot])$*, and a parameter interpretation* $\langle\cdot\rangle$*. The following assertions hold:*

(1) The problem of model checking ϕ *over* $\mathcal{S}$ *with respect to* $\langle\cdot\rangle$ *can be solved in time* $O(|S|^2 \cdot |\delta| \cdot |\phi|)$*.*
(2) If ϕ *does not contain the* $\triangle$ *operator, then the problem of model checking* ϕ *over* $\mathcal{S}$ *with respect to* $\langle\cdot\rangle$ *can be solved in time* $O((|\delta| + |S| \log |S|) \cdot |\phi|)$*.*

3.2 Model Checking DCTL over Markov Chains

As stated by Theorem 2, the path and fixpoint semantics over Markov chains coincide for the formula $\exists\triangle_c r$. Hence, in Section 3.2.4 we present an algorithm for model checking this formula over Markov chains in either semantics. By contrast, the path and the fixpoint semantics over Markov chains may differ for the formulas $\exists\Diamond_c r$ and $\forall\Diamond_c r$. Hence, we need to provide algorithms for both semantics. Because of the absence of nondeterministic choice in a Markov chain, $[\![\exists\Diamond_c r]\!]^* = [\![\forall\Diamond_c r]\!]^*$ for $* \in \{\mathrm{f}, \mathrm{p}\}$; so giving algorithms for $\exists\Diamond_c$ suffices. Section 3.2.1 gives the algorithm for model checking $\exists\Diamond_c r$ over a Markov chain with respect to the path semantics; Section 3.2.3 treats the formula $\exists\Diamond_c r$ in the fixpoint semantics. In the following, we consider a fixed Markov chain $(S, \tau, \Sigma, [\cdot])$ and its probability transition matrix P. We write I for the identity matrix.

3.2.1 Model checking $\exists\Diamond$ in the path semantics. When evaluating $[\![\exists\Diamond_c r]\!]^{\mathrm{p}}$ in a state s, we start with the initial estimate $q(s)$. If s is the state

$s_{\max}$ with the maximum value of q, the initial estimate is the correct value. If s has the second largest value for q, the estimate can only be improved if $s_{\max}$ is hit within a certain number l of steps, namely, before the discount α^l becomes smaller than $\frac{q(s)}{q(s_{\max})}$. This argument is recursively applied to all states.

Let $s_1, \ldots, s_n$ be an ordering of the states in S such that $q(s_1) \geq q(s_2) \geq \cdots \geq q(s_n)$. We use integers as matrix indices, thus writing $P(i,j)$ for p_{s_i,s_j}. For all $1 \leq j < i \leq n$, let

$$k_{i,j} = \begin{cases} \lfloor \log_\alpha \frac{q(s_i)}{q(s_j)} \rfloor & \text{if } q(s_i) > 0 \\ 0 & \text{if } q(s_i) = 0 \text{ and } q(s_j) = 0 \\ \infty & \text{otherwise} \end{cases}$$

Let $v(s_i) = [\![\exists \Diamond_\alpha r]\!]^{\mathrm{p}}(s_i)$. Then, $v(s_1) = q(s_1)$, and we can express the value of $v(s_i)$ in terms of the values $v(s_1), \ldots, v(s_{i-1})$. Let $K = \max\{k_{i,j} \mid k_{i,j} < \infty\}$, and for all $l > 0$, let $B^i_l = \{s_j \mid 1 \leq j < i \text{ and } 1 \leq l \leq k_{i,j}\}$. Intuitively, B^i_l contains those states that, if hit in exactly l steps from s_i, can increase the value of $v(s_i)$.

For the (arbitrary) state s_i, the following holds:

$$v(s_i) = q(s_i) \cdot \mathit{stay}^i + \sum_{j=1}^{i-1} v(s_j) \cdot \sum_{l=1}^{k_{i,j}} \alpha^l \cdot \mathit{go}^i_{j,l}, \tag{9}$$

where $\mathit{stay}^i = \Pr_{s_i}\left[\bigwedge_{l>0} Z_l \notin B^i_l\right]$, $\mathit{go}^i_{j,l} = \Pr_{s_i}\left[Z_l = s_j \wedge \bigwedge_{m=1}^{l-1} Z_m \notin B^i_m\right]$, and the random variable Z_l was defined in Section 2.3 as the state of the markov chain after l steps. It is easy to check that $\mathit{stay}^i + \sum_{j=1}^{i-1} \sum_{l=1}^{k_{i,j}} \mathit{go}^i_{j,l} = 1$. We proceed in two phases. The first phase handles states s_i with $q(s_i) > 0$. Since the sequence $(B^i_l)_{l>0}$ is decreasing, it can have at most $|S|$ different values. It follows that there exist $m \leq |S|$ and $b^i_1 \leq \cdots \leq b^i_{m+1} \in \mathbb{N}$ and sets $X^i_1, \ldots, X^i_m \subseteq S$ such that $b^i_1 = 1$, $b^i_{m+1} = K + 1$, and for all $k = 1, \ldots, m$ and all $b^i_k \leq l < b^i_{k+1}$, we have $B^i_l = X^i_k$. Let P^i_k be the substochastic matrix obtained from P by disabling all transitions leading to states in X^i_k, i.e., $P^i_k(j', j) = 0$ for all j', j with $s_j \in X^i_k$. Then, for given $b^i_k \leq l < b^i_{k+1}$, we have

$$\mathit{go}^i_{j,l} = \left((P^i_1)^{b^i_2 - b^i_1} \cdot (P^i_2)^{b^i_3 - b^i_2} \cdot \ldots \cdot (P^i_{k-1})^{b^i_k - b^i_{k-1}} \cdot (P^i_k)^{l - b^i_k} \cdot P \right)(i,j).$$

Let $m^i_j = \max\{k \mid s_j \in X^i_k\}$ be the index of the last X^i_k containing s_j. We have

$$\sum_{l=1}^{k_{i,j}} \alpha^l \cdot go^i_{j,l} = \sum_{k=1}^{m^i_j} \sum_{l=b^i_k}^{b^i_{k+1}-1} \alpha^l \cdot go^i_{j,l} =$$

$$\Big(\sum_{k=1}^{m^i_j} \alpha^{b^i_k} \cdot (P^i_1)^{b^i_2-b^i_1} \cdot (P^i_2)^{b^i_3-b^i_2} \cdot \ldots \cdot (P^i_{k-1})^{b^i_k-b^i_{k-1}} \cdot \Big(\sum_{l=0}^{b^i_{k+1}-b^i_k-1} \alpha^l \cdot (P^i_k)^l \cdot P \Big)\Big)(i,j) =$$

$$\Big(\sum_{k=1}^{m^i_j} \alpha^{b^i_k} \cdot (P^i_1)^{b^i_2-b^i_1} \cdot (P^i_2)^{b^i_3-b^i_2} \cdot \ldots \cdot (P^i_{k-1})^{b^i_k-b^i_{k-1}} \cdot \Big(\frac{I-(\alpha P^i_k)^{b^i_{k+1}-b^i_k}}{I-\alpha P^i_k}\Big) \cdot P\Big)(i,j).$$

Each matrix $(P^i_k)^{b^i_{k+1}-b^i_k}$ can be computed by repeated squaring in time $O(|S|^3 \cdot \log b^i_k)$. Some further calculations show that, for a fixed i, both $\sum_{l=1}^{k_{i,j}} \alpha^l \cdot go^i_{j,l}$ and $\sum_{l=1}^{k_{i,j}} go^i_{j,l}$ can be computed in time $O(|S|^4 \cdot \log K)$. The value $stay^i$ is given by $1 - \sum_{j,l} go^i_{j,l}$. The total complexity of this phase is thus $O(|S|^5 \cdot \log K)$.

The second phase considers those states s_i with $q(s_i) = 0$. Let u be the smallest index i such that $q(s_i) = 0$. Now, $go^i_{j,l}$ is the probability of hitting s_j after exactly l steps, meanwhile avoiding all states with indices smaller than u. To compute $v(s_i)$ efficiently, we define a stochastic matrix P_0 from P by adding an absorbing state s_{n+1} and using s_{n+1} to turn all states s_j with $j < u$ into transient states (so, for all $j < u$, $P_0(j, n+1) = 1$ and $P_0(j, j') = 0$ for $j' \neq n+1$). Also, we set $\bar{v}$ to be the column vector with $\bar{v}_j = v(s_j)$ (computed in phase 1), if $j < u$, and $\bar{v}_j = 0$ otherwise. Then,

$$v(s_i) = \sum_{j=1}^{u-1} v(s_j) \cdot \sum_{l=1}^{\infty} \alpha^l \cdot (P_0)^l(i,j) = ((I - \alpha P_0)^{-1} \cdot \bar{v})(i). \qquad (10)$$

Solving the system (10) takes time $O(|S|^3)$ using LUP decomposition. The time spent in the two phases amounts to $O(|S|^5 \cdot \log K)$, which is polynomial in the size of the input.

Lemma 18 *The value* $[\![\exists \Diamond_c r]\!]^{\mathrm{p}}$ *can be computed in time* $O(|S|^5 \cdot \log K)$.

3.2.2 Alternative algorithm for $\exists\Diamond$ in the path semantics. We can solve the system (9) using an alternative recursion. We obtain an algorithm that takes time $O(|S|^3 \cdot K)$. As K can be exponential in the number of bits used to encode the numerical constants in the system, this algorithm is only pseudo-polynomial. However, in principle this algorithm performs better than the previous one when $K < |S|^2 \cdot \log K$. We outline this solution below.

The main step in the algorithm presented in the previous section is to compute the values $go^i_{j,l}$ for states s_i for which $q(s_i) > 0$. For $l > 0$, let C^i_l be the event

"$Z_l \notin B_l^i$". It holds that

$$\begin{aligned}
go^i_{j,l} &= \Pr_{s_i}[Z_l = s_j \wedge C^i_1 \wedge \ldots \wedge C^i_{l-1}] \\
&= \Pr_{s_i}[Z_l = s_j \mid C^i_1 \wedge \ldots \wedge C^i_{l-1}] \cdot \Pr_{s_i}[C^i_{l-1} \mid C^i_1 \wedge \ldots \wedge C^i_{l-2}] \cdot \ldots \cdot \Pr_{s_i}[C^i_1].
\end{aligned}$$

For each $j = 1, \ldots, i-1$ and $l > 0$, let $p(s_j, l) = \Pr_{s_i}[Z_l = s_j \mid \bigwedge_{m=1}^{l-1} Z_m \notin B^i_m]$. In words, $p(s_j, l)$ is the probability that, starting in s_i, the system reaches s_j after exactly l steps, given that in each previous step it does not hit states that can influence $v(s_i)$. For all $j = 1, \ldots, n$ and $0 < l \leq K$, we can compute $\Pr_{s_i}[Z_l = s_j \mid C^i_1 \wedge \ldots \wedge C^i_{l-1}]$ together with $p(s_j, l)$ using the following recursion:

$$\begin{aligned}
p(s_j, 1) &= P(i, j) \\
\Pr_{s_i}[C^i_1] &= \sum\{p(s_t, l) \mid s_t \notin B^i_1\} \\
p(s_j, l+1) &= \frac{1}{\Pr_{s_i}[C^i_l \mid C^i_1 \wedge \ldots \wedge C^i_{l-1}]} \cdot \sum\{P(t, j) \cdot p(s_t, l) \mid s_t \notin B^i_l\} \\
\Pr_{s_i}[C^i_{l+1} \mid C^i_1 \wedge \ldots \wedge C^i_l] &= \sum\{p(s_t, l) \mid s_t \notin B^i_{l+1}\}
\end{aligned}$$

For a fixed i, the previous recursion takes time $O(|S|^2 \cdot K)$. Then,

$$go^i_{j,l} = p(s_j, l) \cdot \prod_{m=1}^{l-1} \Pr_{s_i}\left[C^i_m \mid C^i_1 \wedge \ldots \wedge C^i_{m-1}\right].$$

It follows that, for a fixed i, all values $go^i_{j,l}$ can be computed in time $O(|S|^2 \cdot K)$. The total complexity is thus $O(|S|^3 \cdot K)$. For states s_i such that $q(s_i) = 0$, we again solve the system (10) using LUP decomposition. Overall, this gives an algorithm that runs in $O(|S|^3 \cdot K)$.

Lemma 19 *The value $[\![\exists\Diamond_c r]\!]^{\mathrm{p}}$ can be computed in time $O(|S|^3 \cdot K)$.*

3.2.3 Model checking $\exists\Diamond$ in the fixpoint semantics. The value $[\![\exists\Diamond_c r]\!]^{\mathrm{f}}$ on a MC can be computed by transforming the fixpoint (1) into a linear-programming problem, following a standard approach. Expanding the definition of (1) for MCs, we have that $[\![\exists\Diamond_c r]\!]^{\mathrm{f}}$ is the unique fixpoint of the following equation in $v : S \mapsto \mathbb{R}$: for all $s \in S$,

$$v(s) = q(s) \sqcup \alpha \cdot \sum_{t \in S} v(t) \cdot p_{s,t}. \tag{11}$$

The following lemma enables us to compute this fixpoint via linear programming.

Lemma 20 *Consider the following linear-programming problem in the set $\{v(s) \mid s \in S\}$ of variables: minimize $\sum_{s \in S} v(s)$ subject to*

$$v(s) \geq q(s) \qquad v(s) \geq \alpha \cdot \sum_{t \in S} v(t) \cdot p_{s,t}$$

for all $s \in S$. Let $\hat{v} \in \mathcal{V}_S$ be an optimal solution, we have $\hat{v} = [\![\exists\Diamond_c r]\!]^{\mathrm{f}}$.

The above linear programming problem can be solved in time polynomial in $|\mathcal{S}|_b$ and $|\alpha|_b$.

3.2.4 Model checking $\forall\triangle$ in both semantics. Formulas of the type $\forall\triangle_c r$ can be evaluated by the following classical equation [12].

Lemma 21 *Let $[\![\exists\triangle_c r]\!]$ and q denote column vectors, we have*

$$[\![\exists\triangle_c r]\!] = (1-\alpha)\cdot\sum_{i\geq 0}\alpha^i P^i q = (1-\alpha)\cdot(I-\alpha P)^{-1}\cdot q.$$

Thus, we can compute the value $[\![\exists\triangle_c r]\!](s)$ for each state $s \in S$ by solving a linear system with $|S|$ variables. This takes time $O(|S|^{\log_2 7})$ using Strassen's algorithm or $O(|S|^3)$ using LUP decomposition.

3.2.5 Complexity of Dctl model checking over Markov chains. The overall complexity is polynomial in the size of the system, and exponential in the size of the formula. The latter exponential complexity is due to the fact that the number of arithmetic operations is polynomial in the size of the bit-wise encoding of the valuations, and these encodings grow exponentially with respect to the number of nestings of temporal operators.

Theorem 6 *Given a* Dctl *formula ϕ, a Markov chain $\mathcal{S} = (S, \tau, P, [\cdot])$, and a parameter interpretation $\langle\cdot\rangle$, the problem of model checking ϕ over $\mathcal{S}$ with respect to $\langle\cdot\rangle$ can be solved in time polynomial in $|S|$, $|[\cdot]|_b$, and $|\langle\cdot\rangle|_b$, and exponential in $|\phi|$.*

3.3 Model Checking Dctl over Markov Decision Processes

As it is the case for Markov chains, also for MDPs the path and fixpoint semantics do not coincide for the formulas $\exists\Diamond_c r$ and $\forall\Diamond_c r$, so that separate algorithms are needed. The two semantics do coincide for the formula $\forall\triangle_c r$ on MDPs, hence one algorithm suffices. We consider a fixed MDP $\mathcal{S} = (S, \tau, \Sigma, [\cdot])$.

3.3.1 Model checking $\exists\Diamond$ and $\forall\Diamond$ in the path semantics. If $\alpha = 0$, then trivially $[\![\exists\Diamond_c r]\!]^{\mathrm{p}}(s) = [\![\forall\Diamond_c r]\!]^{\mathrm{p}}(s) = q(s)$ at all $s \in S$, so in the following we assume $0 < \alpha < 1$. The problem of computing $[\![\exists\Diamond_c r]\!]^{\mathrm{p}}$ on an MDP can be viewed as an optimization problem, where the goal is to maximize the expected value of the sup of q over a path. As a preliminary step to solve the problem, we note that in general the optimal strategy is *history dependent,*

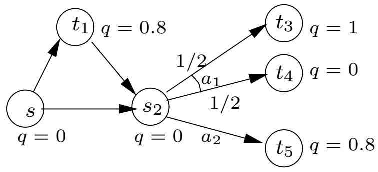

Fig. 3. An MDP requiring a memory strategy for $[\![\exists\Diamond_c r]\!]^{\mathrm{P}}(s)$.

that is, the choice of distribution at a state depends in general on the past sequence of states visited by the path.

Example 2 Consider the system depicted in Figure 3 and assume $\alpha = 1$. The optimal choice in state s_2 depends on whether t_1 was hit or not. If it was, the current sup is 0.8 and the best choice is a_1, because with probability $\frac{1}{2}$ the sup will increase to 1. If t_1 was not hit, the best choice is a_2, because it gives a certain gain of 0.8, rather than an expected gain of 0.5. The same argument holds if α is sufficiently close to 1.

While the above example indicates that the optimal strategy is in general history-dependent, it also suggests that all a strategy needs to remember is the maximum value that has occurred so far along the path. For $s \in S$ and $x \in \mathbb{R}$, we define

$$h^{\exists}(s,x) = \sup_{\pi\in\Pi} \mathrm{E}_s^{\pi}\Big[x \sqcup \sup_{i\geq 0} \alpha^i q(Z_i)\Big]$$

$$h^{\forall}(s,x) = \inf_{\pi\in\Pi} \mathrm{E}_s^{\pi}\Big[x \sqcup \sup_{i\geq 0} \alpha^i q(Z_i)\Big].$$

Obviously, we have $[\![\exists\Diamond_c r]\!]^{\mathrm{P}}(s) = h^{\exists}(s,0)$ and $[\![\forall\Diamond_c r]\!]^{\mathrm{P}}(s) = h^{\forall}(s,0)$. Note that the type of $h^{\exists}$ and $h^{\forall}$ is $S \times \mathbb{R} \mapsto \mathbb{R}$. To compute these quantities, we define two operators $H^{\exists}, H^{\forall} : (S \times \mathbb{R} \mapsto \mathbb{R}) \mapsto (S \times \mathbb{R} \mapsto \mathbb{R})$ as follows, for all $v : S \times \mathbb{R} \mapsto \mathbb{R}$, $s \in S$, and $x \in \mathbb{R}$:

$$H^{\exists}(v)(s,x) = \begin{cases} x & \text{if } x \geq 1 \\ \alpha \cdot \max\limits_{a\in\tau(s)} \sum\limits_{t\in S} v\Big(t, \dfrac{x \sqcup q(s)}{\alpha}\Big) \cdot a(t) & \text{otherwise} \end{cases} \tag{12}$$

$$H^{\forall}(v)(s,x) = \begin{cases} x & \text{if } x \geq 1 \\ \alpha \cdot \min\limits_{a\in\tau(s)} \sum\limits_{t\in S} v\Big(t, \dfrac{x \sqcup q(s)}{\alpha}\Big) \cdot a(t) & \text{otherwise} \end{cases} \tag{13}$$

Intuitively, the equation (12) can be understood as follows. At a state $s = s_m$ of a path $s_0 s_1 \ldots$, the quantity $v(s_m, x)$ represents the maximum over all strategies of $E_{s_m}[\sup_{i\geq 0} \alpha^i q(Z_i)]$ given that $\max_{0<i\leq m} \alpha^{-i} q(s_{m-i}) = x$. The recursion (12) then relates $v(s,x)$ to $v(t,y)$ at the successors t of s, where at t we consider the new conditioning $y = (x \sqcup q(s))/\alpha$, thus discounting $x \sqcup q(s)$ by α^{-1} (as s is one step before t). The following lemma states that $h^{\exists}$ and $h^{\forall}$ are the unique fixpoints of $H^{\exists}$ and $H^{\forall}$, respectively.

Lemma 22 *$h^\exists$ and $h^\forall$ are the unique fixpoints of $H^\exists$ and $H^\forall$.*

Proof. It is easy to see that the operators $H^\exists$ and $H^\forall$ admit a unique fixpoint, as they are α-contractions. We show that $h^\exists$ is a fixpoint of $H^\exists$; the case for $h^\forall$ and $H^\forall$ is analogous: we show thus that $H^\exists(h^\exists)(s,x) = h^\exists(s,x)$, for all $s \in S$ and $x \in \mathbb{R}$. First, note that for $x \geq 1$ we have $h^\exists(s,x) = x$, as the expectation of $\sup_{i\geq 0} \alpha^i q(Z_i)$ can be no larger than 1. For $x < 1$, we have:

$$\begin{aligned}
H^\exists(h^\exists)(s,x) &= \alpha \cdot \max_{a\in\tau(s)} \sum_{t\in S} a(t) \cdot h^\exists\Big(t, \frac{x \sqcup q(s)}{\alpha}\Big) \\
&= \alpha \cdot \max_{a\in\tau(s)} \sum_{t\in S} a(t) \cdot \sup_{\pi\in\Pi} \mathrm{E}_t^\pi\left[\frac{x \sqcup q(s)}{\alpha} \sqcup \sup_{i\geq 0} \alpha^i q(Z_i)\right] \\
&= \max_{a\in\tau(s)} \sum_{t\in S} a(t) \cdot \sup_{\pi\in\Pi} \mathrm{E}_t^\pi\left[x \sqcup q(s) \sqcup \sup_{i\geq 0} \alpha^{i+1} q(Z_i)\right] \\
&= \sup_{\pi\in\Pi} \mathrm{E}_s^\pi\left[x \sqcup \sup_{i\geq 0} \alpha^i q(Z_i)\right] = h^\exists(s,x).
\end{aligned}$$

$\square$

Since we are ultimately interested in the value of $h^\exists(s,0)$ for $s \in S$, and since if $x \geq 1$ we have $\mathrm{E}_s^\pi\left[x \sqcup \sup_{i\geq 0} \alpha^i q(Z_i)\right] = x$ for all $s \in S$ and $\pi \in \Pi$, it suffices to consider values for x that belong to the finite set

$$X = \{q(s)/\alpha^k \mid s \in S \wedge k \in \mathbb{N} \wedge q(s)/\alpha^k < 1\}.$$

To estimate the cardinality of X, consider any state s: if $q(s) \in \{0\} \cup [\alpha, 1)$, then s has only one representative in X, namely $q(s)$. If $q(s) = 1$ then s has no representative at all in X. Finally, if $q(s) \in (0, \alpha)$, s has k_s representatives $q(s), q(s)/\alpha, q(s)/\alpha^2, \ldots, q(s)/\alpha^{k_s-1}$, where $k_s = \lceil \log_\alpha q(s) \rceil$. Thus, let $Y = \{q(s) \mid s \in S \wedge q(s) \in (0,\alpha)\}$; if $Y = \emptyset$, $|X| \leq |S|$; otherwise, $|X| \leq |S| \cdot \lceil \log_\alpha(\min Y) \rceil$.

The fixpoints of $H^\exists$ and $H^\forall$ can be computed via linear programming, following a standard approach, enabling us to compute the path semantics of $\exists\Diamond$ and $\forall\Diamond$ in MDPs.

Lemma 23 *The following assertions hold:*

(1) Consider the following linear program in the set $\{v(s,x) \mid s \in S \wedge x \in X\}$ of variables: minimize $\sum_{s\in S} \sum_{x\in X} v(s,x)$ subject to

$$v(s,x) \geq \alpha \cdot \sum_{t\in S} \tilde{v}\Big(t, \frac{x \sqcup q(s)}{\alpha}\Big) \cdot a(t)$$

for all $s \in S$, all $x \in X$, and all $a \in \tau(s)$, where $\tilde{v}(t,x)$ is 1 if $x \geq 1$, and is $v(t,x)$ otherwise. Denoting by $\{\hat{v}(s,x) \mid s \in S \wedge x \in X\}$ an optimal solution, we have $[\![\exists\Diamond_c r]\!]^{\mathrm{p}}(s) = \hat{v}(s,q(s))$ for all $s \in S$.

(2) Consider the following linear program in the set $\{v(s,x) \mid s \in S \wedge x \in X\}$ of variables: maximize $\sum_{s\in S}\sum_{x\in X} v(s,x)$ subject to

$$v(s,x) \leq \alpha \cdot \sum_{t\in S} \tilde{v}\Big(t, \frac{x \sqcup q(s)}{\alpha}\Big) \cdot a(t)$$

for all $s \in S$, all $x \in X$, and all $a \in \tau(s)$, where $\tilde{v}(t,x)$ is 1 if $x \geq 1$, and is $v(t,x)$ otherwise. Denoting by $\{\hat{v}(s,x) \mid s \in S \wedge x \in X\}$ an optimal solution, we have $[\![\forall\Diamond_c r]\!]^{\mathrm{p}}(s) = \hat{v}(s,q(s))$ for all $s \in S$.

The linear programming problems in the above theorem contain at most $2\cdot|S|\cdot|X|$ variables. Hence, if q-values are encoded in binary notation, the number of variables in the encoding is linear in the size of the input encoding of the MDP.

3.3.2 Model checking $\exists\Diamond$ and $\forall\Diamond$ in the fixpoint semantics. The computation of $[\![\exists\Diamond_c r]\!]^{\mathrm{f}}$ on an MDP can be performed by transforming the fixpoint (1) into a linear-programming problem, following a standard approach. Expanding the definition of (1), we have that $[\![\exists\Diamond_c r]\!]^{\mathrm{f}}$ is the unique fixpoint of the following equation in $v \in \mathcal{V}_S$: for all $s \in S$,

$$v(s) = q(s) \sqcup \alpha \cdot \max_{a\in\tau(s)} \sum_{t\in S} v(t) \cdot a(t). \tag{14}$$

The following theorem enables us to compute this fixpoint via linear programming.

Lemma 24 *Consider the following linear-programming problem in the set $\{v(s) \mid s \in S\}$ of variables: minimize $\sum_{s\in S} v(s)$ subject to*

$$v(s) \geq q(s) \qquad v(s) \geq \alpha \cdot \sum_{t\in S} v(t) \cdot a(t)$$

for all $s \in S$ and all $a \in \tau(s)$. Denoting by $\{\hat{v}(s) \mid s \in S\}$ an optimal solution, we have $[\![\exists\Diamond_c r]\!]^{\mathrm{f}} = \hat{v}$.

The above reduction to linear programming yields an algorithm for $[\![\exists\Diamond_c r]\!]^{\mathrm{f}}$ that requires time polynomial in $|\mathcal{S}|_b$ and $|\alpha|_b$. The computation of $[\![\forall\Diamond_c r]\!]^{\mathrm{f}}$, on the other hand, is not known to be reducible in this fashion to linear programming, and as a consequence, we are only able to provide an algorithm that is in nondeterministic polynomial time in $|\mathcal{S}|_b$ and $|\alpha|_b$.

Define the two operators $L_\gamma, \hat{L} : \mathcal{V}_S \mapsto \mathcal{V}_S$, where $\gamma \in \Gamma$, as follows, for all $v \in \mathcal{V}_S$ and $s \in S$:

$$L_\gamma(v)(s) = q(s) \sqcup \alpha \cdot \sum_{t \in S} v(t) \cdot \gamma(s)(t)$$
$$\hat{L}(v)(s) = q(s) \sqcup \alpha \cdot \min_{a \in \tau(s)} \sum_{t \in S} v(t) \cdot a(t).$$

Comparing the definition of $\hat{L}$ with (2), we have that $[\![\forall \Diamond_c r]\!]^{\mathrm{f}} = \mu v. \hat{L} v$. Unfortunately, while (14) consisted only of max-operators, the operator $\hat{L}$ contains a mixture of max and min, and it is not known how to reduce its computation to the solution of a single linear programming problem.

The fixpoint of $\hat{L}$ can be computed using a standard policy-improvement scheme [2]. A *policy* is a mapping $\gamma : S \mapsto \mathrm{Distr}(S)$ such that $\gamma(s) \in \tau(s)$ for all $s \in S$; we denote by Γ the set of all policies. For a fixed policy γ, the operator L_γ involves only max, and its fixpoint can be computed by linear programming.

Lemma 25 *For $\gamma \in \Gamma$, the fixpoint $\mu v. L_\gamma v$ coincides with the optimal solution of the linear programming problem in $v \in \mathcal{V}_S$ that asks to minimize $\sum_{s \in S} v(s)$ subject to $v(s) \geq q(s)$ and $v(s) \geq \alpha \cdot \sum_{t \in S} v(t) \cdot \gamma_k(s)(t)$ for all $s \in S$.*

For $\gamma \in \Gamma$, we denote the fixpoint of L_γ by $v_\gamma = \mu v. L_\gamma v$. To obtain a policy iteration scheme, we define the *policy improvement* operator $H : \Gamma \mapsto \Gamma$ as follows, for all $\gamma \in \Gamma$:

$$H(\gamma)(s) = \arg \min_{a \in \tau(s)} \sum_{t \in S} v_\gamma(t) \cdot a(t).$$

We construct a sequence of policies $\gamma_0, \gamma_1, \gamma_2, \ldots$ by letting γ_0 be arbitrary, and for $k \geq 0$, by letting $\gamma_{k+1} = H(\gamma_k)$. The convergence of this sequence follows from the fact that there are only finitely many policies, and from the following lemma, which prevents cycles in the sequence.

Lemma 26 *For any $\gamma_0 \in \Gamma$, let $\gamma_{k+1} = H(\gamma_k)$ for $k \geq 0$. We have that $v_{\gamma_{k+1}} \leq v_{\gamma_k}$ for all $k \geq 0$.*

Proof. The result is a consequence of the fact that $v_{\gamma_{k+1}} = \lim_{n \to \infty} L^n_{\gamma_{k+1}}(\hat{L}(v_{\gamma_k}))$, and of the fact that $L_{\gamma_{k+1}}$ and $\hat{L}$ are monotonic, with respect to the pointwise ordering of $\mathcal{V}_S$. $\square$

The following lemma then enables the computation of $[\![\forall \Diamond_c r]\!]^{\mathrm{f}}$.

Lemma 27 *Let $\gamma \in \Gamma$ be arbitrary, and let $\hat{\gamma} = \lim_{k \to \infty} H^k(\gamma)$. Then, $[\![\forall \Diamond_c r]\!]^{\mathrm{f}} = v_{\hat{\gamma}}$.*

Proof. First, note that the sequence $\{H^k(\gamma)\}_{k\geq 0}$ converges, by Lemma 26. Second, note that since $H(\hat{\gamma}) = \hat{\gamma}$, it must be also $\hat{L}(v_{\hat{\gamma}}) = L_{\hat{\gamma}}(v_{\hat{\gamma}}) = v_{\hat{\gamma}}$, so that $v_{\hat{\gamma}}$ is a fixpoint of $\hat{L}$. $\square$

The set Γ is of size exponential in $\sum_{s\in S} |\tau(s)|$, and this type of policy iteration is not known to terminate in polynomial time. However, the problem can be solved in NPTIME in $|\mathcal{S}|_b$ by guessing an optimal policy.

Lemma 28 *The value $[\![\forall\Diamond_c r]\!]^{\mathrm{f}}$ can be computed in NPTIME in $|\mathcal{S}|_b$.*

Proof. To compute $[\![\forall\Diamond_c r]\!]^{\mathrm{f}}$, we can guess $\gamma \in \Gamma$ and check that $\gamma = H(\gamma)$; we have then that $[\![\forall\Diamond_c r]\!]^{\mathrm{f}} = \mu v.L_\gamma v$. All the required computation can be performed via linear programming. $\square$

3.3.3 Model checking $\forall\triangle$ in both semantics. With the two semantics for $\forall\triangle_c r$ coinciding, a single algorithm suffices for model checking $\forall\triangle$ in both semantics. The fixpoint semantics of this formula immediately suggests an algorithm based on standard methods used for discounted long-run average problems [2]. Expanding the definition (6), we have that $[\![\forall\triangle_c r]\!]$ is the unique fixpoint of the following equations in $v \in \mathcal{V}_S$: for all $s \in S$,

$$v(s) = (1-\alpha)\cdot q(s) + \min_{a\in\tau(s)} \sum_{t\in S} v(t)a(t).$$

The fixpoint can be easily computed by linear programming, again following a standard approach [2].

Lemma 29 *Consider the following linear-programming problem in the set $\{v(s) \mid s \in S\}$ of variables: maximize $\sum_{s\in S} v(s)$ subject to*

$$v(s) \leq (1-\alpha)\cdot q(s) + \alpha\cdot \sum_{t\in S} v(t)\cdot a(t)$$

for all $s \in S$ and all $a \in \tau(s)$. Denoting by $\{\hat{v}(s) \mid s \in S\}$ an optimal solution, we have $[\![\forall\triangle_c r]\!] = v^*$.

3.3.4 Complexity of Dctl model checking over MDPs. The complexity of the model-checking problem for Dctl formulas in MDPs is summarized by the following theorem.

Theorem 7 *Given a* Dctl *formula ϕ, an MDP $\mathcal{S} = (S, \tau, P, [\cdot])$, and a parameter interpretation $\langle\cdot\rangle$, the following assertions hold:*

(1) The problem of computing $[\![\phi]\!]^{\mathrm{p}}$ over $\mathcal{S}$ with respect to $\langle\cdot\rangle$ can be solved in time polynomial in $|\mathcal{S}|_b$ and $|\langle\cdot\rangle|_b$, and exponential in $|\phi|$.

(2) The problem of computing $[\![\phi]\!]^{\mathrm{f}}$ *over* $\mathcal{S}$ *with respect to* $\langle\cdot\rangle$ *can be solved in nondeterministic polynomial time in* $|\mathcal{S}|_b$ *and* $|\langle\cdot\rangle|_b$*, and exponential in* $|\phi|$*.*

The first part of the theorem follows from Theorems 23 and 29; the second part follows from Theorems 24, 28, and 29. Note that the algorithms presented for solving model-checking problem for DCTL in MDPs have different complexities for the path and fixpoint semantics. This contrasts with the situation for transition systems and Markov chains, where we have presented algorithms for DCTL model-checking that are of polynomial time-complexity with respect to the size of the system for both the path and the fixpoint semantics. As in the case of Markov chains, also in MDPs the complexity of the model-checking problem is exponential in the size of the DCTL formula, due to the blow-up of the binary representations of subformula valuations.

4 Conclusions

The traditional theories of discrete transition systems are boolean: the value of a proposition at a state is boolean, and the value of a temporal property at a state is boolean. In boolean theories, property values are sensitive to small perturbations of a system: if the value of a proposition at a single state s is switched, then the value of a temporal property may switch at an arbitrary distance from s. This is problematic, first, because there may be imprecision in models, and second, because engineering artifacts that are based on boolean models are equally fragile.

We built a continuous theory of discrete transition systems by systematically replacing boolean values with real values: the value of a proposition at a state is a real, and so is the value of a temporal property at a state. In a systems theory based on the reals, it is natural to introduce discounting over time, and probabilities over transitions. We achieved continuity in the sense that small perturbations of the reals that specify a system lead to small changes in the values of discounted temporal properties. The resulting theory is therefore robust against imprecisions in measurement and implementation.

We showed that over probabilistic systems, the standard temporal operators can be given two different natural, continuous interpretations: a path semantics and a fixpoint semantics. The fixpoint semantics corresponds to a continuous generalization of state bisimilarity [10], while no such characterization is known for the path semantics. On the other hand, the path semantics gives a natural limit interpretation to infinite behaviors of a system. We presented model-checking algorithms for both semantics, but the question whether the fixpoint semantics of $\forall\Diamond$ properties over MDPs can be computed in polynomial time remains open.

References

[1] C. Baier, B.R. Haverkort, H. Hermanns, and J.-P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *Computer-Aided Verification*, LNCS 1855, pages 358–372. Springer, 2000.

[2] D.P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 1995. Volumes I and II.

[3] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Foundations of Software Technology and Theoretical Computer Science*, LNCS 1026, pages 499–513. Springer, 1995.

[4] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking.* MIT Press, 1999.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms.* McGraw-Hill, 2001.

[6] L. de Alfaro. *Formal Verification of Probabilistic Systems.* PhD thesis. Technical Report STAN-CS-TR-98-1601, Stanford University, 1997.

[7] L. de Alfaro. Quantitative Verification and Control via the Mu-Calculus. In *CONCUR 03: Proceedings of the 14th International Conference,* Lectures Notes in Computer Science 2761, pages 103–127, Springer-Verlag, 2003.

[8] L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. Technical Report UCSC-CRL-03-12, University of California, Santa Cruz, 2003.

[9] L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. In *TACAS04*, LNCS 2988, 2004.

[10] L. de Alfaro, T.A. Henzinger, and R. Majumdar. Discounting the future in systems theory. In *Automata, Languages, and Programming*, LNCS 2719, pages 1022–1037. Springer, 2003.

[11] J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labeled Markov processes. *Information and Computation*, 179:163–193, 2002.

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

[13] H. Hansson. *Time and Probabilities in Formal Design of Distributed Systems.* Elsevier, 1994.

[14] M. Huth and M.Z. Kwiatkowska. Quantitative analysis and model checking. In *Proc. Logic in Computer Science*, pages 111–122. IEEE, 1997.

[15] J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains.* Van Nostrand, 1966.

[16] D. Kozen. A probabilistic PDL. In *Proc. Theory of Computing*, pages 291–297. ACM, 1983.

[17] M.Z. Kwiatkowska. Model checking for probability and time: From theory to practice. In *Proc. Logic in Computer Science*, pages 351–360. IEEE, 2003.

[18] A. Lluch-Lafuente and U. Montanari. Quantitative μ-calculus and CTL based on constraint semirings. In *Proc. 2nd Workshop on Quantitative Aspects of Programming Languages.* ENTCS, 2004.

[19] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer, 1991.

[20] A. McIver. Reasoning about efficiency within a probabilistic μ-calculus. In *Proc. Probabilistic Methods in Verification*, pages 45–58. Technical Report CSR-98-4, University of Birmingham, 1998.

[21] A. McIver and C. Morgan. Games, probability, and the quantitative μ-calculus. In *Logic Programming, Artificial Intelligence, and Reasoning*, LNCS 2514, pages 292–310. Springer, 2002.

[22] D. Williams. *Probability with Martingales.* Cambridge University Press, 1991.

[23] U. Zwick and M.S. Paterson. The complexity of mean-payoff games on graphs. *Theoretical Computer Science*, 158:343–359, 1996.