CEGAR for Qualitative Analysis of Probabilistic Systems[⋆,⋆⋆]

Krishnendu Chatterjee, Martin Chmelík, and Przemysław Daca

IST Austria

Abstract. We consider Markov decision processes (MDPs) which are a standard model for probabilistic systems. We focus on qualitative properties for MDPs that can express that desired behaviors of the system arise almost-surely (with probability 1) or with positive probability. We introduce a new simulation relation to capture the refinement relation of MDPs with respect to qualitative properties, and present discrete graph theoretic algorithms with quadratic complexity to compute the simulation relation. We present an automated technique for assume-guarantee style reasoning for compositional analysis of MDPs with qualitative properties by giving a counterexample guided abstraction-refinement approach to compute our new simulation relation. We have implemented our algorithms and show that the compositional analysis leads to significant improvements.

1 Introduction

Markov decision processes. *Markov decision processes (MDPs)* are standard models for analysis of probabilistic systems that exhibit both probabilistic and nondeterministic behavior [46,39]. In verification of probabilistic systems, MDPs have been adopted as models for concurrent probabilistic systems [32], probabilistic systems operating in open environments [60], under-specified probabilistic systems [9], and applied in diverse domains [6,52] such as analysis of randomized communication and security protocols, stochastic distributed systems, biological systems, etc.

Compositional Analysis and CEGAR. One of the key challenges in analysis of probabilistic systems (as in the case of non-probabilistic systems) is the *state explosion* problem [29], as the size of concurrent systems grows exponentially in the number of components. One key technique to combat the state explosion problem is the *assume-guarantee* style composition reasoning [58], where the analysis problem is decomposed into components and the results for components are used to reason about the whole system, instead of verifying the whole system directly. For a system with two components, the compositional reasoning can be captured as the following simple rule: consider a system with two components G_1 and G_2, and a specification G' to be satisfied by the system; if A is an abstraction of G_2 (i.e., G_2 refines A) and G_1 in composition with A

⋆ The research was partly supported by Austrian Science Fund (FWF) Grant No P 23499- N23, FWF NFN Grant No S11407-N23 and S11402-N23 (RiSE), ERC Start grant (279307: Graph Games), Microsoft faculty fellows award, the ERC Advanced Grant QUAREM (Quantitative Reactive Modeling).

⋆⋆ Full version [15]: Link `http://arxiv.org/abs/1405.0835`

A. Biere and R. Bloem (Eds.): CAV 2014, LNCS 8559, pp. 473–490, 2014.

satisfies G', then the composite systems of G_1 and G_2 also satisfies G'. Intuitively, A is an assumption on G_1's environment that can be ensured by G_2. This simple, yet elegant asymmetric rule is very effective in practice, specially with a *counterexample guided abstraction-refinement* (CEGAR) loop [30]. There are many symmetric [56] as well as circular compositional reasoning [35,56,53] rules; however the simple asymmetric rule is most effective in practice and extensively studied, mostly for non-probabilistic systems [56,38,12,44].

Compositional Analysis for Probabilistic Systems. There are many works that have studied the abstraction-refinement and compositional analysis for probabilistic systems [11,45,51,37]. Our work is most closely related to and inspired by [50] where a CEGAR approach was presented for analysis of MDPs (or labeled probabilistic transition systems); and the refinement relation was captured by *strong simulation* that captures the logical relation induced by safe-pCTL [41,4,9].

Qualitative Analysis and Its Importance. In this work we consider the fragment of pCTL* [41,4,9] that is relevant for *qualitative analysis*, and refer to this fragment as QCTL^*. The qualitative analysis for probabilistic systems refers to *almost-sure* (resp. *positive*) properties that are satisfied with probability 1 (resp. positive probability). The qualitative analysis for probabilistic systems is an important problem in verification that is of interest independent of the quantitative analysis problem. There are many applications where we need to know whether the correct behavior arises with probability 1. For instance, when analyzing a randomized embedded scheduler, we are interested in whether every thread progresses with probability 1 [17]. Even in settings where it suffices to satisfy certain specifications with probability $\lambda < 1$, the correct choice of λ is a challenging problem, due to the simplifications introduced during modeling. For example, in the analysis of randomized distributed algorithms it is quite common to require correctness with probability 1 (see, e.g., [59,62]). Furthermore, in contrast to quantitative analysis, qualitative analysis is robust to numerical perturbations and modeling errors in the transition probabilities. The qualitative analysis problem has been extensively studied for many probabilistic models, such as for MDPs [24,25,26], perfect-information stochastic games [27,13], concurrent stochastic games [36,18], partial-observation MDPs [5,28,16,20], and partial-observation stochastic games [22,8,19,21,55,23].

Our Contributions. In this work we focus on the compositional reasoning of probabilistic systems with respect to qualitative properties, and our main contribution is a CEGAR approach for qualitative analysis of probabilistic systems. The details of our contributions are as follows:

1. To establish the logical relation induced by QCTL^* we consider the logic ATL^* for two-player games and the two-player game interpretation of an MDP where the probabilistic choices are resolved by an adversary. In case of non-probabilistic systems and games there are two classical notions for refinement, namely, *simulation* [54] and *alternating-simulation* [1]. We first show that the logical relation induced by QCTL^* is *finer* than the intersection of simulation and alternating simulation. We then introduce a new notion of simulation, namely, *combined simulation*, and show that it captures the logical relation induced by QCTL^*.

2. We show that our new notion of simulation, which captures the logic relation of QCTL^*, can be computed using discrete graph theoretic algorithms in quadratic time. In contrast, the current best known algorithm for strong simulation is polynomial of degree seven and requires numerical algorithms. The other advantage of our approach is that it can be applied uniformly both to qualitative analysis of probabilistic systems as well as analysis of two-player games (that are standard models for open non-probabilistic systems).
3. We present a CEGAR approach for the computation of combined simulation, and the counterexample analysis and abstraction refinement is achieved using the ideas of [43] proposed for abstraction-refinement for games.
4. We have implemented our approach both for qualitative analysis of MDPs as well as games, and experimented on a number of well-known examples of MDPs and games. Our experimental results show that our method achieves significantly better performance as compared to the non-compositional verification as well as compositional analysis of MDPs with strong simulation.

Related Works. Compositional and assume-guarantee style reasoning has been extensively studied mostly in the context of non-probabilistic systems [56,38,12,44]. Game-based abstraction refinement has been studied in the context of probabilistic systems [51]. The CEGAR approach has been adapted to probabilistic systems for reachability [45] and safe-pCTL [11] under non-compositional abstraction refinement. The work of [50] considers CEGAR for compositional analysis of probabilistic system with strong simulation. An abstraction-refinement algorithm for a class of quantitative properties was studied in [33,34] and also implemented [49]. Our logical characterization of the simulation relation is similar in spirit to [31], which shows how a fragment of the modal μ-calculus can be used to efficiently decide behavioral preorders between components. Our work focuses on CEGAR for compositional analysis of probabilistic systems for qualitative analysis: we characterize the required simulation relation; present a CEGAR approach for the computation of the simulation relation; and show the effectiveness of our approach both for qualitative analysis of MDPs and games.

2 Game Graphs and Alternating-Time Temporal Logics

Notations. Let AP denote a non-empty finite set of atomic propositions. Given a finite set S we will denote by S^* (respectively S^ω) the set of finite (resp. infinite) sequences of elements from S, and let $S^+ = S^* \setminus \{\epsilon\}$, where ϵ is the empty string.

2.1 Two-player Games

Two-player Games. A *two-player* game is a tuple $G = (S, A, \mathsf{Av}, \delta, \mathcal{L}, s_0)$, where

- S is a finite set of states and $s_0 \in S$ is an initial state; and A is a finite set of actions.
- $\mathsf{Av} : S \to 2^A \setminus \emptyset$ is an *action-available* function that assigns to every state $s \in S$ the set $\mathsf{Av}(s)$ of actions available in s.
- $\delta : S \times A \to 2^S \setminus \emptyset$ is a non-deterministic *transition* function that given a state $s \in S$ and an action $a \in \mathsf{Av}(s)$ gives the set $\delta(s, a)$ of successors of s given action a.
- $\mathcal{L} : S \to 2^{\mathsf{AP}}$ is a *labeling* function that labels the states $s \in S$ with the set $\mathcal{L}(s)$ of atomic propositions true at s.

Alternating Games. A two-player game G is *alternating* if in every state either Player 1 or Player 2 can make choices. Formally, for all $s \in S$ we have either (i) $|\mathsf{Av}(s)| = 1$ (then we refer to s as a Player-2 state); or (ii) for all $a \in \mathsf{Av}(s)$ we have $|\delta(s,a)| = 1$ (then we refer to s as a Player-1 state). For technical convenience we consider that in the case of alternating games, there is an atomic proposition $\mathsf{turn} \in \mathsf{AP}$ such that for Player-1 states s we have $\mathsf{turn} \in \mathcal{L}(s)$, and for Player 2 states s' we have $\mathsf{turn} \notin \mathcal{L}(s')$.

Plays. A two-player game is played for infinitely many rounds as follows: the game starts at the initial state, and in every round Player 1 chooses an available action from the current state and then Player 2 chooses a successor state, and the game proceeds to the successor state for the next round. Formally, a *play* in a two-player game is an infinite sequence $\omega = s_0 a_0 s_1 a_1 s_2 a_2 \cdots$ of states and actions such that for all $i \geq 0$ we have that $a_i \in \mathsf{Av}(s_i)$ and $s_{i+1} \in \delta(s_i, a_i)$. We denote by Ω the set of all plays.

Strategies. Strategies are recipes that describe how to extend finite prefixes of plays. Formally, a *strategy* for Player 1 is a function $\sigma : (S \times A)^* \times S \to A$, that given a finite history $w \cdot s \in (S \times A)^* \times S$ of the game gives an action from $\mathsf{Av}(s)$ to be played next. We write Σ for the set of all Player-1 strategies. A strategy for Player 2 is a function $\theta : (S \times A)^+ \to S$, that given a finite history $w \cdot s \cdot a$ of a play selects a successor state from the set $\delta(s,a)$. We write Θ for the set of all Player-2 strategies. *Memoryless* strategies are independent of the history, but depend only on the current state for Player 1 (resp. the current state and action for Player 2) and hence can be represented as functions $S \to A$ for Player 1 (resp. as $S \times A \to S$ for Player 2).

Outcomes. Given a strategy σ for Player 1 and θ for Player 2 the *outcome* is a unique play, denoted as $\mathsf{Plays}(s,\sigma,\theta) = s_0 a_0 s_1 a_1 \cdots$, which is defined as follows: (i) $s_0 = s$; and (ii) for all $i \geq 0$ we have $a_i = \sigma(s_0 a_0 \ldots s_i)$ and $s_{i+1} = \theta(s_0 a_0 \ldots s_i a_i)$. Given a state $s \in S$ we denote by $\mathsf{Plays}(s,\sigma)$ (resp. $\mathsf{Plays}(s,\theta)$) the set of possible plays given σ (resp. θ), i.e., $\bigcup_{\theta' \in \Theta} \mathsf{Plays}(s,\sigma,\theta')$ (resp. $\bigcup_{\sigma' \in \Sigma} \mathsf{Plays}(s,\sigma',\theta)$).

Parallel Composition of Two-Player Games. Given games $G = (S, A, \mathsf{Av}, \delta, \mathcal{L}, s_0)$ and $G' = (S', A, \mathsf{Av}', \delta', \mathcal{L}', s'_0)$ the *parallel composition* of the games $G \parallel G' = (\overline{S}, A, \overline{\mathsf{Av}}, \overline{\delta}, \overline{\mathcal{L}}, \overline{s}_0)$ is defined as follows: (1) The states of the composition are $\overline{S} = S \times S'$. (2) The set of actions is A. (3) For all (s,s') we have $\overline{\mathsf{Av}}((s,s')) = \mathsf{Av}(s) \cap \mathsf{Av}'(s')$. (4) The transition function for a state $(s,s') \in \overline{S}$ and an action $a \in \overline{\mathsf{Av}}((s,s'))$ is defined as $\overline{\delta}((s,s'),a) = \{(t,t') \mid t \in \delta(s,a) \wedge t' \in \delta'(s',a)\}$. (5) The labeling function $\overline{\mathcal{L}}((s,s'))$ is defined as $\mathcal{L}(s) \cup \mathcal{L}'(s')$. (6) The initial state is $\overline{s}_0 = (s_0, s'_0)$.

Remark 1. For simplicity we assume that the set of actions in both components is identical, and for every pair of states the intersection of their available actions is non-empty. Parallel composition can be extended to cases where the sets of actions are different [2].

2.2 Alternating-time Temporal Logic

We consider the Alternating-time Temporal Logic (ATL*) [3] as a logic to specify properties for two-player games.

Syntax. The syntax of the logic is given in positive normal form by defining the set of *path formulas* (φ) and *state formulas* (ψ) according to the following grammar:

$$\begin{aligned} &\text{state formulas:} && \psi ::= q \mid \neg q \mid \psi \vee \psi \mid \psi \wedge \psi \mid \mathrm{PQ}(\varphi) \\ &\text{path formulas:} && \varphi ::= \psi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \,\mathcal{U}\, \varphi \mid \varphi \,\mathcal{W}\, \varphi; \end{aligned}$$

where $q \in \mathsf{AP}$ is an atomic proposition and PQ is a path quantifier. The operators $\bigcirc$ (next), $\mathcal{U}$ (until), and $\mathcal{W}$ (weak until) are the temporal operators. We will use true as a shorthand for $q \vee \neg q$ and false for $q \wedge \neg q$ for some $q \in \mathsf{AP}$. The path quantifiers PQ are as follows: ATL^* path quantifiers: $\langle\langle 1 \rangle\rangle$, $\langle\langle 2 \rangle\rangle$, $\langle\langle 1,2 \rangle\rangle$, and $\langle\langle \emptyset \rangle\rangle$.

Semantics. Given a play $\omega = s_0 a_0 s_1 a_1 \cdots$ we denote by $\omega[i]$ the suffix starting at the i-th state element of the play ω, i.e., $\omega[i] = s_i a_i s_{i+1} a_{i+1} \cdots$. The semantics of path formulas is defined inductively in a standard way. Given a path formula φ, we denote by $[\![\varphi]\!]_G$ the set of plays ω such that $\omega \models \varphi$. We omit the G lower script when the game is clear from context. The semantics of state formulas for ATL^* is defined as follows (the semantics for Boolean formulas is omitted):

$$
\begin{array}{ll}
s \models \langle\langle 1 \rangle\rangle(\varphi) & \text{iff } \exists \sigma \in \Sigma, \forall \theta \in \Theta : \mathsf{Plays}(s,\sigma,\theta) \in [\![\varphi]\!] \\
s \models \langle\langle 2 \rangle\rangle(\varphi) & \text{iff } \exists \theta \in \Theta, \forall \sigma \in \Sigma : \mathsf{Plays}(s,\sigma,\theta) \in [\![\varphi]\!] \\
s \models \langle\langle 1,2 \rangle\rangle(\varphi) & \text{iff } \exists \sigma \in \Sigma, \exists \theta \in \Theta : \mathsf{Plays}(s,\sigma,\theta) \in [\![\varphi]\!] \\
s \models \langle\langle \emptyset \rangle\rangle(\varphi) & \text{iff } \forall \sigma \in \Sigma, \forall \theta \in \Theta : \mathsf{Plays}(s,\sigma,\theta) \in [\![\varphi]\!];
\end{array}
$$

where $s \in S$. Given an ATL^* state formula ψ and a two-player game G, we denote by $[\![\psi]\!]_G = \{s \in S \mid s \models \psi\}$ the set of states that satisfy the formula ψ. We omit the G lower script when the game is clear from context.

Logic Fragments. We define several fragments of the logic ATL^*:

- *Restricted temporal operator use.* An important fragment of ATL^* is ATL where every temporal operator is immediately preceded by a path quantifier.
- *Restricting path quantifiers.* We also consider fragments of ATL^* (resp. ATL) where the path quantifiers are restricted. We consider (i) 1-fragment (denoted 1-ATL^*) where only $\langle\langle 1 \rangle\rangle$ path quantifier is used; (ii) the $(1,2)$-fragment (denoted $(1,2)$-ATL^*) where only $\langle\langle 1,2 \rangle\rangle$ path quantifier is used; and (iii) the combined fragment (denoted C-ATL^*) where both $\langle\langle 1 \rangle\rangle$ and $\langle\langle 1,2 \rangle\rangle$ path quantifiers are used. We use a similar notation for the respective fragments of ATL formulas.

Logical Characterization of States. Given two games G and G', and a logic fragment $\mathcal{F}$ of ATL^*, we consider the following relations on the state space induced by the logic fragment $\mathcal{F}$: $\preccurlyeq_{\mathcal{F}}\ (G, G') = \{(s, s') \in S \times S' \mid \forall \psi \in \mathcal{F} :\ \text{if } s \models \psi \text{ then } s' \models \psi\}$; and when the games are clear from context we simply write $\preccurlyeq_{\mathcal{F}}$ for $\preccurlyeq_{\mathcal{F}}\ (G, G')$. We will use the following notations for the relation induced by the logic fragments we consider: (i) $\preccurlyeq^*_1$ (resp. $\preccurlyeq_1$) for the relation induced by the 1-ATL^* (resp. 1-ATL) fragment; (ii) $\preccurlyeq^*_{1,2}$ (resp. $\preccurlyeq_{1,2}$) for the relation induced by the $(1,2)$-ATL^* (resp. $(1,2)$-ATL) fragment; and (iii) $\preccurlyeq^*_C$ (resp. $\preccurlyeq_C$) for the relation induced by the C-ATL^* (resp. C-ATL) fragment. Given G and G' we can also consider G'' which is the disjoint union of the two games, and consider the relations on G''; and hence we will often consider a single game as input for the relations.

3 Combined Simulation Relation Computation

In this section we first recall the notion of simulation [54] and alternating simulation [1]; and then present a new notion of *combined simulation*.

Simulation. Given two-player games $G = (S, A, \mathsf{Av}, \delta, \mathcal{L}, s_0)$ and $G' = (S', A', \mathsf{Av}', \delta', \mathcal{L}', s'_0)$, a relation $\mathcal{S} \subseteq S \times S'$ is a *simulation* from G to G' if for all $(s, s') \in \mathcal{S}$ the following conditions hold:

1. *Proposition match:* The atomic propositions match, i.e., $\mathcal{L}(s) = \mathcal{L}'(s')$.
2. *Step-wise simulation condition:* For all actions $a \in \mathsf{Av}(s)$ and states $t \in \delta(s,a)$ there exists an action $a' \in \mathsf{Av}'(s')$ and a state $t' \in \delta(s',a')$ such that $(t,t') \in \mathcal{S}$.

We denote by $\mathcal{S}^{G,G'}_{\max}$ the largest simulation relation between the two games (we write $\mathcal{S}_{\max}$ instead of $\mathcal{S}^{G,G'}_{\max}$ when G and G' are clear from the context). We write $G \sim_{\mathcal{S}} G'$ when $(s_0, s'_0) \in \mathcal{S}_{\max}$. The largest simulation relation characterizes the logic relation of $(1,2)$-ATL and $(1,2)$-ATL*: the $(1,2)$-ATL fragment interprets a game as a transition system and the formulas coincide with existential CTL, and hence the logic characterization follows from the classical results on simulation and CTL [54,2].

Proposition 1. *For all games G and G' we have $\mathcal{S}_{\max} = \preccurlyeq^*_{1,2} = \preccurlyeq_{1,2}$.*

Alternating Simulation. Given two games $G = (S, A, \mathsf{Av}, \delta, \mathcal{L}, s_0)$ and $G' = (S', A', \mathsf{Av}', \delta', \mathcal{L}', s'_0)$, a relation $\mathcal{A} \subseteq S \times S'$ is an *alternating simulation* from G to G' if for all $(s,s') \in \mathcal{A}$ the following conditions hold:

1. *Proposition match:* The atomic propositions match, i.e., $\mathcal{L}(s) = \mathcal{L}'(s')$.
2. *Step-wise alternating-simulation condition:* For all actions $a \in \mathsf{Av}(s)$ there exists an action $a' \in \mathsf{Av}'(s')$ such that for all states $t' \in \delta'(s',a')$ there exists a state $t \in \delta(s,a)$ such that $(t,t') \in \mathcal{A}$.

We denote by $\mathcal{A}^{G,G'}_{\max}$ the largest alternating-simulation relation between the two games (we write $\mathcal{A}_{\max}$ instead of $\mathcal{A}^{G,G'}_{\max}$ when G and G' are clear from the context). We write $G \sim_{\mathcal{A}} G'$ when $(s_0, s'_0) \in \mathcal{A}_{\max}$. The largest alternating-simulation relation characterizes the logic relation of 1-ATL and 1-ATL* [1].

Proposition 2. *For all games G and G' we have $\mathcal{A}_{\max} = \preccurlyeq^*_1 = \preccurlyeq_1$.*

Combined Simulation. We present a new notion of combined simulation that extends both simulation and alternating simulation, and we show how the combined simulation characterizes the logic relation induced by C-ATL* and C-ATL. Intuitively, the requirements on the combined-simulation relation combine the requirements imposed by alternating simulation and simulation in a step-wise fashion. Given two-player games $G = (S, A, \mathsf{Av}, \delta, \mathcal{L}, s_0)$ and $G' = (S', A', \mathsf{Av}', \delta', \mathcal{L}', s'_0)$, a relation $\mathcal{C} \subseteq S \times S$ is a *combined simulation* from G to G' if for all $(s,s') \in \mathcal{C}$ the following conditions hold:

1. *Proposition match:* The atomic propositions match, i.e., $\mathcal{L}(s) = \mathcal{L}'(s')$.
2. *Step-wise simulation condition:* For all actions $a \in \mathsf{Av}(s)$ and states $t \in \delta(s,a)$ there exists an action $a' \in \mathsf{Av}'(s')$ and a state $t' \in \delta(s',a')$ such that $(t,t') \in \mathcal{C}$.
3. *Step-wise alternating-simulation condition:* For all actions $a \in \mathsf{Av}(s)$ there exists an action $a' \in \mathsf{Av}'(s')$ such that for all states $t' \in \delta'(s',a')$ there exists a state $t \in \delta(s,a)$ such that $(t,t') \in \mathcal{C}$.

We denote by $\mathcal{C}^{G,G'}_{\max}$ the largest combined-simulation relation between the two games (and write $\mathcal{C}_{\max}$ when G and G' are clear from the context). We also write $G \sim_{\mathcal{C}} G'$ when $(s_0, s'_0) \in \mathcal{C}_{\max}$. We first illustrate with an example that the logic relation $\preccurlyeq_C$ induced by C-ATL is finer than the intersection of simulation and alternating-simulation relation; then present a game theoretic characterization of $\mathcal{C}_{\max}$; and finally show that $\mathcal{C}_{\max}$ gives the relations $\preccurlyeq^*_C$ and $\preccurlyeq_C$.

Example 1. Consider the games G and G' shown in Figure 1. White nodes are labeled by an atomic proposition p and gray nodes by q. The largest simulation and alternating-simulation relations between G and G' are: $\mathcal{S}_{\max} = \{(s_0,t_0),(s_1,t_1)\}, \mathcal{A}_{\max} =$

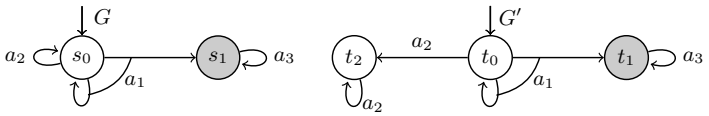

Fig. 1. Games G, G' such that $G \sim_{\mathcal{S}} G'$ and $G \sim_{\mathcal{A}} G'$, but $G \not\sim_{\mathcal{C}} G'$

$\{(s_0, t_0), (s_0, t_2), (s_1, t_1)\}$. However, consider the formula $\psi = \langle\langle 1 \rangle\rangle(\bigcirc(p \wedge \langle\langle 1, 2 \rangle\rangle(\bigcirc q)))$. We have that $s_0 \models \psi$, but $t_0 \not\models \psi$. It follows that $(s_0, t_0) \notin \preccurlyeq_C$. □

Combined-Simulation Games. The simulation and the alternating-simulation relation can be obtained by solving two-player safety games [42,1,14]. We now define a two-player game $G^{\mathcal{C}}$ for the combined-simulation relation characterization. The game is played on the synchronized product of the two input games. Given a state (s, s'), first Player 2 decides whether to check for the step-wise simulation condition or the step-wise alternating-simulation condition.

1. The step-wise simulation condition is checked by playing a two-step game. Intuitively, first Player 2 chooses an action $a \in \mathsf{Av}(s)$ and a successor $t \in \delta(s, a)$ and challenges Player 1 to match, and Player 1 responds with an action $a' \in \mathsf{Av}'(s')$ and a state $t' \in \delta'(s', a')$.
2. The step-wise alternating-simulation condition is checked by playing a four-step game. Intuitively, first Player 2 chooses an action a from $\mathsf{Av}(s)$ and Player 1 responds with an action $a' \in \mathsf{Av}'(s')$ (in the first two-steps); then Player 2 chooses a successor $t' \in \delta'(s', a')$ and Player 1 responds by choosing a successor $t \in \delta(s, a)$.

After checking the step-wise conditions, the game proceeds from the state (t, t'). Intuitively, Player 2's goal is to reach a state (s, s') where the labeling of the original games do not match; states that satisfy this condition are labeled by atomic proposition p.

In the combined simulation game we refer to Player 1 as the *proponent* (trying to establish the combined simulation) and Player 2 as the *adversary* (trying to violate the combined simulation).

Shorthand for Safety Objectives. We will use the following shorthand for *safety* objectives: $\Box\, \varphi \equiv \varphi\, \mathcal{W}\, \mathsf{false}$.

Theorem 1. *For all games G and G' we have $\mathcal{C}_{\max} = [\![\langle\langle 1 \rangle\rangle(\Box \neg p)]\!]_{G^{\mathcal{C}}} \cap (S \times S')$.*

We establish the relation between combined simulation and C-ATL*.

Theorem 2. *For all games G and G' we have $\mathcal{C}_{\max} = \preccurlyeq_C^* = \preccurlyeq_C$.*

Remark 2. Theorem 2 also holds for alternating games. Note that in most cases the action set is constant and the state space of the games are huge. Then the combined simulation game construction is quadratic, and solving safety games on them can be achieved in linear time using discrete graph theoretic algorithms [47,7].

Theorem 3. *Given two-player games G and G', the $\mathcal{C}_{\max}$, $\preccurlyeq_C^*$, and $\preccurlyeq_C$ relations can be computed in quadratic time using discrete graph theoretic algorithms.*

4 MDPs and Qualitative Logics

In this part we consider Markov decisions processes (MDPs) and logics to reason qualitatively about them. We consider MDPs which can be viewed as a variant of two-player games defined in Section 2. First, we fix some notation: a probability distribution f on a finite set X is a function $f : X \to [0,1]$ such that $\sum_{x \in X} f(x) = 1$, and we denote by $\mathcal{D}(X)$ the set of all probability distributions on X. For $f \in \mathcal{D}(X)$ we denote by $\mathrm{Supp}(f) = \{x \in X \mid f(x) > 0\}$ the *support of* f.

4.1 Markov Decision Processes

MDPs. A *Markov decision process* (MDP) is a tuple $G = (S, (S_1, S_P), A, \mathsf{Av}, \delta_1, \delta_P, \mathcal{L}, s_0)$; where (i) S is a finite set of states with a partition of S into Player-1 states S_1 and probabilistic states S_P; (ii) A is a finite set of actions; (iii) $\mathsf{Av} : S_1 \to 2^A \setminus \emptyset$ is an action-available function that assigns to every Player-1 state the non-empty set $\mathsf{Av}(s)$ of actions available in s; (iv) $\delta_1 : S_1 \times A \to S$ is a deterministic transition function that given a Player-1 state and an action gives the next state; (v) $\delta_P : S_P \to \mathcal{D}(S)$ is a probabilistic transition function that given a probabilistic state gives a probability distribution over the successor states (i.e., $\delta_P(s)(s')$ is the transition probability from s to s'); (vi) the function $\mathcal{L}$ is the proposition labeling function as for two-player games; and (vii) s_0 is the initial state. Strategies for Player 1 are defined as for games.

Interpretations. We interpret an MDP in two distinct ways: (i) as a $1\frac{1}{2}$-player game and (ii) as an alternating two-player game. In the $1\frac{1}{2}$-player setting in a state $s \in S_1$, Player 1 chooses an action $a \in \mathsf{Av}(s)$ and the MDP moves to a unique successor s'. In probabilistic states $s_p \in S_P$ the successor is chosen according to the probability distribution $\delta_P(s_p)$. In the alternating two-player interpretation, we regard the probabilistic states as Player-2 states, i.e., in a state $s_p \in S_P$, Player 2 chooses a successor state s' from the support of the probability distribution $\delta_P(s)$. Given an MDP G we denote by $\widehat{G}$ its two-player interpretation, and $\widehat{G}$ is an alternating game. The $1\frac{1}{2}$-player interpretation is the classical definition of MDPs. We will use the two-player interpretation to relate logical characterizations of MDPs and logical characterization of two-player games with fragments of ATL^*.

$1\frac{1}{2}$-Player Interpretation. Once a strategy $\sigma \in \Sigma$ for Player 1 is fixed, the outcome of the MDP is a random walk for which the probabilities of *events* are uniquely defined, where an *event* $\Phi \subseteq \Omega$ is a measurable set of plays [40]. For a state $s \in S$ and an event $\Phi \subseteq \Omega$, we write $\Pr_s^\sigma(\Phi)$ for the probability that a play belongs to Φ if the game starts from the state s and Player 1 follows the strategy σ.

4.2 Qualitative Logics for MDPs

We consider the qualitative fragment of pCTL^* [41,4,9] and refer to the logic as *qualitative pCTL** (denoted as QCTL^*) as it can express qualitative properties of MDPs.

Syntax and Semantics. The syntax of the logic is given in positive normal form and is similar to the syntax of ATL^*. It has the same state and path formulas as ATL^* with the exception of path quantifiers. The logic QCTL^* comes with two path quantifiers (PQ), namely $\langle \mathsf{Almost} \rangle$ and $\langle \mathsf{Positive} \rangle$ (instead of $\langle\langle 1 \rangle\rangle$, $\langle\langle 2 \rangle\rangle$, $\langle\langle 1,2 \rangle\rangle$, and $\langle\langle \emptyset \rangle\rangle$). The semantics of the logic QCTL^* is the same for the fragment shared with ATL^*, therefore we only give semantics for the new path quantifiers. Given a path formula φ, we denote

by $[\![\varphi]\!]_G$ the set of plays ω such that $\omega \models \varphi$. For a state s and a path formula φ we have: $s \models \langle\mathsf{Almost}\rangle(\varphi)$ (resp. $s \models \langle\mathsf{Positive}\rangle(\varphi)$) iff $\exists \sigma \in \Sigma : \Pr_s^\sigma([\![\varphi]\!]) = 1$ (resp. $\Pr_s^\sigma([\![\varphi]\!]) > 0$). As before, we denote by QCTL the fragment of QCTL* where every temporal operator is immediately preceded by a path quantifier, and for a state formula ψ the set $[\![\psi]\!]_G$ denotes the set of states in G that satisfy the formula ψ.

Logical Relation Induced by QCTL **and** QCTL***.** Given two MDPs G and G', the logical relation induced by QCTL*, denoted as $\preccurlyeq_Q^*$, (resp. by QCTL, denoted as $\preccurlyeq_Q$), is defined as: $\preccurlyeq_Q^* = \{(s, s') \in S \times S' \mid \forall \psi \in \text{QCTL}^* : \text{ if } s \models \psi \text{ then } s' \models \psi\}$ (resp. $\forall \psi \in$ QCTL).

4.3 Characterization of Qualitative Simulation for MDPs

In this section we establish the equivalence of the $\preccurlyeq_Q^*$ relation on MDPs with the $\preccurlyeq_C^*$ relation on the two-player interpretation of MDPs, i.e., we prove that for all MDPs G and G' we have $\preccurlyeq_Q^* (G, G') = \preccurlyeq_C (\widehat{G}, \widehat{G'})$, where $\widehat{G}$ (resp. $\widehat{G'}$) is the two-player interpretation of the MDP G (resp. G'). In the first step we show how to translate some of the QCTL formulas into C-ATL formulas. We only need to translate the path quantifiers due to the similarity of path formulas in the logics.

Lemma 1. *For all atomic propositions q, r and for all MDPs, we have:*
(i) $[\![\langle\mathsf{Almost}\rangle(\bigcirc q)]\!] = [\![\langle\langle 1\rangle\rangle(\bigcirc q)]\!]$; *(ii)* $[\![\langle\mathsf{Almost}\rangle(q \mathcal{W} r)]\!] = [\![\langle\langle 1\rangle\rangle(q \mathcal{W} r)]\!]$;
(iii) $[\![\langle\mathsf{Positive}\rangle(\bigcirc q)]\!] = [\![\langle\langle 1,2\rangle\rangle(\bigcirc q)]\!]$; *(iv)* $[\![\langle\mathsf{Positive}\rangle(q \mathcal{U} r)]\!] = [\![\langle\langle 1,2\rangle\rangle(q \mathcal{U} r)]\!]$;
(v) $[\![\langle\mathsf{Positive}\rangle(q \mathcal{W} r)]\!] = [\![\langle\langle 1,2\rangle\rangle(q \mathcal{U} r)]\!] \cup [\![\langle\langle 1,2\rangle\rangle(q \mathcal{U}(\langle\langle 1\rangle\rangle(q \mathcal{W}\mathsf{false})))]\!]$.

To complete the translation of temporal operators we also express the QCTL formula $[\![\langle\mathsf{Almost}\rangle(q \mathcal{U} r)]\!]$ in terms of C-ATL* [15]. We establish the following result.

Theorem 4. *For all MDPs G and G' we have $\preccurlyeq_Q = \preccurlyeq_C$; and $\preccurlyeq_Q^* = \preccurlyeq_Q$. The relation $\preccurlyeq_Q^*$ can be computed in quadratic time using discrete graph theoretic algorithms.*

5 CEGAR for Combined Simulation

In this section we present a CEGAR approach for computing combined simulation.

5.1 Simulation Abstraction and Alternating-Simulation Abstraction

Abstraction. An *abstraction* of a game consists of a partition of the game graph such that in each partition the atomic proposition labeling match for all states. Given an abstraction of a game, the abstract game can be defined by collapsing states of each partition and redefining the action-available and transition functions. The redefinition of the action-available and transition functions can either increase or decrease the power of the players. If we increase the power of Player 1 and decrease the power of Player 2, then the abstract game will be in alternating simulation with the original game, and if we increase the power of both players, then the abstract game will simulate the original game. We now formally define the partitions, and the two abstractions.

Partitions for Abstraction. A *partition* of a game $G = (S, A, \mathsf{Av}, \delta, \mathcal{L}, s_0)$ is an equivalence relation $\Pi = \{\pi_1, \pi_2, \ldots, \pi_k\}$ on S such that: (i) for all $1 \leq i \leq k$ we have $\pi_i \subseteq S$ and for all $s, s' \in \pi_i$ we have $\mathcal{L}(s) = \mathcal{L}(s')$ (labeling match); (ii) $\bigcup_{1 \leq i \leq k} \pi_i = S$ (covers the state space); and (iii) for all $1 \leq i, j \leq k$, such that $i \neq j$

we have $\pi_i \cap \pi_j = \emptyset$ (disjoint). Note that in alternating games Player 1 and Player 2 states are distinguished by proposition turn, so they belong to different partitions.

Simulation Abstraction. Given a two-player game $G = (S, A, \mathsf{Av}, \delta, \mathcal{L}, s_0)$ and a partition Π of G, we define the *simulation abstraction of* G as a two-player game $Abs^{\Pi}_{\mathcal{S}}(G) = (\overline{S}, A, \overline{\mathsf{Av}}, \overline{\delta}, \overline{\mathcal{L}}, \overline{s}_0)$, where: (i) $\overline{S} = \Pi$: the partitions in Π are the states of the abstract game. (ii) For all $\pi_i \in \Pi$ we have $\overline{\mathsf{Av}}(\pi_i) = \bigcup_{s \in \pi_i} \mathsf{Av}(s)$: the set of available actions is the union of the actions available to the states in the partition, and this gives more power to Player 1. (iii) For all $\pi_i \in \Pi$ and $a \in \overline{\mathsf{Av}}(\pi_i)$ we have $\overline{\delta}(\pi_i, a) = \{\pi_j \mid \exists s \in \pi_i : \ (a \in \mathsf{Av}(s) \wedge \exists s' \in \pi_j : \ s' \in \delta(s, a))\}$: there is a transition from a partition π_i given an action a to a partition π_j if some state $s \in \pi_i$ can make an a-transition to some state in $s' \in \pi_j$. This gives more power to Player 2. (iv) For all $\pi_i \in \Pi$ we have $\overline{\mathcal{L}}(\pi_i) = \mathcal{L}(s)$ for some $s \in \pi_i$: the abstract labeling is well-defined, since all states in a partition are labeled by the same atomic propositions. (v) $\overline{s}_0$ is the partition in Π that contains state s_0.

Alternating-Simulation Abstraction. Given a two-player game $G = (S, A, \mathsf{Av}, \delta, \mathcal{L}, s_0)$ and a partition Π of G, we define the *alternating-simulation abstraction of* G as a two-player game $Abs^{\Pi}_{\mathcal{A}}(G) = (\widetilde{S}, A, \widetilde{\mathsf{Av}}, \widetilde{\delta}, \widetilde{\mathcal{L}}, \widetilde{s}_0)$, where: (i) $\widetilde{S} = \Pi$; (ii) for all $\pi_i \in \Pi$ we have $\widetilde{\mathsf{Av}}(\pi_i) = \bigcup_{s \in \pi_i} \mathsf{Av}(s)$; (iii) for all $\pi_i \in \Pi$ we have $\widetilde{\mathcal{L}}(\pi_i) = \mathcal{L}(s)$ for some $s \in \pi_i$; (iv) $\widetilde{s}_0$ is the partition in Π that contains state s_0 (as in the case of simulation abstraction). (v) For all $\pi_i \in \Pi$ and $a \in \widetilde{\mathsf{Av}}(\pi_i)$ we have $\widetilde{\delta}(\pi_i, a) = \{\pi_j \mid \forall s \in \pi_i : \ (a \in \mathsf{Av}(s) \wedge \ \exists s' \in \pi_j : \ s' \in \delta(s, a))\}$: there is a transition from a partition π_i given an action a to a partition π_j if all states $s \in \pi_i$ can make an a-transition to some state in $s' \in \pi_j$. This gives less power to Player 2. For technical convenience we assume $\widetilde{\delta}(\pi_i, a)$ is non-empty.

The following proposition states that (alternating-)simulation abstraction of a game G is in (alternating-)simulation with G.

Proposition 3. *For all partitions Π of a two-player game G we have: (1) $G \sim_{\mathcal{A}} Abs^{\Pi}_{\mathcal{A}}(G)$; and (2) $G \sim_{\mathcal{S}} Abs^{\Pi}_{\mathcal{S}}(G)$.*

5.2 Sound Assume-Guarantee Rule

We now present the sound assume-guarantee rule for the combined-simulation problem. To achieve this we first need an extension of the notion of combined-simulation game.

Modified Combined-Simulation Games. Consider games $G^{\mathsf{Alt}} = (S, A, \delta^{\mathsf{Alt}}, \mathsf{Av}^{\mathsf{Alt}}, \mathcal{L}, s_0)$, $G^{\mathsf{Sim}} = (S, A, \delta^{\mathsf{Sim}}, \mathsf{Av}^{\mathsf{Sim}}, \mathcal{L}, s_0)$ and $G' = (S', A, \delta', \mathsf{Av}', \mathcal{L}', s'_0)$. The *modified simulation game* $G^{\mathcal{M}} = (S^{\mathcal{M}}, A^{\mathcal{M}}, \mathsf{Av}^{\mathcal{M}}, \delta^{\mathcal{M}}, \mathcal{L}^{\mathcal{M}}, s_0^{\mathcal{M}})$ is defined exactly like the combined simulation game given G^{Alt} and G', with the exception that the step-wise simulation gadget is defined using the transitions of G^{Sim} instead of G^{Alt}. We write $(G^{\mathsf{Alt}} \otimes G^{\mathsf{Sim}}) \sim_{\mathcal{M}} G'$ if and only if $(s_0, s'_0) \in [\![\langle\langle 1 \rangle\rangle(\Box\neg p)]\!]_{G^{\mathcal{M}}}$.

Proposition 4. *Let $G, G', G^{\mathsf{Alt}}, G^{\mathsf{Sim}}$ be games such that $G \sim_{\mathcal{A}} G^{\mathsf{Alt}}$ and $G \sim_{\mathcal{S}} G^{\mathsf{Sim}}$. Then $(G^{\mathsf{Alt}} \otimes G^{\mathsf{Sim}}) \sim_{\mathcal{M}} G'$ implies $G \sim_{\mathcal{C}} G'$.*

The key proof idea for the above proposition is as follows: if $G \sim_{\mathcal{A}} G^{\mathsf{Alt}}$ and $G \sim_{\mathcal{S}} G^{\mathsf{Sim}}$, then in the modified combined-simulation game $G^{\mathcal{M}}$ the adversary is stronger

than in the combined-simulation game $G^{\mathcal{C}}$. Hence winning in $G^{\mathcal{M}}$ for the proponent implies winning in $G^{\mathcal{C}}$ and gives the desired result of the proposition.

Sound Assume-Guarantee Method. Given two games G_1 and G_2, checking whether their parallel composition $G_1 \parallel G_2$ is in combined simulation with a game G' can be done explicitly by constructing the synchronized product. The composition, however, may be much larger than the components and thus make the method ineffective in practical cases. We present an alternative method that proves combined simulation in a compositional manner, by abstracting G_2 with some partition Π and then composing it with G_1. The sound assume-guarantee rule follows from Propositions 3 and 4.

Proposition 5 (Sound assume-guarantee rule). *Given games G_1, G_2, G', and a partition Π of G_2, let $\mathsf{A} = G_1 \parallel Abs^{\Pi}_{\mathcal{A}}(G_2)$ and $\mathsf{S} = G_1 \parallel Abs^{\Pi}_{\mathcal{S}}(G_2)$. If $(\mathsf{A} \otimes \mathsf{S}) \sim_{\mathcal{M}} G'$, then $(G_1 \parallel G_2) \sim_{\mathcal{C}} G'$, i.e.,*

$$\frac{\mathsf{A} = G_1 \parallel Abs^{\Pi}_{\mathcal{A}}(G_2); \quad \mathsf{S} = G_1 \parallel Abs^{\Pi}_{\mathcal{S}}(G_2); \quad (\mathsf{A} \otimes \mathsf{S}) \sim_{\mathcal{M}} G'}{(G_1 \parallel G_2) \sim_{\mathcal{C}} G'} \quad (1)$$

If the partition Π is coarse, then the abstractions in the assume-guarantee rule can be smaller than G_2 and also their composition with G_1. As a consequence, combined simulation can be proved faster as compared to explicitly computing the composition. In Section 5.4 we describe how to effectively compute the partitions Π and refine them using CEGAR approach.

5.3 Counter-examples Analysis

If the premise $(\mathsf{A} \otimes \mathsf{S}) \sim_{\mathcal{M}} G'$ of the assume-guarantee rule (1) is not satisfied, then the adversary (Player 2) has a memoryless winning strategy in $G^{\mathcal{M}}$, and the memoryless strategy is the *counter-example*. To use the sound assume-guarantee rule (1) in a CEGAR loop, we need analysis of counter-examples.

Representation of counter-examples. A counter-example is a memoryless winning strategy for Player 2 in $G^{\mathcal{M}}$. Note that in $G^{\mathcal{M}}$ Player 2 has a reachability objective, and thus a winning strategy ensures that the target set is always reached from the starting state, and hence no cycle can be formed without reaching the target state once the memoryless winning strategy is fixed. Hence we represent counter-examples as directed-acyclic graphs (DAG), where the leafs are the target states and every non-leaf state has a single successor chosen by the strategy of Player 2 and has all available actions for Player 1.

Abstract, concrete, and spurious counter-examples. Given two-player games G_1 and G_2, let $G = (G_1 \parallel G_2)$ be the parallel composition. Given G and G', let $G^{\mathcal{C}}$ be the combined-simulation game of G and G'. The abstract game $G^{\mathcal{M}}$ is the modified combined-simulation game of $(\mathsf{A} \otimes \mathsf{S})$ and G', where $\mathsf{A} = G_1 \parallel Abs^{\Pi}_{\mathcal{A}}(G_2)$ and $\mathsf{S} = G_1 \parallel Abs^{\Pi}_{\mathcal{S}}(G_2)$. We refer to a counter-example θ_{abs} in $G^{\mathcal{M}}$ as *abstract*, and to a counter-example θ_{con} in $G^{\mathcal{C}}$ as *concrete*. An abstract counter-example is *feasible* if we can substitute partitions in A and S with states of G_2 to obtain a concrete counter-example. An abstract counter-example is *spurious* if it is not feasible.

Concretization of counter-examples. We follow the approach of [43] to check the feasibility of a counter-example by finding a *concretization* function Conc from states in

$G^{\mathcal{M}}$ to a set of states in G_2 that witness a concrete strategy from the abstract strategy. A state in $G^{\mathcal{M}}$ has a component which is a partition for G_2, and the concretization constructs a subset of the partition. Intuitively, for a state $\overline{s}$ of $G^{\mathcal{M}}$ in the counter-example DAG, the concretization represents the subset of states of G_2 in the partition where a concrete winning strategy exists using the strategy represented by the DAG below the state $\overline{s}$. Informally, the witness concrete strategy is constructed inductively, going bottom-up in the DAG as follows: (i) the leaves already represents winning states and hence their concretization is the entire partition; (ii) for non-leaf states in the DAG of the abstract counter-example, the concretization represents the set of states of G_2 of the partition which lead to a successor state that belongs to the concretization of the successor in the DAG. An abstract counter-example is feasible, if the concretization of the root of the DAG contains the initial state of G_2.

5.4 CEGAR

The counter-example analysis presented in the previous section allows us to automatically refine abstractions using the CEGAR paradigm [30]. The algorithm takes games G_1, G_2, G' as arguments and answers whether $(G_1 \parallel G_2) \sim_{\mathcal{C}} G'$ holds. Initially, the algorithms computes the coarsest partition Π of G_2. Then, it executes the CEGAR loop: in every iteration the algorithm constructs A (resp. S) as the parallel composition of G_1 and the alternating-simulation abstraction (resp. simulation abstraction) of G_2. Let $G^{\mathcal{M}}$ be the modified combined-simulation game of $(\mathsf{A} \otimes \mathsf{S})$ and G'. If Player 1 has a winning strategy in $G^{\mathcal{M}}$ then the algorithm returns YES; otherwise it finds an abstract counter-example Cex in $G^{\mathcal{M}}$. In case Cex is feasible, then it corresponds to a concrete counter-example, and the algorithm returns NO. If Cex is spurious a refinement procedure is called that uses the concretization of Cex to return a partition Π' finer than Π.

Refinement Procedure. Given a partition Π and a spurious counter-example Cex together with its concretization function Conc we describe how to compute the refined partition Π'. Consider a partition $\pi \in \Pi$ and let $\overline{S}_\pi = \{\overline{s}_1, \overline{s}_2, \ldots, \overline{s}_m\}$ denote the states of the abstract counter-example Cex that contain π as its component. Every state $\overline{s}_i$ splits π into at most two sets $\mathsf{Conc}(\overline{s}_i)$ and $\pi \setminus \mathsf{Conc}(\overline{s}_i)$, and let this partition be denoted as T_i. We define a partition $\mathcal{P}_\pi$ as the largest equivalence relation on π that is finer than any equivalence relation T_i for all $1 \leq i \leq m$. Formally, $\mathcal{P}_\pi = \{\overline{\pi}_1, \overline{\pi}_2, \ldots, \overline{\pi}_k\}$ is a partition of π such that for all $1 \leq j \leq k$ and $1 \leq i \leq m$ we have $\overline{\pi}_j \subseteq \mathsf{Conc}(\overline{s}_i)$ or $\overline{\pi}_j \subseteq \pi \setminus \mathsf{Conc}(\overline{s}_i)$. The new partition Π' is then defined as the union over $\mathcal{P}_\pi$ for all $\pi \in \Pi$.

Proposition 6. *Given a partition Π and a spurious counter-example* Cex*, the partition Π' obtained as refinement of Π is finer than Π.*

Since we consider finite games, the refinement procedure only executes for finitely many steps and hence the CEGAR loop eventually terminates.

6 Experimental Results

We implemented our CEGAR approach for combined simulation in Java, and experimented with our tool on a number of MDPs and two-player games examples. We use PRISM [52] model checker to specify the examples and generate input files for our tool.

Observable actions. To be compatible with the existing benchmarks (e.g. [50]) in our tool actions are observable instead of atomic propositions. Our algorithms are easily adapted to this setting. We also allow the user to specify silent actions for components, which are not required to be matched by the specification G'.

Improved (modified) combined-simulation game. We leverage the fact that MDPs are interpreted as alternating games to simplify the (modified) combined-simulation game. When comparing two Player-1 states, the last two steps in the alternating-simulation gadget can be omitted, since the players have unique successors given the actions chosen in the first two steps. Similarly, for two probabilistic states, the first two steps in the alternating-simulation gadget can be skipped.

Improved partition refinement procedure. In the implementation we adopt the approach of [43] for refinement. Given a state $\overline{s}$ of the abstract counter-example with partition π as its component, the equivalence relation may split the set $\pi \setminus \mathsf{Conc}(\overline{s})$ into multiple equivalence classes. Intuitively, this ensures that similar-shaped spurious counter-examples do not reappear in the following iterations. This approach is more efficient than the naive one, and also implemented in our tool.

MDP Examples. We used our tool on all the MDP examples from [50]:

- CS_1 *and* CS_n model a Client-Server protocol with mutual exclusion with probabilistic failures in one or all of the n clients, respectively.
- MER is an arbiter module of NASAs software for Mars Exploration Rovers which grants shared resources for several users.
- SN models a network of sensors that communicate via a bounded buffer with probabilistic behavior in the components.

In addition, we also considered two other classical MDP examples:

- LE is based on a PRISM case study [52] that models the *Leader election protocol* [48], where n agents on a ring randomly pick a number from a pool of K numbers. The agent with the highest number becomes the leader. In case there are multiple agents with the same highest number the election proceed to the next round. The specification requires that two leaders cannot be elected at the same time. The MDP is parametrized by the number of agents and the size of the pool.
- PETP is based on a Peterson's algorithm [57] for mutual exclusion of n threads, where the execution order is controlled by a randomized scheduler. The specification requires that two threads cannot access the critical section at the same time. We extend Peterson's algorithm by giving the threads a non-deterministic choice to restart before entering the critical section. The restart operation succeeds with probability $\frac{1}{2}$ and with probability $\frac{1}{2}$ the thread enters the critical section.

Details of experimental results. Table 1 shows the results for MDP examples we obtained using our assume-guarantee algorithm and the monolithic approach (where the composition is computed explicitly). We also compared our results with the tool presented in [50] that implements both assume-guarantee and monolithic approaches for *strong simulation* [61]. All the results were obtained on a Ubuntu-13.04 64-bit machine running on an Intel Core i5-2540M CPU of 2.60GHz. We imposed a 4.3GB upper bound on Java heap memory and one hour time limit. For $\mathsf{MER}(6)$ and $\mathsf{PETP}(5)$ PRISM cannot parse the input file (probably it runs out of memory).

Table 1. Results for MDPs examples: AGCS stands for our assume-guarantee combined simulation; AGSS stands for assume-guarantee with strong simulation; MONCS stands for our monolithic combined simulation; and MONSS stands for monolithic strong simulation. The number I denotes the number of CEGAR iterations and $|\Pi|$ the size of the abstraction in the last CEGAR iteration. TO and MO stand for a time-out and memory-out, respectively, and Error means an error occurred during execution. The memory consumption is measured using the `time` command.

				AGCS				AGSS				MONCS		MONSS	
Ex.	$\|G_1\|$	$\|G_2\|$	$\|G'\|$	$Time$	Mem	I	$\|\Pi\|$	$Time$	Mem	I	$\|\Pi\|$	$Time$	Mem	$Time$	Mem
CS$_1(5)$	36	405	16	1.13s	112MB	49	85	6.11s	213MB	32	33	**0.04s**	**34MB**	0.18s	95MB
CS$_1(6)$	49	1215	19	2.52s	220MB	65	123	11.41s	243MB	40	41	**0.04s**	**51MB**	0.31s	99MB
CS$_1(7)$	64	3645	22	5.41s	408MB	84	156	31.16s	867MB	56	57	**0.05s**	**82MB**	0.77s	113MB
CS$_n(3)$	125	16	54	0.65s	102MB	9	24	33.43s	258MB	11	12	**0.09s**	**35MB**	11.29s	115MB
CS$_n(4)$	625	25	189	6.22s	495MB	15	42	TO	-	-	-	**0.4s**	**106MB**	1349.6s	577MB
CS$_n(5)$	3k	36	648	117.06s	2818MB	24	60	TO	-	-	-	**2.56s**	**345MB**	TO	-
MER(3)	278	1728	11	**1.42s**	**143MB**	8	14	2.74s	189MB	6	7	1.96s	228MB	128.1s	548MB
MER(4)	465	21k	14	**4.63s**	**464MB**	13	22	10.81s	870MB	10	11	11.02s	1204MB	TO	-
MER(5)	700	250k	17	**29.23s**	**1603MB**	20	32	67s	2879MB	15	16	-	MO	MO	-
SN(1)	43	32	18	0.13s	38MB	3	6	0.28s	88MB	2	3	**0.04s**	**29MB**	3.51s	135MB
SN(2)	796	32	54	0.9s	117MB	3	6	66.09s	258MB	2	3	**0.38s**	**103MB**	3580.83s	1022MB
SN(3)	7k	32	162	4.99s	**408MB**	3	6	TO	-	-	-	4.99s	612MB	TO	-
SN(4)	52k	32	486	**34.09s**	**2448MB**	3	6	TO	-	-	-	44.47s	3409MB	TO	-
LE$(3,4)$	2	652	256	**0.24s**	**70MB**	6	14	1.63s	223MB	6	7	0.38s	103MB	TO	-
LE$(3,5)$	2	1280	500	**0.31s**	**87MB**	6	14	Error	-	-	-	1.77s	253MB	Error	-
LE$(4,4)$	3	3160	1280	**0.61s**	**106MB**	6	16	TO	-	-	-	9.34s	1067MB	TO	-
LE$(5,5)$	4	18k	12k	**3.37s**	**364MB**	6	18	TO	-	-	-	-	MO	TO	-
LE$(6,4)$	5	27k	20k	**6.37s**	**743MB**	6	20	TO	-	-	-	-	MO	TO	-
LE$(6,5)$	5	107k	78k	**23.72s**	**2192MB**	6	20	TO	-	-	-	-	MO	TO	-
PETP(2)	68	3	3	0.04s	31MB	0	2	0.04s	87MB	0	1	0.04s	**30MB**	0.04s	90MB
PETP(3)	4	1730	4	**0.19s**	**65MB**	6	8	0.29s	153MB	3	4	0.24s	72MB	1.07s	170MB
PETP(4)	5	54k	5	**1.58s**	**325MB**	8	10	3.12s	727MB	4	5	7.04s	960MB	31.52s	1741MB

Summary of results. For all examples, other than the Client-Server protocol, the assume-guarantee method scales better than the monolithic reasoning; and in all examples our qualitative analysis scales better than the strong simulation approach.

Two-player Games Examples. We also experimented with our tool on several examples of games, where one of the players controls the choices of the system and the other player represents the environment.

- EC is based on [10] and models an error-correcting device that sends and receives data blocks over a communication channel. Notation EC(n, k, d) means that a data block consists of n bits and it encodes k bits of data; value d is the minimum Hamming distance between two distinct blocks. In the first component Player 2 chooses a message to be sent over the channel and is allowed to flip some bits in the block. The second component restricts the number of bits that Player 2 can flip. The specification requires that every message is correctly decoded.
- PETG is the Peterson's algorithm [57] example for MDPs, with the following differences: (a) the system may choose to restart instead of entering the critical section; (b) instead of a randomized scheduler we consider an adversarial scheduler. As before, the specification requires mutual exclusion.
- VIR1 models a virus that attacks a computer system with n nodes (based on case study from PRISM [52]). Player 1 represents the virus and is trying to infect as

Table 2. Results for two-player games examples

				AGCS				MONCS		AGAS				MONAS	
Ex.	$\|G_1\|$	$\|G_2\|$	$\|G'\|$	*Time*	*Mem*	I	$\|\Pi\|$	*Time*	*Mem*	*Time*	*Mem*	I	$\|\Pi\|$	*Time*	*Mem*
EC$(32,6,16)$	71k	193	129	3.55s	446MB	1	7	**1.15s**	281MB	2.34s	391MB	0	2	1.03s	**251MB**
EC$(64,7,16)$	549k	385	257	70.5s	3704MB	1	131	9.07s	1725MB	16.79s	1812MB	0	2	**4.83s**	**1467MB**
EC$(64,8,16)$	1.1m	769	513	-	MO	-	-	-	MO	**52.63s**	**3619MB**	0	2	-	MO
EC$(64,8,32)$	1.1m	1025	513	-	MO	-	-	-	MO	**54.08s**	**3665MB**	0	2	-	MO
PETG(2)	3	52	3	0.08s	35MB	4	6	0.03s	30MB	0.07s	35MB	4	6	0.03s	**29MB**
PETG(3)	4	1514	4	**0.2s**	63MB	6	8	0.25s	74MB	0.22s	**62MB**	6	8	0.21s	64MB
PETG(4)	5	49k	5	1.75s	316MB	8	10	8.16s	1080MB	**1.6s**	**311MB**	8	10	6.94s	939MB
VIR1(12)	14	4097	1	0.91s	159MB	15	30	1.69s	255MB	**0.35s**	**114MB**	2	4	1.53s	215MB
VIR1(13)	15	8193	1	1.47s	197MB	16	32	4.36s	601MB	**0.6s**	**178MB**	2	4	2.8s	402MB
VIR1(14)	16	16k	1	3.09s	326MB	17	34	8.22s	992MB	**0.75s**	**241MB**	2	4	6.49s	816MB
VIR1(15)	17	32k	1	4.47s	643MB	18	36	15.13s	2047MB	**1.05s**	**490MB**	2	4	9.67s	1361MB
VIR1(16)	18	65k	1	8.65s	1015MB	19	38	41.28s	3785MB	**1.37s**	**839MB**	2	4	23.71s	2591MB
VIR1(17)	19	131k	1	18.68s	1803MB	20	40	-	MO	**2.12s**	**1653MB**	2	4	62.24s	4309MB
VIR1(18)	20	262k	1	38.68s	3079MB	21	42	-	MO	**3.35s**	**2878MB**	2	4	-	MO
VIR2(12)	13	4096	1	1.02s	151MB	19	34	0.81	154MB	0.68s	**122MB**	9	14	**0.57s**	133MB
VIR2(13)	14	8192	1	1.48s	190MB	20	36	1.13s	216MB	1.01s	**183MB**	9	14	1.01s	208MB
VIR2(14)	15	16k	1	2.9s	315MB	21	38	2.33s	389MB	**1.94s**	**311MB**	9	14	2.09s	388MB
VIR2(15)	16	32k	1	5s	631MB	22	40	6.29s	964MB	**2.12s**	**489MB**	9	14	4.69s	757MB
VIR2(16)	17	65k	1	9.82s	949MB	23	42	7.55s	1468MB	**3.96s**	**897MB**	9	14	6.09s	1315MB
VIR2(17)	18	131k	1	23.33s	1815MB	24	44	23.54s	3012MB	**8.16s**	**1676MB**	9	14	15.36s	2542MB
VIR2(18)	19	262k	1	45.89s	3049MB	25	46	55.28s	4288MB	**20.3s**	**2875MB**	9	14	28.79s	3755MB

many nodes of the network as possible. Player 2 represents the system and may recover an infected node to an uninfected state. The specification requires that the virus has a strategy to avoid being completely erased, i.e., maintain at least one infected node in the network. VIR2 is a modified version of VIR1 with two special critical nodes in the network. Whenever both of the nodes are infected, the virus can overtake the system. The specification is as for VIR1, i.e., the virus can play such that at least one node in the network remains infected, but it additionally requires that even if the system cooperates with the virus, the system is designed in a way that the special nodes will never be infected at the same time.

The results for two-player game examples are shown in Table 2. Along with AGCS and MONCS for assume-guarantee and monolithic combined simulation, we also consider AGAS and MONAS for assume-guarantee and monolithic alternating simulation, as for properties in 1-ATL it suffices to consider only alternating simulation. For all the examples, the assume-guarantee algorithms scale better than the monolithic ones. Combined simulation is finer than alternating simulation and therefore combined simulation may require more CEGAR iterations.

Concluding Remarks. In this work we considered compositional analysis of MDPs for qualitative properties and presented a CEGAR approach. Our algorithms are discrete graph theoretic algorithms. An interesting direction of future work would be to consider symbolic approaches to the problem.

Acknowledgements. We thank Anvesh Komuravelli for sharing his implementation.

References

1. Alur, R., Henzinger, T., Kupferman, O., Vardi, M.: Alternating refinement relations. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 163–178. Springer, Heidelberg (1998)
2. Alur, R., Henzinger, T.A.: Computer-aided verification (2004) (unpublished), `http://www.cis.upenn.edu/cis673/`
3. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM 49(5), 672–713 (2002)
4. Aziz, A., Singhal, V., Balarin, F., Brayton, R., Sangiovanni-Vincentelli, A.: It usually works: The temporal logic of stochastic systems. In: Wolper, P. (ed.) CAV 1995. LNCS, vol. 939, pp. 155–165. Springer, Heidelberg (1995)
5. Baier, C., Bertrand, N., Größer, M.: On decision problems for probabilistic büchi automata. In: Amadio, R.M. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 287–301. Springer, Heidelberg (2008)
6. Baier, C., Katoen, J.-P.: Principles of model checking. MIT Press (2008)
7. Beeri, C.: On the membership problem for functional and multivalued dependencies in relational databases. ACM Trans. on Database Systems 5, 241–259 (1980)
8. Bertrand, N., Genest, B., Gimbert, H.: Qualitative determinacy and decidability of stochastic games with signals. In: Proc. of LICS, pp. 319–328. IEEE Computer Society (2009)
9. Bianco, A., de Alfaro, L.: Model checking of probabalistic and nondeterministic systems. In: Thiagarajan, P.S. (ed.) FSTTCS 1995. LNCS, vol. 1026, pp. 499–513. Springer, Heidelberg (1995)
10. Cerný, P., Chmelik, M., Henzinger, T.A., Radhakrishna, A.: Interface simulation distances. In: GandALF, EPTCS 96, pp. 29–42 (2012)
11. Chadha, R., Viswanathan, M.: A counterexample-guided abstraction-refinement framework for Markov decision processes. ACM Trans. Comput. Log. 12, 1 (2010)
12. Chaki, S., Clarke, E.M., Sinha, N., Thati, P.: Automated assume-guarantee reasoning for simulation conformance. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 534–547. Springer, Heidelberg (2005)
13. Chatterjee, K.: Stochastic ω-Regular Games. PhD thesis, UC Berkeley (2007)
14. Chatterjee, K., Chaubal, S., Kamath, P.: Faster algorithms for alternating refinement relations. In: CSL. LIPIcs, vol. 16, pp. 167–182. Schloss Dagstuhl (2012)
15. Chatterjee, K., Chmelík, M., Daca, P.: CEGAR for qualitative analysis of probabilistic systems. CoRR, abs/1405.0835 (2014)
16. Chatterjee, K., Chmelik, M., Tracol, M.: What is decidable about partially observable Markov decision processes with omega-regular objectives. In: Proceedings of CSL 2013: Computer Science Logic (2013)
17. Chatterjee, K., de Alfaro, L., Faella, M., Majumdar, R., Raman, V.: Code-aware resource management. Formal Methods in System Design 42(2), 146–174 (2013)
18. Chatterjee, K., de Alfaro, L., Henzinger, T.A.: Qualitative concurrent parity games. ACM Trans. Comput. Log. 12(4), 28 (2011)
19. Chatterjee, K., Doyen, L.: Partial-observation stochastic games: How to win when belief fails. In: Proceedings of LICS 2012: Logic in Computer Science, pp. 175–184. IEEE Computer Society Press (2012)
20. Chatterjee, K., Doyen, L., Henzinger, T.A.: Qualitative analysis of partially-observable markov decision processes. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 258–269. Springer, Heidelberg (2010)
21. Chatterjee, K., Doyen, L., Henzinger, T.A.: A survey of partial-observation stochastic parity games. Formal Methods in System Design 43(2), 268–284 (2013)

22. Chatterjee, K., Doyen, L., Henzinger, T.A., Raskin, J.-F.: Algorithms for omega-regular games with imperfect information'. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 287–302. Springer, Heidelberg (2006)
23. Chatterjee, K., Doyen, L., Nain, S., Vardi, M.Y.: The complexity of partial-observation stochastic parity games with finite-memory strategies. In: Muscholl, A. (ed.) FOSSACS 2014. LNCS, vol. 8412, pp. 242–257. Springer, Heidelberg (2014)
24. Chatterjee, K., Henzinger, M.: Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification. In: SODA, pp. 1318–1336 (2011)
25. Chatterjee, K., Henzinger, M.: An $O(n^2)$ time algorithm for alternating Büchi games. In: SODA, pp. 1386–1399 (2012)
26. Chatterjee, K., Henzinger, M., Joglekar, M., Shah, N.: Symbolic algorithms for qualitative analysis of Markov decision processes with Büchi objectives. Formal Methods in System Design 42(3), 301–327 (2013)
27. Chatterjee, K., Jurdziński, M., Henzinger, T.A.: Simple stochastic parity games. In: Baaz, M., Makowsky, J.A. (eds.) CSL 2003. LNCS, vol. 2803, pp. 100–113. Springer, Heidelberg (2003)
28. Chatterjee, K., Tracol, M.: Decidable problems for probabilistic automata on infinite words. In: LICS, pp. 185–194 (2012)
29. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (1999)
30. Clarke, E.M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 154–169. Springer, Heidelberg (2000)
31. Cleaveland, R., Steffen, B.: Computing behavioural relations, logically. In: Leach Albert, J., Monien, B., Rodríguez-Artalejo, M. (eds.) ICALP 1991. LNCS, vol. 510, pp. 127–138. Springer, Heidelberg (1991)
32. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. J. ACM 42(4), 857–907 (1995)
33. D'Argenio, P.R., Jeannet, B., Jensen, H.E., Larsen, K.G.: Reachability analysis of probabilistic systems by successive refinements. In: de Luca, L., Gilmore, S. (eds.) PAPM-PROBMIV 2001. LNCS, vol. 2165, pp. 39–56. Springer, Heidelberg (2001)
34. D'Argenio, P.R.: Reduction and refinement strategies for probabilistic analysis. In: Hermanns, H., Segala, R. (eds.) PAPM-PROBMIV 2002. LNCS, vol. 2399, pp. 57–76. Springer, Heidelberg (2002)
35. de Alfaro, L., Henzinger, T.A., Jhala, R.: Compositional methods for probabilistic systems. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 351–365. Springer, Heidelberg (2001)
36. de Alfaro, L., Henzinger, T.A., Kupferman, O.: Concurrent reachability games. In: FOCS, pp. 564–575 (1998)
37. Etessami, K., Kwiatkowska, M.Z., Vardi, M.Y., Yannakakis, M.: Multi-objective model checking of Markov decision processes. Logical Methods in Computer Science 4(4) (2008)
38. Feng, L., Kwiatkowska, M.Z., Parker, D.: Automated learning of probabilistic assumptions for compositional reasoning. In: Giannakopoulou, D., Orejas, F. (eds.) FASE 2011. LNCS, vol. 6603, pp. 2–17. Springer, Heidelberg (2011)
39. Filar, J., Vrieze, K.: Competitive Markov Decision Processes. Springer (1997)
40. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, logics, and infinite games: A guide to current research. LNCS, vol. 2500. Springer, Heidelberg (2002)
41. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Asp. Comput. 6(5), 512–535 (1994)

42. Henzinger, M.R., Henzinger, T.A., Kopke, P.W.: Computing simulations on finite and infinite graphs. In: FOCS, pp. 453–462 (1995)
43. Henzinger, T.A., Jhala, R., Majumdar, R.: Counterexample-guided control. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 886–902. Springer, Heidelberg (2003)
44. Henzinger, T.A., Jhala, R., Majumdar, R., Qadeer, S.: Thread-modular abstraction refinement. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 262–274. Springer, Heidelberg (2003)
45. Hermanns, H., Wachter, B., Zhang, L.: Probabilistic CEGAR. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 162–175. Springer, Heidelberg (2008)
46. Howard, R.A.: Dynamic Programming and Markov Processes. MIT Press (1960)
47. Immerman, N.: Number of quantifiers is better than number of tape cells. Journal of Computer and System Sciences 22, 384–406 (1981)
48. Itai, A., Rodeh, M.: Symmetry breaking in distributed networks. Information and Computation 88(1) (1990)
49. Jeannet, B., dArgenio, P., Larsen., K.: Rapture: A tool for verifying Markov decision processes. Tools Day 2, 149 (2002)
50. Komuravelli, A., Păsăreanu, C.S., Clarke, E.M.: Assume-guarantee abstraction refinement for probabilistic systems. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 310–326. Springer, Heidelberg (2012)
51. Kwiatkowska, M.Z., Norman, G., Parker, D.: Game-based abstraction for Markov decision processes. In: QEST, pp. 157–166 (2006)
52. Kwiatkowska, M.Z., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
53. Kwiatkowska, M.Z., Norman, G., Parker, D., Qu, H.: Assume-guarantee verification for probabilistic systems. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 23–37. Springer, Heidelberg (2010)
54. Milner, R.: An algebraic definition of simulation between programs. IJCAI, 481–489 (1971)
55. Nain, S., Vardi, M.Y.: Solving partial-information stochastic parity games. In: LICS, pp. 341–348 (2013)
56. Pasareanu, C.S., Giannakopoulou, D., Bobaru, M.G., Cobleigh, J.M., Barringer, H.: Learning to divide and conquer: applying the l* algorithm to automate assume-guarantee reasoning. Formal Methods in System Design 32(3), 175–205 (2008)
57. Peterson, G.L.: Myths about the mutual exclusion problem. Information Processing Letters 12(3), 115–116 (1981)
58. Pnueli, A.: In: transition from global to modular temporal reasoning about programs. In: Logics and Models of Concurrent Systems, NATO Advanced Summer Institutes F-13, pp. 123–144. Springer (1985)
59. Pogosyants, A., Segala, R., Lynch, N.: Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. Distributed Computing 13(3), 155–186 (2000)
60. Segala, R.: Modeling and Verification of Randomized Distributed Real-Time Systems. PhD thesis, MIT Press, Technical Report MIT/LCS/TR-676 (1995)
61. Segala, R., Lynch, N.A.: Probabilistic simulations for probabilistic processes. Nord. J. Comput. 2(2), 250–273 (1995)
62. Stoelinga, M.: Fun with FireWire: Experiments with verifying the IEEE1394 root contention protocol. In: Formal Aspects of Computing (2002)