# 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 counter-example 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 non-deterministic behavior [30,23]. In verification of probabilistic systems, MDPs have been adopted as models for concurrent probabilistic systems [18], probabilistic systems operating in open environments [42], under-specified probabilistic systems [7], and applied in diverse domains [5,35] 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 [16], 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 [40], 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 $A$ 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 *counter-example guided abstraction-refinement* (CEGAR) loop [17]. There are many symmetric [38] as well as circular compositional reasoning [19,38,36] rules; however the simple asymmetric

rule is most effective in practice and extensively studied, mostly for non-probabilistic systems [38,22,10,28].

**Compositional analysis for probabilistic systems.** There are many works that have studied the abstraction-refinement and compositional analysis for probabilistic systems [9,29,34,21]. Our work is most closely related to and inspired by [33] 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 [25,4,7].

**Qualitative analysis and its importance.** In this work we consider the fragment of pCTL$^*$ [25,4,7] that is relevant for *qualitative analysis*, and refer to this fragment as $\mathrm{QCTL}^*$. 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 [13]. Even in settings where it suffices to satisfy certain specifications with probability $\lambda < 1$, the correct choice of $\lambda$ 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., [41,44]). Furthermore, in contrast to quantitative analysis, qualitative analysis is robust to numerical perturbations and modeling errors in the transition probabilities.

**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 $\mathrm{QCTL}^*$ we consider the logic $\mathrm{ATL}^*$ 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* [37] and *alternating-simulation* [1]. We first show that the logical relation induced by $\mathrm{QCTL}^*$ 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 $\mathrm{QCTL}^*$.
2. We show that our new notion of simulation, which captures the logic relation of $\mathrm{QCTL}^*$, 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 counter-example analysis and abstraction refinement is achieved using the ideas of [27] 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 [38,22,10,28]. Game-based abstraction refinement has been studied in the context of probabilistic systems [34]. The CEGAR approach has been adapted to probabilistic systems for reachability [29] and safe-pCTL [9] under monolithic (non-compositional) abstraction refinement. The work of [33] considers CEGAR for compositional analysis of probabilistic system with strong simulation. 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.

**Organization of the paper.** In Section 2 we present the basic definitions of games and logic for games. In Section 3 we introduce a new simulation relation for games, show that it is finer than both simulation and alternating simulation, and present algorithms to compute the relation. In Section 4 we present the definitions of MDPs and qualitative logics, and in Section 5 show that the logical relation induced by the qualitative logics on MDPs can be obtained through our simulation relation introduced in Section 3. In Section 6 we present a CEGAR approach for our simulation relation and present experimental results in Section 7.

## 2 Game Graphs and Alternating-time Temporal Logics

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

– $s_0 \in S$ is an initial state.

**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 every Player-1 state $s$ we have $\mathsf{turn} \in \mathcal{L}(s)$, and for every Player 2 state $s'$ 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 $\Omega$ 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 $\Sigma$ 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 $\Theta$ 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 functions $S \times A \to S$ for Player 2).

**Outcomes.** Given a strategy $\sigma$ for Player 1 and $\theta$ 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 $a_i = \sigma(s_0 a_0 \dots s_i)$ and $s_{i+1} = \theta(s_0 a_0 \dots s_i a_i)$. Given a state $s \in S$ we denote by $\mathsf{Plays}(s, \sigma)$ (resp. $\mathsf{Plays}(s, \theta)$) the set of possible plays given $\sigma$ (resp. $\theta$), 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:

– The states of the composition are $\overline{S} = S \times S'$.
– The set of actions does not change with the composition.
– For all $(s, s')$ we have $\overline{\mathsf{Av}}((s, s')) = \mathsf{Av}(s) \cap \mathsf{Av}'(s')$.
– The transition function for a state $(s, s') \in \overline{S}$ and an action $a \in \overline{\mathsf{Av}}((s, s'))$ is defined as $\overline{\delta}((s, s'), a) = \{(t, t') \mid t \in \delta(s, a) \wedge t' \in \delta'(s', a)\}$.
– The labeling function $\overline{\mathcal{L}}((s, s'))$ is defined as $\mathcal{L}(s) \cup \mathcal{L}'(s')$.
– 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. The definition of 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 ($\mathrm{ATL}^*$) [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* ($\varphi$) and *state formulas* ($\psi$) according to the following grammar:

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

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 $\mathsf{true}$ as a shorthand for $q \vee \neg q$ and $\mathsf{false}$ for $q \wedge \neg q$ for some $q \in \mathsf{AP}$. The path quantifiers PQ are as follows:

$$\mathrm{ATL}^* \text{ path quantifiers: } \langle\langle 1 \rangle\rangle, \langle\langle 2 \rangle\rangle, \langle\langle 1, 2 \rangle\rangle, \text{ and } \langle\langle \emptyset \rangle\rangle.$$

**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 $\omega$, i.e., $\omega[i] = s_i a_i s_{i+1} a_{i+1} \cdots$. The semantics of path formulas is defined inductively as follows:

$$\begin{array}{ll} \omega \models \psi & \text{iff } \omega[0] \models \psi \\ \omega \models \varphi_1 \vee \varphi_2 & \text{iff } \omega \models \varphi_1 \text{ or } \omega \models \varphi_2 \\ \omega \models \varphi_1 \wedge \varphi_2 & \text{iff } \omega \models \varphi_1 \text{ and } \omega \models \varphi_2 \\ \omega \models \bigcirc \varphi & \text{iff } \omega[1] \models \varphi \\ \omega \models \varphi_1 \mathcal{U} \varphi_2 & \text{iff } \exists j \in \mathbb{N} : \omega[j] \models \varphi_2 \text{ and } \forall 0 \leq i < j : \omega[i] \models \varphi_1 \\ \omega \models \varphi_1 \mathcal{W} \varphi_2 & \text{iff } \varphi_1 \mathcal{U} \varphi_2 \text{ or } \forall j \in \mathbb{N} : \omega[j] \models \varphi_1. \end{array}$$

Given a path formula $\varphi$, we denote by $[\![\varphi]\!]_G$ the set of plays $\omega$ such that $\omega \models \varphi$. We omit the $G$ lower script when the game is clear from context. The semantics of state formulas for $\mathrm{ATL}^*$ is defined as follows:

$$\begin{array}{ll} s \models q & \text{iff } q \in \mathcal{L}(s) \\ s \models \neg q & \text{iff } q \notin \mathcal{L}(s) \\ s \models \psi_1 \vee \psi_2 & \text{iff } s \models \psi_1 \text{ or } s \models \psi_2 \\ s \models \psi_1 \wedge \psi_2 & \text{iff } s \models \psi_1 \text{ and } s \models \psi_2 \\ s \models \langle\langle 1 \rangle\rangle(\varphi) & \text{iff } \exists \sigma \in \Sigma, \forall \theta \in \Theta : \mathsf{Plays}(s, \sigma, \theta) \in [\![\varphi]\!] \\ s \models \langle\langle 2 \rangle\rangle(\varphi) & \text{iff } \exists \theta \in \Theta, \forall \sigma \in \Sigma : \mathsf{Plays}(s, \sigma, \theta) \in [\![\varphi]\!] \\ s \models \langle\langle 1, 2 \rangle\rangle(\varphi) & \text{iff } \exists \sigma \in \Sigma, \exists \theta \in \Theta : \mathsf{Plays}(s, \sigma, \theta) \in [\![\varphi]\!] \\ s \models \langle\langle \emptyset \rangle\rangle(\varphi) & \text{iff } \forall \sigma \in \Sigma, \forall \theta \in \Theta : \mathsf{Plays}(s, \sigma, \theta) \in [\![\varphi]\!]; \end{array}$$

where $s \in S$ and $q \in \mathsf{AP}$. Given an $\mathrm{ATL}^*$ state formula $\psi$ 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 $\psi$. We omit the $G$ lower script when the game is clear from context.

**Logic fragments.** We define several fragments of the logic $\mathrm{ATL}^*$:

– *Restricted temporal operator use.* An important fragment of $\mathrm{ATL}^*$ 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 [37] 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 [37,2].

**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 $\preccurlyeq_C$.

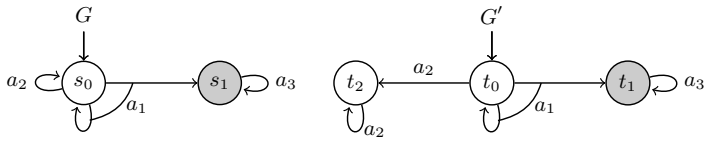


**Fig. 1.** Games $G, G'$ such that $G \sim_{\mathcal{S}} G'$ and $G \sim_{\mathcal{A}} G'$, but $G \not\sim_{\mathcal{C}} G'$.

*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 an atomic proposition $q$. The largest simulation and alternating-simulation relations between $G$ and $G'$ are: $\mathcal{S}_{\max} =$

$\{(s_0,t_0),(s_1,t_1)\}, \mathcal{A}_{\max} = \{(s_0,t_0),(s_0,t_2),(s_1,t_1)\}$. 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 [26,1,11]. We now define a two-player game 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. The step-wise simulation condition is checked by playing a two-step game, and the step-wise alternating-simulation condition is checked by playing a four-step game. Consider two games $G = (S, A, \mathsf{Av}, \delta, \mathcal{L}, s_0)$ and $G' = (S', A', \mathsf{Av}', \delta', \mathcal{L}', s'_0)$. We construct the *combined-simulation game* $G^{\mathcal{C}} = (S^{\mathcal{C}}, A^{\mathcal{C}}, \mathsf{Av}^{\mathcal{C}}, \delta^{\mathcal{C}}, \mathcal{L}^{\mathcal{C}}, s_0^{\mathcal{C}})$ as follows:

- *The set of states.* The set of states $S^{\mathcal{C}}$ is:

$$\begin{aligned} S^{\mathcal{C}} = \quad & (S \times S') \cup (S \times S' \times \{\mathsf{Sim}\} \times \{1,2\}) \cup (S \times S' \times \{\mathsf{Alt}\} \times \{2\}) \\ & \cup (S \times S' \times \{\mathsf{Alt}\} \times A \times \{1\}) \cup (S \times S' \times \{\mathsf{Alt}\} \times A \times A' \times \{1,2\}) \end{aligned}$$

  Intuitively, in states in $S \times S'$ and in states where the last component is 2 it is Player 2's turn to make the choice of successors, and in all other states Player 1 makes the choice of actions.
- *The set of actions.* The set of actions is as follows: $A^{\mathcal{C}} = \{\bot\} \cup S \cup S' \cup A'$.
- *The transition function and the action-available function.*
  1. *Choice of simulation or alternating-simulation.* For a state $(s,s')$ we have only one action $\bot$ available for Player 1 and we have $\delta^{\mathcal{C}}((s,s'),\bot) = \{(s,s',\mathsf{Alt},2),(s,s',\mathsf{Sim},2)\}$, i.e., Player 2 decides whether to check for step-wise simulation or step-wise alternating-simulation conditions.
  2. *Checking step-wise simulation conditions.* We describe the transitions for checking the simulation conditions:
     (a) For a state $(s,s',\mathsf{Sim},2)$ we have only one action $\bot$ available for Player 1 and we have $\delta^{\mathcal{C}}((s,s',\mathsf{Sim},2),\bot) = \{(t,s',\mathsf{Sim},1) \mid \exists a \in \mathsf{Av}(s) : \ t \in \delta(s,a)\}$.
     (b) For a state $\overline{s} = (t,s',\mathsf{Sim},1)$ we have $\mathsf{Av}^{\mathcal{C}}(\overline{s}) = \{t' \mid \exists a' \in \mathsf{Av}(s') : \ t' \in \delta'(s',a')\}$ and $\delta^{\mathcal{C}}(\overline{s},t') = \{(t,t')\}$.

     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 $t' \in \delta'(s',a')$.
  3. *Checking step-wise alternating-simulation conditions.* We describe the transitions for checking the alternating-simulation conditions:
     (a) For a state $(s,s',\mathsf{Alt},2)$ we have only one action $\bot$ available for Player 1 and we have $\delta^{\mathcal{C}}((s,s',\mathsf{Alt},2),\bot) = \{(s,s',\mathsf{Alt},a,1) \mid a \in \mathsf{Av}(s)\}$.
     (b) For a state $\overline{s} = (s,s',\mathsf{Alt},a,1)$ we have $\mathsf{Av}^{\mathcal{C}}(\overline{s}) = \mathsf{Av}'(s')$ and $\delta^{\mathcal{C}}(\overline{s},a') = \{(s,s',\mathsf{Alt},a,a',2)\}$.

(c) For a state $(s, s', \mathsf{Alt}, a, a', 2)$ we have only one action $\bot$ available for Player 1 and we have $\delta^{\mathcal{C}}((s, s', \mathsf{Alt}, a, a', 2), \bot) = \{(s, t', \mathsf{Alt}, a, a', 1) \mid t' \in \delta'(s', a')\}$.

(d) For a state $\overline{s} = (s, t', \mathsf{Alt}, a, a', 1)$ we have $\mathsf{Av}^{\mathcal{C}}(\overline{s}) = \delta(s, a)$ and $\delta^{\mathcal{C}}(\overline{s}, t) = \{(t, t')\}$.

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 $t'$ from $\delta'(s', a')$ and Player 1 responds by choosing a successor $t$ in $\delta(s, a)$.

– *The labeling function.* The set of atomic proposition $\mathsf{AP}$ contains a single proposition $p \in \mathsf{AP}$. The labeling function $\mathcal{L}^{\mathcal{C}}$ given a state $\overline{s} \in S^{\mathcal{C}}$ is defined as follows: $\mathcal{L}^{\mathcal{C}}(\overline{s}) = p$ iff $\overline{s} = (s, s')$ and $\mathcal{L}(s) \neq \mathcal{L}'(s')$. Intuitively, Player 2's goal is to reach a state $(s, s')$ where the propositional labeling of the original games do not match, i.e., to reach a state labeled $p$ by $\mathcal{L}^{\mathcal{C}}$.
– *The initial state.* The state $s_0^{\mathcal{C}}$ is $(s_0, s_0')$.

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).

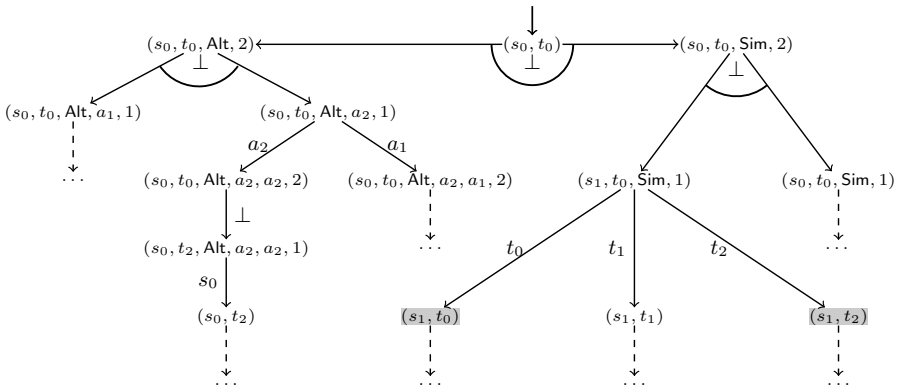


**Fig. 2.** Part of the combined-simulation game of $G$ and $G'$ from Figure 1.

*Example 2.* A part of the combined-simulation game of $G$ and $G'$ from Figure 1 is shown in Figure 2. Dashed arrows indicate that the successors of a given state are omitted in the figure. Gray states are labeled by an atomic proposition $p$, hence are the goal states for the adversary. □

**Shorthand for safety objectives.** We will use the following shorthand for *safety* objectives: $\square\, \varphi \equiv \varphi\, \mathcal{W}\, \mathsf{false}$; i.e., the formula $\square\varphi$ is satisfied by paths where $\varphi$ is always true.

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

*Proof.* The statement follows directly from the definition of combined simulation, and the fact that the game construction mimics the definition of combined simulation (as in the case of simulation and alternating simulation [26,1,11]). □

**Winning strategies.** Given a combined-simulation game $G^{\mathcal{C}}$ we say that a strategy $\sigma$ for the proponent is *winning* from a state $s$ if for all strategies $\theta$ of the adversary we have $\mathsf{Plays}(s, \sigma, \theta) \models \Box(\neg p)$. A strategy $\theta$ for the adversary is *winning* from state $s$ if for all strategies $\sigma$ of the proponent we have $\mathsf{Plays}(s, \sigma, \theta) \models \mathsf{true}\mathcal{U}p$. Whenever the proponent (resp. adversary) has a winning strategy, the proponent (resp. adversary) also has memoryless winning strategy [24].

**Combined simulation logical characterization.** Our next goal is to establish that combined simulation gives the logical characterization of C-ATL$^*$ and C-ATL. To prove the result we first introduce the notion of equivalence between plays: Given two plays $\omega = s_0 a_0 s_1 a_1 s_2 \cdots$ and $\omega' = s'_0 a'_0 s'_1 a'_1 s'_2 \cdots$ we write $\omega \sim_{\mathcal{C}} \omega'$ if for all $i \geq 0$ we have $(s_i, s'_i) \in \mathcal{C}_{\max}$.

**Lemma 1.** *Given two games $G$ and $G'$, let $\mathcal{C}_{\max}$ be the combined simulation. For all $(s, s') \in \mathcal{C}_{\max}$ the following assertions hold:*

- *For all Player 1 strategies $\sigma$ in $G$, there exists a Player 1 strategy $\sigma'$ in $G'$ such that for every play $\omega' \in \mathsf{Plays}(s', \sigma')$ there exists a play $\omega \in \mathsf{Plays}(s, \sigma)$ such that $\omega \sim_{\mathcal{C}} \omega'$.*
- *For all pair of strategies $\sigma$ and $\theta$ in $G$, there exists a pair of strategies $\sigma'$ and $\theta'$ in $G'$ such that $\mathsf{Plays}(s, \sigma, \theta) \sim_{\mathcal{C}} \mathsf{Plays}(s', \sigma', \theta')$,*

*Proof.* We present the details of the first item.

- Consider a winning strategy $\sigma^{\mathcal{C}}$ for the proponent in $G^{\mathcal{C}}$ such that for all $(s, s') \in \mathcal{C}_{\max}$ and against all strategies $\theta^{\mathcal{C}}$ we have $\mathsf{Plays}(s, \sigma^{\mathcal{C}}, \theta^{\mathcal{C}}) \in [\![\Box(\neg p)]\!]$. Given the Player 1 strategy $\sigma$ in $G$ we construct $\sigma'$ in $G'$ using the strategy $\sigma^{\mathcal{C}}$. Consider a history $w \cdot s$ in $G$ and $w' \cdot s' \in G'$ such that $(s, s') \in \mathcal{C}_{\max}$. Let $\sigma(w \cdot s) = a$. We define $\sigma'(w' \cdot s')$ as follows. Let $h$ be an arbitrary history in $G^{\mathcal{C}}$ that only visits state in $\mathcal{C}_{\max}$ and ends in $(s, s')$. Let $a' = \sigma^{\mathcal{C}}(h \cdot (s, s', \mathsf{Alt}, 2) \cdot (s, s', \mathsf{Alt}, a, 2))$; (i.e., the action played by the strategy $\sigma^{\mathcal{C}}$ in response to the choice of checking alternating simulation and the action $a$ by Player 2 in $G^{\mathcal{C}}$). Then the strategy $\sigma'$ plays accordingly, i.e., $\sigma'(w' \cdot s') = a'$. In the next step for every choice $t'$ of the adversary there exists a choice $t$ of the proponent such that $\mathcal{L}(t) = \mathcal{L}'(t')$ and $(t, t') \in \mathcal{C}_{\max}$ and the matching can proceed.
- The proof is similar to the first item, and instead of using the step-wise alternating-simulation gadget for strategy construction (of the first item) we use the step-wise simulation gadget from $G^{\mathcal{C}}$ to construct the strategy pairs.

The desired result follows. □

In the following theorem we establish the relation between combined simulation and the C-ATL$^*$ fragment of ATL$^*$.

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

*Proof. First implication.* We first prove the implication $\mathcal{C}_{\max} \subseteq \preccurlyeq_C^*$. We will show the following assertions:

- For all states $s$ and $s'$ such that $(s, s') \in \mathcal{C}_{\max}$, we have that every C-ATL$^*$ state formula satisfied in $s$ is also satisfied in $s'$.
- For all plays $\omega$ and $\omega'$ such that $\omega \sim_{\mathcal{C}} \omega'$, we have that every C-ATL$^*$ path formula satisfied in $\omega$ is also satisfied in $\omega'$.

We will prove the theorem by induction on the structure of the formulas. The interesting cases for the induction step are formulas $\langle\langle 1 \rangle\rangle(\varphi)$ and $\langle\langle 1, 2 \rangle\rangle(\varphi)$, where $\varphi$ is a path formula.

- Assume $s \models \langle\langle 1 \rangle\rangle(\varphi)$ and $(s, s') \in \mathcal{C}_{\max}$. It follows that there exists a strategy $\sigma \in \Sigma$ that ensures the path formula $\varphi$ from state $s$ against any strategy $\theta \in \Theta$. We want to show that $s' \models \langle\langle 1 \rangle\rangle(\varphi)$. By Lemma 1(item 1) we have that there exists a strategy $\sigma'$ for Player 1 from $s'$ such that for every play $\omega' \in \mathsf{Plays}(s', \sigma')$ there exists a play $\omega \in \mathsf{Plays}(s, \sigma)$ such that $\omega \sim_{\mathcal{C}} \omega'$. By inductive hypothesis we have that $s' \models \langle\langle 1 \rangle\rangle(\varphi)$.
- Assume $s \models \langle\langle 1, 2 \rangle\rangle(\varphi)$ and $\mathcal{C}(s, s')$. It follows that there exist strategies $\sigma \in \Sigma, \theta \in \Theta$ that ensure the path formula $\varphi$ from state $s$. By Lemma 1(item 2) we have that there exist strategies $\sigma'$ and $\theta'$ such that the two plays $\omega' = \mathsf{Plays}(s', \sigma', \theta')$ and $\omega = \mathsf{Plays}(s, \sigma, \theta)$ satisfy $\omega \sim_{\mathcal{C}} \omega'$. By inductive hypothesis we have that $s' \models \langle\langle 1, 2 \rangle\rangle(\varphi)$.
- Consider a path formula $\varphi$. If $\omega \sim_{\mathcal{C}} \omega'$, then by inductive hypothesis for every subformula $\varphi'$ of $\varphi$ we have that if $\omega \models \varphi'$ then $\omega' \models \varphi'$. It follows that if $\omega \models \varphi$ then $\omega' \models \varphi$.

*Second implication.* It remains to prove the second implication $\preccurlyeq_C^* \subseteq \preccurlyeq_C \subseteq \mathcal{C}_{\max}$. Assume that given states $s$ and $s'$ we have that $(s, s') \notin \mathcal{C}_{\max}$, then there exists a winning strategy in the corresponding combined-simulation game for the adversary from state $(s, s')$, i.e., there exists a strategy $\theta^{\mathcal{C}}$ such that against all strategies $\sigma^{\mathcal{C}}$ we have $\mathsf{Plays}((s, s'), \sigma^{\mathcal{C}}, \theta^{\mathcal{C}})$ reaches a state labeled $p$. As memoryless strategies are sufficient for both players in $G^{\mathcal{C}}$ [24], there also exists a bound $i \in \mathbb{N}$, such that the proponent fails to match the choice of the adversary in at most $i$ turns. We sketch the inductive proof that there exists a formula with $i$ nested operators $\langle\langle 1 \rangle\rangle\bigcirc$ or $\langle\langle 1, 2 \rangle\rangle\bigcirc$ that is satisfied in $s$ but not in $s'$. For $i$ equal to 0 the states can be distinguished by atomic propositions. For the inductive step one can express the simulation turns by a $\langle\langle 1, 2 \rangle\rangle(\bigcirc \ldots)$ formula and alternating simulation turns by a $\langle\langle 1 \rangle\rangle(\bigcirc \ldots)$ formula. It follows that $(s, s') \notin \preccurlyeq_C$. The result follows. □

*Remark 2.* Lemma 1 and Theorem 2 also hold 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 (on the size of the game) using discrete graph theoretic algorithms [31,6].

**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 section 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 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 $\delta_P(s)$. 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 $\mathrm{ATL}^*$.

**$1\frac{1}{2}$-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 [24]. 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 $\Phi$ if the game starts from the state s and Player 1 follows the strategy $\sigma$.

**Two-player Interpretation.** The two-player interpretation corresponds to alternating two-player games introduced in Section 2, where the probabilistic aspect of the MDP is replaced by a second player. Formally, given an MDP $G = (S, (S_1, S_P), A, \mathsf{Av}, \delta_1, \delta_P, \mathcal{L}, s_0)$ we define an alternating two-player game $\widehat{G} = (\widehat{S}, \widehat{A}, \widehat{\mathsf{Av}}, \widehat{\delta}, \widehat{\mathcal{L}}, \widehat{s_0})$ as follows: (i) the states are $\widehat{S} = S_1 \cup S_P$; (ii) the set of actions contains a new action $\perp$ not present in $A$, i.e., $\widehat{A} = A \cup \{\perp\}$; (iii) the action-available function for states $s \in S_1$ is defined as $\widehat{\mathsf{Av}}(s) = \mathsf{Av}(s)$ and for states $s_p \in S_P$ as $\widehat{\mathsf{Av}}(s_p) = \{\perp\}$; (iv) for $s \in S_1$ and $a$ in $\widehat{\mathsf{Av}}(s)$ we have $\widehat{\delta}(s,a) = \{\delta_1(s,a)\}$, and for

$s_p \in S_P$ we have $\widehat{\delta}(s_p, \bot) = \mathrm{Supp}(\delta_p(s_p))$; (v) the labeling function for a Player-1 state $s$ is $\widehat{\mathcal{L}}(s) = \mathcal{L}(s) \cup \{\mathsf{turn}\}$ and for a Player-2 state $s'$ coincides with $\mathcal{L}(s')$; and (vi) the initial state is the same $\widehat{s}_0 = s_0$. Given an MDP $G$ we denote by $\widehat{G}$ the two-player interpretation of the MDP. Note that for all Player-1 states $s \in S_1$ we have $|\widehat{\delta}(s)| = 1$ and for all Player-2 states $s_p \in S_P$ we have $|\mathsf{Av}(s_p)| = 1$. Therefore for any MDP the corresponding two-player interpretation is an alternating game.

**Parallel composition of MDPs.** An MDP is said to be strictly alternating if the initial state is a Player-1 state and all the successors of Player-1 states are probabilistic states, and vice versa. Given two strictly alternating MDPs $G = (S, (S_1, S_P), A, \mathsf{Av}, \delta_1, \delta_P, \mathcal{L}, s_0)$ and $G' = (S', (S'_1, S'_P), A, \mathsf{Av}', \delta'_1, \delta'_P, \mathcal{L}', s'_0)$, the parallel composition is an MDP $G \parallel G' = (\overline{S}, (\overline{S}_1, \overline{S}_P), A, \overline{\mathsf{Av}}, \overline{\delta}_1, \overline{\delta}_P, \overline{\mathcal{L}}, \overline{s}_0)$ defined as follows: (i) the states are $\overline{S} = \overline{S}_1 \cup \overline{S}_P$, where $\overline{S}_1 = S_1 \times S'_1$ and $\overline{S}_P = S_P \times S'_P$; (ii) for a state $(s, s') \in \overline{S}_1$ we have $\overline{\mathsf{Av}}((s, s')) = \mathsf{Av}(s) \cap \mathsf{Av}'(s')$; (iii) for a state $(s, s') \in \overline{S}_1$ and an action $a \in \overline{\mathsf{Av}}((s, s'))$ we have $\overline{\delta}_1((s, s'), a) = (\delta_1(s, a), \delta'_1(s', a))$; (iv) for a state $(s_p, s'_p) \in \overline{S}_P$ we have $\overline{\delta}((s_p, s'_p))(t, t') = \delta_P(s_p)(t) \cdot \delta'_P(s'_p)(t')$; (v) for a state $(s, s') \in \overline{S}$ we have $\overline{\mathcal{L}}((s, s')) = \mathcal{L}(s) \cup \mathcal{L}'(s')$, and (vi) the initial state is $(s_0, s'_0)$.

*Example 3.* In Figure 3 we present three MDPs $G_1, G_2$, and $G'$ that we use as running examples. We thoroughly describe only MDP $G' = (S, (S_1, S_P), A, \mathsf{Av}, \delta_1, \delta_P, \mathcal{L}, s_0)$. Player-1 states, depicted as circles, are $S_1 = \{s'_0, s'_2, s'_3\}$ and probabilistic states, depicted as rectangles, are $S_P = \{s'_1, s'_4\}$. The set of actions is $A = \{a, b\}$. Action $a$ is available in states $s'_0, s'_2$ and action $b$ is available only in states $s'_0, s'_3$. The deterministic transition function is $\delta_1(s'_0, a) = s'_1, \delta_1(s'_0, b) = s'_4, \delta_1(s'_2, a) = s'_4, \delta_1(s'_2, b) = s'_4, \delta_1(s'_3, b) = s'_4$. The probabilistic transition function $\delta_P$ gives the following probability distributions over possible successor states: $\delta_P(s'_1)(s'_2) = \frac{1}{2}, \delta_P(s'_1)(s'_3) = \frac{1}{2}, \delta_P(s'_4)(s'_3) = 1$. There is a single atomic proposition $p \in \mathsf{AP}$ and the states labeled by $p$ are depicted in gray. The initial state is $s'_0$. □

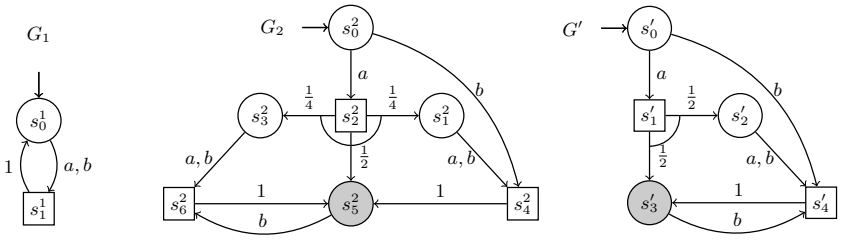


**Fig. 3.** Examples of MDPs.

### 4.2 Qualitative Logics for MDPs

We consider the qualitative fragment of $\mathrm{pCTL}^*$ [25,4,7] and refer to the logic as *qualitative pCTL*$^*$ (denoted as $\mathrm{QCTL}^*$) 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 $\mathrm{ATL}^*$. It has the same state and path formulas as $\mathrm{ATL}^*$ with the exception of path quantifiers. The logic $\mathrm{QCTL}^*$ 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 $\langle\langle\emptyset\rangle\rangle$).

$$\mathrm{QCTL}^* \text{ path quantifiers: } \langle\mathsf{Almost}\rangle, \langle\mathsf{Positive}\rangle.$$

The semantics of the logic $\mathrm{QCTL}^*$ is the same for the fragment shared with $\mathrm{ATL}^*$, therefore we only give semantics for the new path quantifiers. Given a path formula $\varphi$, we denote by $[\![\varphi]\!]_G$ the set of plays $\omega$ such that $\omega \models \varphi$. For a state $s$ and a path formula $\varphi$ we have:

$$\begin{aligned} s \models \langle\mathsf{Almost}\rangle(\varphi) \qquad & \text{iff } \exists \sigma \in \Sigma : \Pr_s^\sigma([\![\varphi]\!]) = 1 \\ s \models \langle\mathsf{Positive}\rangle(\varphi) \qquad & \text{iff } \exists \sigma \in \Sigma : \Pr_s^\sigma([\![\varphi]\!]) > 0. \end{aligned}$$

As before, we denote by QCTL the fragment of $\mathrm{QCTL}^*$ where every temporal operator is immediately preceded by a path quantifier, and for a state formula $\psi$ the set $[\![\psi]\!]_G$ denotes the set of states in $G$ that satisfy the formula $\psi$.

**Logical relation induced by** QCTL **and** $\mathrm{QCTL}^*$**.** Given two MDPs $G$ and $G'$, the logical relation induced by $\mathrm{QCTL}^*$, denoted as $\preccurlyeq_Q^*$, (resp. by QCTL, denoted as $\preccurlyeq_Q$), is defined as follows:

$$\preccurlyeq_Q^* = \{(s, s') \in S \times S' \mid \forall \psi \in \mathrm{QCTL}^* : \text{ if } s \models \psi \text{ then } s' \models \psi\}$$

(resp. $\forall \psi \in \mathrm{QCTL}$).

## 5 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 2.** *For all atomic propositions $q, r$ and for all MDPs, we have:*

$$[\![\langle\mathsf{Almost}\rangle(\bigcirc q)]\!] = [\![\langle\langle 1\rangle\rangle(\bigcirc q)]\!] \tag{1}$$
$$[\![\langle\mathsf{Almost}\rangle(q \mathcal{W} r)]\!] = [\![\langle\langle 1\rangle\rangle(q \mathcal{W} r)]\!] \tag{2}$$
$$[\![\langle\mathsf{Positive}\rangle(\bigcirc q)]\!] = [\![\langle\langle 1,2\rangle\rangle(\bigcirc q)]\!] \tag{3}$$
$$[\![\langle\mathsf{Positive}\rangle(q \mathcal{U} r)]\!] = [\![\langle\langle 1,2\rangle\rangle(q \mathcal{U} r)]\!] \tag{4}$$

*Proof. Point 1.* The inclusion $[\![\langle \mathsf{Almost}\rangle(\bigcirc q)]\!] \supseteq [\![\langle\langle 1\rangle\rangle(\bigcirc q)]\!]$ follows from the fact that there exists a strategy for Player 1 such that for all strategies of Player 2 the next state reached satisfies $q$. It follows that the same strategy for Player 1 ensures the formula with probability 1. For the second inclusion $[\![\langle \mathsf{Almost}\rangle(\bigcirc q)]\!] \subseteq [\![\langle\langle 1\rangle\rangle(\bigcirc q)]\!]$ we consider two cases: (i) let $s \in [\![\langle \mathsf{Almost}\rangle(\bigcirc q)]\!]$ be a Player-1 state. Then there exists an available action $a$ that leads to a state that satisfies formula $q$. As $s$ is a Player-1 state, the transition function under $a$ has a unique successor. Therefore, playing the same action ensures $q$ also in the two-player interpretation. The second case is that $s$ is a probabilistic states. In that case all the successors in the support of the probabilistic transition function satisfy $q$. Therefore formula $q$ is also satisfied in the two-player interpretation.

*Point 2.* As for the previous point the inclusion $[\![\langle \mathsf{Almost}\rangle(q\mathcal{W}r)]\!] \supseteq [\![\langle\langle 1\rangle\rangle(q\mathcal{W}r)]\!]$ follows easily from the definition. For the second inclusion assume towards contradiction that for every strategy $\sigma$ for Player 1 there exists a strategy $\theta$ for Player 2 such that the play $\mathsf{Plays}(s,\sigma,\theta)$ violates $q\mathcal{W}r$. It follows that for every strategy $\sigma$ for Player 1 there exists a strategy $\theta$ for Player 2 such that play $\mathsf{Plays}(s,\sigma,\theta)$ satisfies $\neg r\,\mathcal{U}\neg q$. This is possible only if there exists a finite path to a $\neg q$ state that uses only $\neg r$ states, and the finite path has a positive probability in the $1\frac{1}{2}$-player interpretation of the MDP. It follows that for every strategy of Player 1 there is a positive probability of violating $q\mathcal{W}r$ and the contradiction follows.

*Point 3. and 4.* Point 3 follows similarly to Point 1, and Point 4 follows the same arguments as in Point 2. □

**Lemma 3.** *For all atomic propositions $r$ and for all MDPs we have:* $[\![\langle \mathsf{Positive}\rangle(\Box\, r)]\!] = [\![\langle \mathsf{Positive}\rangle(r\,\mathcal{U}\langle \mathsf{Almost}\rangle(\Box\, r))]\!]$.

*Proof.* The result follows from [15, Lemma 1] (shown even for a more general class of partially observable MDPs). □

**Lemma 4.** *For all atomic propositions $q, r$ and for all MDPs, we have:* $[\![\langle \mathsf{Positive}\rangle(q\mathcal{W}r)]\!] = [\![\langle\langle 1,2\rangle\rangle(q\,\mathcal{U}r)]\!] \cup [\![\langle\langle 1,2\rangle\rangle(q\,\mathcal{U}(\langle\langle 1\rangle\rangle(q\mathcal{W}\mathsf{false})))]\!]$.

*Proof.* By definition we have that $[\![\langle \mathsf{Positive}\rangle(q\mathcal{W}r)]\!] = [\![\langle \mathsf{Positive}\rangle((q\,\mathcal{U}r) \vee (\Box q))]\!]$. We write the formula as follows: $[\![\langle \mathsf{Positive}\rangle((q\,\mathcal{U}r) \vee (\Box q))]\!] = [\![\langle \mathsf{Positive}\rangle(q\,\mathcal{U}r)]\!] \cup [\![\langle \mathsf{Positive}\rangle(\Box q)]\!]$. By Lemma 3 we have that $[\![\langle \mathsf{Positive}\rangle(\Box q)]\!] = [\![\langle \mathsf{Positive}\rangle(q\,\mathcal{U}\langle \mathsf{Almost}\rangle(\Box q))]\!]$. Note that $\Box q \equiv q\mathcal{W}\mathsf{false}$. All these facts together with the already established translations presented in Lemma 2 give us the desired result. □

To complete the translation of temporal operators it remains to express the QCTL formula $[\![\langle \mathsf{Almost}\rangle(q\,\mathcal{U}r)]\!]$ in terms of C-ATL. We first introduce the $\mathsf{Apre}$ function:

***Apre.*** Given two sets of states $X \subseteq Y \subseteq S$ we define the predecessor operator $\mathsf{Apre}$ as follows:

$$\begin{aligned}\mathsf{Apre}(Y,X) = \{&s \in S_1 \mid \exists a \in \mathsf{Av}(s) \,:\, \delta_1(s,a) \in X\} \cup \\ &\{s_p \in S_P \mid \mathrm{Supp}(\delta_P(s_p)) \subseteq Y \wedge \mathrm{Supp}(\delta_P(s_p)) \cap X \neq \emptyset\}.\end{aligned}$$

As is shown in [20] we can express the states $[\![\langle \mathsf{Almost}\rangle(q\,\mathcal{U}r)]\!]$ using the following $\mu$-calculus notation, where $\mu$ (resp. $\nu$) denotes the least (resp. greatest) fixpoint:

$$[\![\langle \mathsf{Almost}\rangle(q\,\mathcal{U}r)]\!] = \nu Y.\mu X.([\![r]\!] \cup ([\![q]\!] \cap \mathsf{Apre}(Y,X))). \tag{5}$$

The fixpoint computation on an MDP with $n$ states can be described as follows: $Y_0$ is initialized to all states, and in each iteration $i$ the set $X_{i,0}$ is initialized to the empty set; and $X_{i,j+1}$ is obtained from $X_{i,j}$ applying the one step operators, and $Y_i$ is set as the fixpoint of iteration $i$. Formally, for $1 \leq i \leq n$ and $0 \leq j \leq n-1$ we have

$$Y_0 = [\![\mathsf{true}]\!]; \quad X_{i,0} = [\![\mathsf{false}]\!]; \quad X_{i,j+1} = ([\![r]\!] \cup ([\![q]\!] \cap \mathsf{Apre}(Y_{i-1}, X_{i,j}))); \quad Y_i = X_{i,n};$$

and then $Y_n = [\![\langle \mathsf{Almost} \rangle (q \, \mathcal{U} r)]\!]$. Next we show that the $\mathsf{Apre}$ function can be expressed in C-ATL. For C-ATL formulas $\psi_1, \psi_2$ such that $[\![\psi_1]\!] \subseteq [\![\psi_2]\!]$ we define:

$$F_{\mathsf{Apre}}(\psi_1, \psi_2) = \langle\!\langle 1 \rangle\!\rangle (\bigcirc \psi_1) \wedge \langle\!\langle 1, 2 \rangle\!\rangle (\bigcirc \psi_2)$$

**Lemma 5.** *For* C-ATL *state formulas* $\psi_1, \psi_2$ *such that* $[\![\psi_1]\!] \subseteq [\![\psi_2]\!]$ *we have:* $[\![F_{\mathsf{Apre}}(\psi_1, \psi_2)]\!] = \mathsf{Apre}([\![\psi_1]\!], [\![\psi_2]\!])$.

*Proof.* We prove the two inclusions. We start with $\mathsf{Apre}([\![\psi_1]\!], [\![\psi_2]\!]) \subseteq [\![F_{\mathsf{Apre}}(\psi_1, \psi_2)]\!]$. Let $s$ be a state in $\mathsf{Apre}([\![\psi_1]\!], [\![\psi_2]\!])$, we consider two cases: (i) $s \in S_1$; and (ii) $s \in S_P$. For the case (i) it follows from the definition of $\mathsf{Apre}$ that there exists an action $a \in \mathsf{Av}(s)$ such that the unique state $\delta_1(s, a)$ satisfies $\psi_1 \wedge \psi_2$. It follows that $s \in [\![\langle\!\langle 1 \rangle\!\rangle (\bigcirc \psi_1) \wedge \langle\!\langle 1, 2 \rangle\!\rangle (\bigcirc \psi_2)]\!]$ and therefore $s \in [\![F_{\mathsf{Apre}}(\psi_1, \psi_2)]\!]$. In case (ii) we have $s \in S_P$, $\mathrm{Supp}(\delta_P(s)) \subseteq [\![\psi_1]\!]$, and $\mathrm{Supp}(\delta_P(s)) \cap [\![\psi_2]\!] \neq \emptyset$. It follows that $s \in [\![\langle\!\langle 1 \rangle\!\rangle (\bigcirc \psi_1) \wedge \langle\!\langle 1, 2 \rangle\!\rangle (\bigcirc \psi_2)]\!]$ and therefore $s \in [\![F_{\mathsf{Apre}}(\psi_1, \psi_2)]\!]$.

We continue with the second inclusion $[\![F_{\mathsf{Apre}}(\psi_1, \psi_2)]\!] \subseteq \mathsf{Apre}([\![\psi_1]\!], [\![\psi_2]\!])$. Let $s$ be a state in $[\![F_{\mathsf{Apre}}(\psi_1, \psi_2)]\!]$, we again consider two cases: (i) $s \in S_1$; and (ii) $s \in S_P$. For case (i) assume $s \in [\![\langle\!\langle 1 \rangle\!\rangle (\bigcirc \psi_1) \wedge \langle\!\langle 1, 2 \rangle\!\rangle (\bigcirc \psi_2)]\!]$, it follows that there exists an available action $a \in \mathsf{Av}(s)$ such that the state $\delta_1(s, a)$ is in $[\![\psi_2]\!]$ and as we have $[\![\psi_2]\!] \subseteq [\![\psi_1]\!]$, we have that there exists an action $a \in \mathsf{Av}(s)$ such that $\delta_1(s, a) \in [\![\psi_1]\!] \cap [\![\psi_2]\!]$. For the second case (ii) when $s \in S_P$ we again assume $s \in [\![\langle\!\langle 1 \rangle\!\rangle (\bigcirc \psi_1) \wedge \langle\!\langle 1, 2 \rangle\!\rangle (\bigcirc \psi_2)]\!]$. The first part of the formula ensures that $\delta_P(s) \subseteq [\![\psi_1]\!]$ and the second part ensures that $\delta_P(s) \cap [\![\psi_2]\!] \neq \emptyset$. The desired result follows. □

The following lemma shows the first of the two inclusions:

**Lemma 6.** *For an MDP we have* $\preccurlyeq_C \subseteq \preccurlyeq_Q$.

*Proof.* We prove the counter-positive, i.e., we construct a mapping of formulas $f$ : QCTL $\rightarrow$ C-ATL such that given two states $s, s'$ and a QCTL formula $\psi$ we have that if $s \models \psi$ and $s' \not\models \psi$ then the C-ATL formula $f(\psi)$ is true in $s$ and not true in $s'$. We proceed by structural induction on the QCTL formula and replace parts that are in scope of a path quantifier by their C-ATL version. The cases where $\psi$ is an atomic proposition or a Boolean combination of formulas are straightforward. It remains to translate the formulas $\langle \mathsf{Almost} \rangle (\bigcirc \varphi_1)$, $\langle \mathsf{Almost} \rangle (\varphi_1 \mathcal{W} \varphi_2)$, and $\langle \mathsf{Almost} \rangle (\varphi_1 \mathcal{U} \varphi_2)$ for QCTL formulas $\varphi_1, \varphi_2$. The translation of the first two follows directly from Lemma 2, therefore it remains to translate the QCTL formula $\langle \mathsf{Almost} \rangle (\varphi_1 \mathcal{U} \varphi_2)$. We proceed by encoding the fixpoint computation of the $\langle \mathsf{Almost} \rangle (\varphi_1 \mathcal{U} \varphi_2)$ formula into nested C-ATL formulas. Let $n$ be the number of states of the MDP. Let $\{\widetilde{\phi}_i, \ \phi_{i,j} \mid 0 \leq i, j \leq$

$n\}$ be a set of formulas defined by the following clauses:

$$\begin{aligned} \widetilde{\phi}_0 &= \mathsf{true}; \\ \forall 1 \leq i \leq n: \ \phi_{i,0} &= \mathsf{false} \\ \forall 1 \leq i \leq n. \forall 0 \leq j \leq n-1: \ \phi_{i,j+1} &= f(\varphi_2) \vee (f(\varphi_1) \wedge F_{\mathsf{Apre}}(\widetilde{\phi}_{i-1}, \phi_{i,j})) \\ \forall 1 \leq i \leq n: \ \widetilde{\phi}_i &= \phi_{i,n}; \end{aligned}$$

By Lemma 5 the set of nested formulas $\phi_{i,j}$ represents the computation of $X_{i,j}$ and $\widetilde{\phi}_i$ the computation of $Y_i$ (for the computation of the fixpoint formula). It follows that we have $[\![\langle \mathsf{Almost} \rangle(\varphi_1 \mathcal{U} \varphi_2)]\!] = [\![\widetilde{\phi}_n]\!]$ and concludes the translation. The translation for formulas $\langle \mathsf{Positive} \rangle(\bigcirc \varphi_1)$, $\langle \mathsf{Positive} \rangle(\varphi_1 \mathcal{W} \varphi_2)$, and $\langle \mathsf{Positive} \rangle(\varphi_1 \mathcal{U} \varphi_2)$ to C-ATL formulas follows from Lemma 2 and Lemma 4. The desired result follows. □

**Lemma 7.** *For an MDP $G$ we have $\preccurlyeq_Q \subseteq \preccurlyeq_C$.*

*Proof.* Given an MDP with $n$ states, it follows from the proof of Theorem 2 for the combined-simulation game that the $n$-step approximation $\preccurlyeq_C^n$ is exactly the same as $\preccurlyeq_C$. We define a sequence $\Psi_0, \Psi_1, \ldots, \Psi_n$ of sets of formulas of QCTL with the property that $s \preccurlyeq_C^i t$ iff every formula $\psi \in \Psi_i$ that is true in $s$ is also true in $t$. We denote by $\mathsf{BoolC}(\Psi)$ all the formulas that consist of disjunctions and conjunctions of formulas in $\Psi$. We assume that $\mathsf{BoolC}(\Psi)$ does not contain repeated elements, therefore from finiteness of $\Psi$ follows finiteness of $\mathsf{BoolC}(\Psi)$. We define $\Psi_0 = \mathsf{BoolC}(\{q, \neg q \mid q \in \mathsf{AP}\})$, and for all $0 \leq i < n$ we define $\Psi_{i+1} = \mathsf{BoolC}(\{\Psi_i \cup \{\langle \mathsf{Positive} \rangle(\bigcirc \psi), \langle \mathsf{Almost} \rangle(\bigcirc \psi) \mid \psi \in \Psi_i\}\})$. The formulas in $\Psi_0, \Psi_1, \ldots, \Psi_n$ provide witnesses that for all $0 \leq i \leq n$ we have that $\preccurlyeq_Q \subseteq \preccurlyeq_C^i$, in particular we have that $\preccurlyeq_Q \subseteq \preccurlyeq_C$. □

**Theorem 4.** *For all MDPs $G$ and $G'$ we have $\preccurlyeq_Q = \preccurlyeq_C$.*

**Theorem 5.** *For all MDPs $G$ and $G'$ we have $\preccurlyeq_Q^* = \preccurlyeq_Q$*

*Proof. (Sketch).* We need to show that if a QCTL$^*$ formula distinguishes two states, then there is a QCTL formula that also distinguishes them. The basic idea is similar to the proof of [12, Theorem 7.1, assertion 2]. We first construct a deterministic parity automata given the formula in QCTL$^*$, and the almost-sure or positive solutions for MDPs with parity objectives can be encoded as a $\mu$-calculus formula [14]. The translation of $\mu$-calculus formulas to a QCTL formula is done as in Lemma 6. □

**Theorem 6.** *Given an MDP the relation $\preccurlyeq_Q^*$ can be computed in quadratic time using discrete graph theoretic algorithms.*

*Proof.* Follows directly from Theorems 3, 4, and 5. □

## 6 CEGAR for Combined Simulation

In this section we present a CEGAR approach for the computation of combined simulation.

### 6.1 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 $\mathsf{turn}$, 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 $\Pi$ 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

- $\overline{S} = \Pi$: the partitions in $\Pi$ are the states of the abstract game.
- 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.
- 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 $\pi_i$ given an action $a$ to a partition $\pi_j$ if some state $s \in \pi_i$ can make an $a$-transition to some state in $s' \in \pi_j$, and this gives more power to Player 2.
- 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.
- $\overline{s}_0$ is the partition in $\Pi$ 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 $\Pi$ 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 $\Pi$ that contains state $s_0$ (as in the case of simulation abstraction).
- 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 $\pi_i$ given an action $a$ to a partition $\pi_j$ if all states $s \in \pi_i$ can make an $a$-transition to some state in $s' \in \pi_j$, and 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 $\Pi$ of a two-player game $G$ we have: (1) $G \sim_{\mathcal{A}} Abs^{\Pi}_{\mathcal{A}}(G)$; and (2) $G \sim_{\mathcal{S}} Abs^{\Pi}_{\mathcal{S}}(G)$.*

*Example 4.* Consider a two-player interpretation of the MDP $G_2$ from Figure 3. The coarsest partition of $G_2$ is $\Pi = \{\pi_0, \pi_1, \pi_2\}$, where $\pi_0 = \{s^2_0, s^2_1, s^2_3\}, \pi_1 = \{s^2_2, s^2_4, s^2_6\}, \pi_2 = \{s^2_5\}$. The alternating-simulation abstraction and the simulation abstraction of $\Pi$ are depicted in Figure 4. □


**Fig. 4.** Alternating-simulation and simulation abstractions of $G_2$ from Figure 3.

### 6.2 Sound Assume-Guarantee Rule

In this section we 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* $G^{\mathcal{M}} = (S^{\mathcal{M}}, A^{\mathcal{M}}, \mathsf{Av}^{\mathcal{M}}, \delta^{\mathcal{M}}, \mathcal{L}^{\mathcal{M}}, s^{\mathcal{M}}_0)$ is defined exactly like the combined simulation game given $G^{\mathsf{Alt}}$ and $G'$, with the exception that the step-wise simulation gadget is defined using the transitions of $G^{\mathsf{Sim}}$ instead of $G^{\mathsf{Alt}}$. Formally, we change the transitions as follows:

– *Checking step-wise simulation conditions.* Transition (a) is redefined: for a state $(s, s', \mathsf{Sim}, 2)$ we have only one action $\bot$ available for Player 1 and we have $\delta^{\mathcal{M}}((s, s', \mathsf{Sim}, 2), \bot) = \{(t, s', \mathsf{Sim}, 1) \mid \exists a \in \mathsf{Av}^{\mathsf{Sim}}(s) : \ t \in \delta^{\mathsf{Sim}}(s, a)\}$.

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 (Player 2) is stronger than in the combined-simulation game $G^{\mathcal{C}}$. Hence winning in $G^{\mathcal{M}}$ for the proponent (Player 1) 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 $\Pi$ and then composing it with $G_1$. The sound assume-guarantee rule follows from Proposition 3 and Proposition 4.

**Proposition 5 (Sound assume-guarantee rule).** *Given games $G_1, G_2, G'$, and a partition $\Pi$ 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.,*

$$\frac{\mathsf{A} = G_1 \parallel Abs^{\Pi}_{\mathcal{A}}(G_2); \quad \mathsf{S} = G_1 \parallel Abs^{\Pi}_{\mathcal{S}}(G_2); \quad (\mathsf{A} \otimes \mathsf{S}) \sim_{\mathcal{M}} G'}{(G_1 \parallel G_2) \sim_{\mathcal{C}} G'} \tag{6}$$

If the partition $\Pi$ 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 6.4 we describe how to effectively compute the partitions $\Pi$ and refine them using CEGAR approach.

### 6.3 Counter-examples Analysis

If the premise $(\mathsf{A} \otimes \mathsf{S}) \sim_{\mathcal{M}} G'$ of the assume-guarantee rule (6) 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 (6) 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 $\theta_{\mathsf{abs}}$ in $G^{\mathcal{M}}$ as *abstract*, and to a counter-example $\theta_{\mathsf{con}}$ in $G^{\mathcal{C}}$ as *concrete*. An abstract counter-example is *feasible* if we can substitute partitions in $\mathsf{A}$ and $\mathsf{S}$ 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 [27] to check the feasibility of a counter-example by finding a *concretization* function $\mathsf{Conc}$ 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 $G_2$.

*Computation of the concretization.* Given an abstract counter-example $\theta_{\mathsf{abs}}$ and a state $\overline{s}$ in $G^{\mathcal{M}}$, let $\mathsf{Succ}(\overline{s})$ be the set of all successor of $\overline{s}$ in $G^{\mathcal{M}}$ given $\theta_{\mathsf{abs}}$ is fixed by Player 2. The formal description of the concretization is given in Figure 5, where the concretization of a state $\overline{s}$ in the abstract counter-example is computed from its successors in the DAG. We use the notation $\mathsf{Av}^1$, $\mathsf{Av}^2$, and $\delta^2$ to represent the action-available functions of $G_1$ and $G_2$, and the transition function of $G_2$, respectively.

*Illustrative examples.* We present intuitive description of two representative cases of concretization from Figure 5: (1) Consider a state $\overline{s} = ((s_1, \pi_2), s', \mathsf{Alt}, 2)$ where the abstract counter-example chooses the successor $\overline{s}' = ((s_1, \pi_2), s', \mathsf{Alt}, a, 1)$ (intuitively this corresponds to choice of action $a$). The concretization $\mathsf{Conc}(\overline{s}) = \{s \in \pi_2 \mid a \in \mathsf{Av}^2(s) \wedge s \in \mathsf{Conc}(\overline{s}')\}$ is the subset of states in $\pi_2$ where the action $a$ is available and $s$ also belongs to the concretization of the successor state $\overline{s}'$. (2) For a state $\overline{s} = ((s_1, \pi_2), s', \mathsf{Alt}, a, a', 1)$, the concretization is the set of states where action $a$ is not available or all successors given action $a$ belong to the concretization of the successors of $\overline{s}$.

*Example 5.* Consider MDPs $G_1, G_2, G'$ in Figure 3 interpreted as games and the abstract games $Abs^{\Pi}_{\mathcal{A}}(\widehat{G_2})$, $Abs^{\Pi}_{\mathcal{S}}(\widehat{G_2})$ in Figure 4. Let $\mathsf{A} = \widehat{G_1} \parallel Abs^{\Pi}_{\mathcal{A}}(\widehat{G_2})$ and $\mathsf{S} = \widehat{G_1} \parallel Abs^{\Pi}_{\mathcal{S}}(\widehat{G_2})$. Figure 6 shows part of an abstract counter-example to the modified combined-simulation game of $(\mathsf{A} \otimes \mathsf{S})$ and $G'$. In this counter-example the adversary first plays in the simulation gadget and the proponent responds by moving to a state $((s^1_1, \pi_1), s'_1)$ or a state $((s^1_1, \pi_1), s'_4)$ (their successors are not depicted in Figure 6). From the state $((s^1_1, \pi_1), s'_1)$ the adversary has a winning strategy by playing in the alternating-simulation gadget, and from $((s^1_1, \pi_1), s'_4)$ by playing in the simulation gadget. The dashed shows assign the concretization of states in the abstract counter-example. The counter-example is spurious, since the initial state of $G_2$ does not belong to the concretization of the initial state of the counter-example. □

### 6.4 CEGAR

The counter-example analysis presented in the previous section allows us to automatically refine abstractions using the CEGAR paradigm [17]. The code of the CEGAR algorithm for the assume-guarantee combined simulation is shown in Algorithm 1. The

$$\overline{s} = ((s_1, \pi_2), s') \quad : \quad \mathsf{Conc}(\overline{s}) = \begin{cases} \pi_2 & \overline{s} \text{ is a leaf} \\ \mathsf{Conc}(\overline{s}') & \text{otherwise, where } \mathsf{Succ}(\overline{s}) = \{\overline{s}'\} \end{cases}$$

$$\overline{s} = ((s_1, \pi_2), s', \mathsf{Sim}, 2) \quad : \quad \mathsf{Conc}(\overline{s}) = \{s \in \pi_2 \mid \exists a \in \mathsf{Av}^1(s_1) \cap \mathsf{Av}^2(s) : \delta^2(s,a) \cap \mathsf{Conc}(\overline{s}') \neq \emptyset\} \text{ where } \mathsf{Succ}(\overline{s}) = \{\overline{s}'\}$$

$$\overline{s} = ((s_1, \pi_2), s', \mathsf{Sim}, 1) \quad : \quad \mathsf{Conc}(\overline{s}) = \bigcap_{\overline{s}' \in \mathsf{Succ}(\overline{s})} \mathsf{Conc}(\overline{s}')$$

$$\overline{s} = ((s_1, \pi_2), s', \mathsf{Alt}, 2) \quad : \quad \mathsf{Conc}(\overline{s}) = \{s \in \pi_2 \mid a \in \mathsf{Av}^2(s) \wedge s \in \mathsf{Conc}(\overline{s}'), \} \text{ where } \mathsf{Succ}(\overline{s}) = \{\overline{s}'\} \text{ and } \overline{s}' = ((s_1, \pi_2), s', \mathsf{Alt}, 2, a)$$

$$\overline{s} = ((s_1, \pi_2), s', \mathsf{Alt}, a, 1) \quad : \quad \mathsf{Conc}(\overline{s}) = \bigcap_{\overline{s}' \in \mathsf{Succ}(\overline{s})} \mathsf{Conc}(\overline{s}')$$

$$\overline{s} = ((s_1, \pi_2), s', \mathsf{Alt}, a, a', 2) : \quad \mathsf{Conc}(\overline{s}) = \mathsf{Conc}(\overline{s}'), \text{ where } \mathsf{Succ}(\overline{s}) = \{\overline{s}'\}$$

$$\overline{s} = ((s_1, \pi_2), s', \mathsf{Alt}, a, a', 1) : \quad \mathsf{Conc}(\overline{s}) = \{s \in \pi_2 \mid a \notin \mathsf{Av}^2(s) \vee \delta^2(s,a) \subseteq \bigcup_{\overline{s}' \in \mathsf{Succ}(\overline{s})} \mathsf{Conc}(\overline{s}')\}$$

**Fig. 5.** Concretization function; $\overline{s}$ is a state in an abstract counter-example.

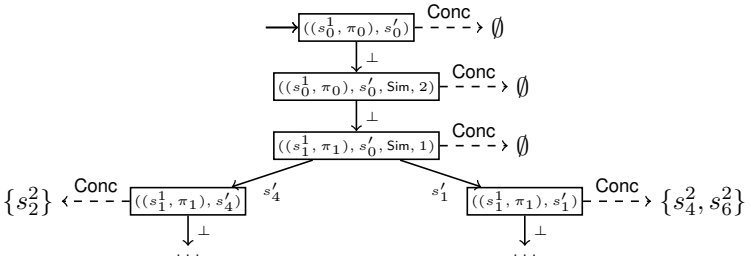


**Fig. 6.** Abstract counter-example to the modified combined-simulation game of $(\mathsf{A} \otimes \mathsf{S})$ and $G'$, where $\mathsf{A} = \widehat{G_1} \parallel Abs^{\Pi}_{\mathcal{A}}(\widehat{G_2})$ and $\mathsf{S} = \widehat{G_1} \parallel Abs^{\Pi}_{\mathcal{S}}(\widehat{G_2})$.

algorithm takes $G_1, G_2, G'$ as arguments and answers whether $(G_1 \parallel G_2) \sim_{\mathcal{C}} G'$ holds. Initially, the algorithms computes the coarsest partition $\Pi$ of $G_2$. Then, it executes the CEGAR loop: in every iteration the algorithm constructs $\mathsf{A}$ (resp. $\mathsf{S}$) 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 $\mathsf{Cex}$ in $G^{\mathcal{M}}$. In case the counter-example is feasible, then it corresponds to a concrete counter-example, and the algorithm returns NO. If $\mathsf{Cex}$ is spurious, the algorithm calls a refinement procedure that uses the concretization of $\mathsf{Cex}$ to return a partition $\Pi'$ finer than partition $\Pi$.

**Refinement procedure.** Given a partition $\Pi$ and a spurious counter-example $\mathsf{Cex}$ together with its concretization function $\mathsf{Conc}$ we describe how to compute the refined partition $\Pi'$. 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 $\mathsf{Cex}$ that contain $\pi$ as its component. Every state

**Algorithm 1** Assume-guarantee CEGAR for $\sim_{\mathcal{C}}$.

**Input:** Two-player games $G_1, G_2, G'$.
**Output:** **yes** if $G_1 \parallel G_2 \sim_{\mathcal{C}} G'$, otherwise **no**
  $\Pi \leftarrow$ coarsest partitioning of $G_2$
  **loop**
    $\mathsf{A} \leftarrow G_1 \parallel Abs^{\Pi}_{\mathcal{A}}(G_2)$;  $\mathsf{S} \leftarrow G_1 \parallel Abs^{\Pi}_{\mathcal{S}}(G_2)$
    $G^{\mathcal{M}} \leftarrow$ modified combined simulation game of $(\mathsf{A} \otimes \mathsf{S})$ and $G'$
    **if** Player 1 wins in $G^{\mathcal{M}}$ **then**  **return yes**
    **else**
      $\mathsf{Cex} \leftarrow$abstract counter-example in $G^{\mathcal{M}}$
      **if** Feasible(Cex) **then**  **return no**
      **else**  $\Pi \leftarrow$ Refine(Cex, $\Pi$)

$\overline{s}_i$ splits $\pi$ 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 $\pi$ that is finer than any of the 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 $\pi$ 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 $\Pi'$ is then defined as the union over $\mathcal{P}_\pi$ for all $\pi \in \Pi$.

*Example 6.* We continue with our running example. In Example 5 we showed that the abstractions of $\widehat{G_2}$ by the coarsest partition $\Pi$ lead to a spurious counter-example depicted in Figure 6. Consider the partition $\pi_1 = \{s_2^2, s_4^2, s_6^2\}$. There are three states in the counter-example that have $\pi_1$ as its component and the concretization function assigns to them three subsets of states: $\emptyset, \{s_2^2\}, \{s_4^2, s_6^2\}$. After the refinement partition $\pi_1$ is split into two partitions $\pi_1' = \{s_2^2\}$ and $\pi_1'' = \{s_4^2, s_6^2\}$. □

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

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

## 7 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 [35] model checker to specify the examples and generate input files for our tool.

*Observable actions.* To be compatible with the existing benchmarks (e.g. [33]) 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 [27] for refinement. Given a state $\overline{s}$ of the abstract counter-example with partition $\pi$ 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 [33]:

- $\mathsf{CS}_1$ *and* $\mathsf{CS}_n$ 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 [35] that models the *Leader election protocol* [32], 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 [39] 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 [33] that implements both assume-guarantee and monolithic approaches for *strong simulation* [43]. 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).

*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 [8] and models an error-correcting device that sends and receives data blocks over a communication channel. Notation $\mathsf{EC}(n, k, d)$ means that a data

| | | | | AGCS | | | | AGSS | | | | MONCS | | MONSS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ex. | $\|G_1\|$ | $\|G_2\|$ | $\|G'\|$ | $Time$ | $Mem$ | $I$ | $\|\Pi\|$ | $Time$ | $Mem$ | $I$ | $\|\Pi\|$ | $Time$ | $Mem$ | $Time$ | $Mem$ |
| $\mathsf{CS}_1(5)$ | 36 | 405 | 16 | 1.2s | 111MB | 49 | 85 | 6.11s | 213MB | 32 | 33 | **0.04s** | **34MB** | 0.18s | 95MB |
| $\mathsf{CS}_1(6)$ | 49 | 1215 | 19 | 2.55s | 210MB | 65 | 123 | 11.41s | 243MB | 40 | 41 | **0.05s** | **51MB** | 0.31s | 99MB |
| $\mathsf{CS}_1(7)$ | 64 | 3645 | 22 | 4.49s | 426MB | 84 | 156 | 31.16s | 867MB | 56 | 57 | **0.05s** | **82MB** | 0.77s | 113MB |
| $\mathsf{CS}_n(3)$ | 125 | 16 | 54 | 0.57s | 100MB | 9 | 24 | 33.43s | 258MB | 11 | 12 | **0.09s** | **35MB** | 11.29s | 115MB |
| $\mathsf{CS}_n(4)$ | 625 | 25 | 189 | 6.38s | 491MB | 15 | 42 | TO | - | - | - | **0.36s** | **107MB** | 1349.6s | 577MB |
| $\mathsf{CS}_n(5)$ | 3k | 36 | 648 | 106.39s | 2579MB | 24 | 60 | TO | - | - | - | **4.41s** | **384MB** | TO | - |
| $\mathsf{MER}(3)$ | 278 | 1728 | 11 | 0.38s | 92MB | 6 | 10 | 2.74s | 189MB | 6 | 7 | **0.02s** | **48MB** | 128.1s | 548MB |
| $\mathsf{MER}(4)$ | 465 | 21k | 14 | **3.72s** | **471MB** | 13 | 22 | 10.81s | 870MB | 10 | 11 | 10.73s | 1218MB | TO | - |
| $\mathsf{MER}(5)$ | 700 | 250k | 17 | **31.26s** | **1773MB** | 20 | 32 | 67s | 2879MB | 15 | 16 | - | MO | MO | - |
| $\mathsf{SN}(1)$ | 43 | 32 | 18 | 0.13s | 39MB | 3 | 6 | 0.28s | 88MB | 2 | 3 | **0.04s** | **30MB** | 3.51s | 135MB |
| $\mathsf{SN}(2)$ | 796 | 32 | 54 | 0.86s | 125MB | 3 | 6 | 66.09s | 258MB | 2 | 3 | **0.4s** | **104MB** | 3580.83s | 1022MB |
| $\mathsf{SN}(3)$ | 7k | 32 | 162 | 5.1s | **439MB** | 3 | 6 | TO | - | - | - | **5.33s** | 619MB | TO | - |
| $\mathsf{SN}(4)$ | 52k | 32 | 486 | **35.21s** | **2462MB** | 3 | 6 | TO | - | - | - | 44.14s | 3357MB | TO | - |
| $\mathsf{LE}(3,4)$ | 2 | 652 | 256 | **0.2s** | **64MB** | 6 | 14 | 1.63s | 223MB | 6 | 7 | 0.4s | 103MB | TO | - |
| $\mathsf{LE}(3,5)$ | 2 | 1280 | 500 | **0.37s** | **91MB** | 6 | 14 | Error | - | - | - | 1.98s | 255MB | Error | - |
| $\mathsf{LE}(4,4)$ | 3 | 3160 | 1280 | **0.55s** | **104MB** | 6 | 16 | TO | - | - | - | 10.41s | 1230MB | TO | - |
| $\mathsf{LE}(5,5)$ | 4 | 18k | 12k | **3.78s** | **389MB** | 6 | 18 | TO | - | - | - | - | MO | TO | - |
| $\mathsf{LE}(6,4)$ | 5 | 27k | 20k | **6.52s** | **734MB** | 6 | 20 | TO | - | - | - | - | MO | TO | - |
| $\mathsf{LE}(6,5)$ | 5 | 107k | 78k | **21.04s** | **1932MB** | 6 | 20 | TO | - | - | - | - | MO | TO | - |
| $\mathsf{PETP}(2)$ | 68 | 3 | 3 | 0.07s | 31MB | 0 | 2 | **0.04s** | 87MB | 0 | 1 | 0.07s | **30MB** | 0.04s | 90MB |
| $\mathsf{PETP}(3)$ | 4 | 1730 | 4 | **0.22s** | **65MB** | 6 | 8 | 0.29s | 153MB | 3 | 4 | 0.27s | 72MB | 1.07s | 170MB |
| $\mathsf{PETP}(4)$ | 5 | 54k | 5 | **1.52s** | **316MB** | 8 | 10 | 3.12s | 727MB | 4 | 5 | 8.17s | 957MB | 31.52s | 1741MB |

**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 that an error occurred during execution. The memory consumption is obtained using the Unix `time` command.

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 during the transmission. 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 [39] 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 [35]). Player 1 represents the virus and is trying to infect as 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

| | | | | 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 | 4.53s | 665MB | 1 | 7 | **1.11s** | 289MB | 1.94s | 389MB | 0 | 2 | 1.17s | **266MB** |
| EC$(64,7,16)$ | 549k | 385 | 257 | 73.21s | 3816MB | 1 | 131 | 13.65s | 1786MB | 19.99s | 2172MB | 0 | 2 | **8.73s** | **1590MB** |
| EC$(64,8,16)$ | 1.1m | 769 | 513 | - | MO | - | - | - | MO | **49.64s** | **3308MB** | 0 | 2 | - | MO |
| EC$(64,8,32)$ | 1.1m | 1025 | 513 | - | MO | - | - | - | MO | **50.64s** | **3164MB** | 0 | 2 | - | MO |
| PETG$(2)$ | 3 | 52 | 3 | 0.08s | 36MB | 4 | 6 | 0.03s | 30MB | 0.07s | 35MB | 4 | 6 | 0.03s | 30MB |
| PETG$(3)$ | 4 | 1514 | 4 | 0.2s | 63MB | 6 | 8 | 0.23s | 74MB | 0.22s | 62MB | 6 | 8 | 0.2s | 62MB |
| PETG$(4)$ | 5 | 49k | 5 | 1.77s | 314MB | 8 | 10 | 8.53s | 1105MB | **1.59s** | **305MB** | 8 | 10 | 5s | 729MB |
| VIR1$(12)$ | 14 | 4097 | 1 | 0.81s | 153MB | 15 | 30 | 1.38s | 259MB | **0.31s** | **108MB** | 2 | 4 | 1.46s | 209MB |
| VIR1$(13)$ | 15 | 8193 | 1 | 1.56s | 204MB | 16 | 32 | 4.4s | 597MB | **0.53s** | **172MB** | 2 | 4 | 2.69s | 404MB |
| VIR1$(14)$ | 16 | 16k | 1 | 2.74s | 349MB | 17 | 34 | 7.38s | 1020MB | **0.75s** | **246MB** | 2 | 4 | 6.92s | 826MB |
| VIR1$(15)$ | 17 | 32k | 1 | 4.56s | 630MB | 18 | 36 | 14.5s | 2071MB | **0.99s** | **486MB** | 2 | 4 | 0.92s | 1368MB |
| VIR1$(16)$ | 18 | 65k | 1 | 8.86s | 1010MB | 19 | 38 | 41.23 | 3761MB | **1.36s** | **851MB** | 2 | 4 | 25.23s | 2957MB |
| VIR1$(17)$ | 19 | 131k | 1 | 19.6s | 1808MB | 20 | 40 | - | MO | **2.09** | **1679MB** | 2 | 4 | 65.38s | 4300MB |
| VIR1$(18)$ | 20 | 262k | 1 | 38.52s | 2889MB | 21 | 42 | - | MO | **3.57s** | **2712MB** | 2 | 4 | - | MO |
| VIR2$(12)$ | 13 | 4096 | 1 | 0.76s | 142MB | 19 | 34 | 0.84 | 190MB | 0.68s | **118MB** | 9 | 14 | **0.54s** | 133MB |
| VIR2$(13)$ | 14 | 8192 | 1 | 1.57s | 192MB | 20 | 36 | 1.35s | 217MB | **0.94s** | **177MB** | 9 | 14 | 1.03s | 209MB |
| VIR2$(14)$ | 15 | 16k | 1 | 3.09s | 319MB | 21 | 38 | 2.37s | 391MB | **1.88s** | **309MB** | 9 | 14 | 2.02s | 389MB |
| VIR2$(15)$ | 16 | 32k | 1 | 5.04s | 633MB | 22 | 40 | 4.05s | 858MB | **2.13s** | **491MB** | 9 | 14 | 3.66s | 755MB |
| VIR2$(16)$ | 17 | 65k | 1 | 10.24s | 944MB | 23 | 42 | 7.11s | 1398MB | **3.94s** | **898MB** | 9 | 14 | 5.95s | 1359MB |
| VIR2$(17)$ | 18 | 131k | 1 | 24.42s | 1742MB | 24 | 44 | 24.27s | 2840MB | **8.03s** | **1681MB** | 9 | 14 | 18.67s | 2422MB |
| VIR2$(18)$ | 19 | 262k | 1 | 47.91s | 2906MB | 25 | 46 | 55.02s | 4238MB | **15.74s** | **2728MB** | 9 | 14 | 33.66s | 4071MB |

**Table 2.** Results for two-player games examples.

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.

## References


1. R. Alur, T. Henzinger, O. Kupferman, and M. Vardi. Alternating refinement relations. In *CONCUR*, LNCS 1466, pages 163–178. Springer, 1998.
2. R. Alur and T. A. Henzinger. Computer-aided verification. Unpublished, 2004.
3. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.
4. A. Aziz, V. Singhal, F. Balarin, R. Brayton, and A. Sangiovanni-Vincentelli. It usually works: The temporal logic of stochastic systems. In *CAV*, LNCS 939, pages 155–165. Springer, 1995.
5. C. Baier and J.-P. Katoen. *Principles of model checking*. MIT Press, 2008.
6. C. Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Trans. on Database Systems*, 5:241–259, 1980.

7. A. Bianco and L. de Alfaro. Model checking of probabalistic and nondeterministic systems. In *FSTTCS*, LNCS 1026, pages 499–513. Springer, 1995.
8. P. Cerný, M. Chmelik, T. A. Henzinger, and A. Radhakrishna. Interface simulation distances. In *GandALF*, EPTCS 96, pages 29–42, 2012.
9. R. Chadha and M. Viswanathan. A counterexample-guided abstraction-refinement framework for Markov decision processes. *ACM Trans. Comput. Log. 12*, page 1, 2010.
10. S. Chaki, E. M. Clarke, N. Sinha, and P. Thati. Automated assume-guarantee reasoning for simulation conformance. In *CAV*, LNCS 3576, pages 534–547. Springer, 2005.
11. K. Chatterjee, S. Chaubal, and P. Kamath. Faster algorithms for alternating refinement relations. In *CSL*, LIPIcs 16, pages 167–182. Schloss Dagstuhl, 2012.
12. K. Chatterjee, L. de Alfaro, M. Faella, and A. Legay. Qualitative logics and equivalences for probabilistic systems. *Logical Methods in Computer Science*, 5(2), 2009.
13. K. Chatterjee, L. de Alfaro, M. Faella, R. Majumdar, and V. Raman. Code-aware resource management. *Formal Methods in System Design*, 42(2):146–174, 2013.
14. K. Chatterjee, L. de Alfaro, and T. A. Henzinger. Qualitative concurrent parity games. *ACM Trans. Comput. Log.*, 12(4):28, 2011.
15. K. Chatterjee, L. Doyen, and T. A. Henzinger. Qualitative analysis of partially-observable Markov decision processes. In *MFCS*, LNCS 6281, pages 258–269. Springer, 2010.
16. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
17. E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In *CAV*, LNCS 1855, pages 154–169, 2000.
18. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.
19. L. de Alfaro, T. A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In *CONCUR*, LNCS 2154, pages 351–365. Springer, 2001.
20. L. de Alfaro, T. A. Henzinger, and O. Kupferman. Concurrent reachability games. In *FOCS*, pages 564–575, 1998.
21. K. Etessami, M. Z. Kwiatkowska, M. Y. Vardi, and M. Yannakakis. Multi-objective model checking of Markov decision processes. *Logical Methods in Computer Science*, 4(4), 2008.
22. L. Feng, M. Z. Kwiatkowska, and D. Parker. Automated learning of probabilistic assumptions for compositional reasoning. In *FASE*, LNCS 6603, pages 2–17. Springer, 2011.
23. J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1997.
24. E. Grädel, W. Thomas, and T. Wilke. *Automata, logics, and infinite games: a guide to current research*. LNCS 2500. Springer, 2002.
25. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994.
26. M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *FOCS*, pages 453–462, 1995.
27. T. A. Henzinger, R. Jhala, and R. Majumdar. Counterexample-guided control. In *ICALP*, LNCS 2719, pages 886–902. Springer, 2003.
28. T. A. Henzinger, R. Jhala, R. Majumdar, and S. Qadeer. Thread-modular abstraction refinement. In *CAV*, LNCS 2725, pages 262–274. Springer, 2003.
29. H. Hermanns, B. Wachter, and L. Zhang. Probabilistic CEGAR. In *CAV*, LNCS 5123, pages 162–175. Springer, 2008.
30. R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.
31. N. Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22:384–406, 1981.
32. A. Itai and M. Rodeh. Symmetry breaking in distributed networks. *Information and Computation*, 88(1), 1990.
33. A. Komuravelli, C. S. Pasareanu, and E. M. Clarke. Assume-guarantee abstraction refinement for probabilistic systems. In *CAV*, LNCS 7358, pages 310–326. Springer, 2012.

34. M. Z. Kwiatkowska, G. Norman, and D. Parker. Game-based abstraction for Markov decision processes. In *QEST*, pages 157–166, 2006.
35. M. Z. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *CAV*, LNCS 6806, pages 585–591, 2011.
36. M. Z. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Assume-guarantee verification for probabilistic systems. In *TACAS*, LNCS 6015, pages 23–37. Springer, 2010.
37. R. Milner. An algebraic definition of simulation between programs. In *IJCAI*, pages 481–489, 1971.
38. C. S. Pasareanu, D. Giannakopoulou, M. G. Bobaru, J. M. Cobleigh, and H. Barringer. Learning to divide and conquer: applying the l* algorithm to automate assume-guarantee reasoning. *Formal Methods in System Design*, 32(3):175–205, 2008.
39. G. L. Peterson. Myths about the mutual exclusion problem. *Information Processing Letters*, 12(3):115–116, 1981.
40. A. Pnueli. In transition from global to modular temporal reasoning about programs. In *Logics and Models of Concurrent Systems*, NATO Advanced Summer Institutes F-13, pages 123–144. Springer, 1985.
41. A. Pogosyants, R. Segala, and N. Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. *Distributed Computing*, 13(3):155–186, 2000.
42. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT Press, 1995. Technical Report MIT/LCS/TR-676.
43. R. Segala and N. A. Lynch. Probabilistic simulations for probabilistic processes. *Nord. J. Comput.*, 2(2):250–273, 1995.
44. M. Stoelinga. Fun with FireWire: Experiments with verifying the IEEE1394 root contention protocol. In *Formal Aspects of Computing*, 2002.

# A Technical appendix

We start with an example that shows that also for alternating games combined simulation is finer that the intersection of simulation and alternating-simulation relation.

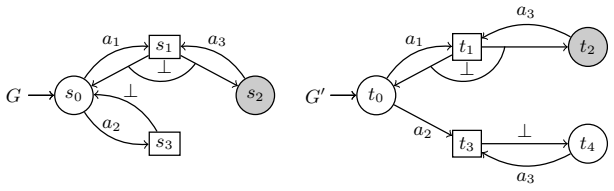


**Fig. 7.** Games $G, G'$ such that $G \sim_{\mathcal{S}} G'$ and $G \sim_{\mathcal{A}} G'$, but $G \not\sim_{\mathcal{C}} G'$.

*Example 7.* Figure 7 shows two alternating games $G, G'$, where the circular states belong to Player 1 and the rectangular states belong to Player 2, white nodes are labeled by proposition $p$ and gray nodes by proposition $q$. The largest simulation and alternating-simulation relations between $G$ and $G'$ are: $\mathcal{S}_{\max} = \{(s_0,t_0),(s_1,t_1),(s_2,t_2),(s_3,t_1)\}$, $\mathcal{A}_{\max} = \{(s_0,t_0),(s_0,t_4),(s_2,t_2),(s_3,t_3),(s_1,t_3),(s_1,t_1)\}$. Formula $\langle\!\langle 1 \rangle\!\rangle(\Box(p \wedge \langle\!\langle 1,2 \rangle\!\rangle(\mathsf{true}\ \mathcal{U}\ q)))$ is satisfied in state $s_0$, but not in state $t_0$, hence $(s_0,t_0) \notin \mathcal{C}_{\max}$. □

We now present detailed proofs of Lemma 1 and Theorem 2 in the context of alternating games.

**Lemma 8.** *Given two alternating games $G$ and $G'$, let $\mathcal{C}_{\max}$ be the combined simulation. For all $(s,s') \in \mathcal{C}_{\max}$ the following assertions hold:*

1. *For all Player 1 strategies $\sigma$ in $G$, there exists a Player 1 strategy $\sigma'$ in $G'$ such that for every play $\omega' \in \mathsf{Plays}(s',\sigma')$ there exists a play $\omega \in \mathsf{Plays}(s,\sigma)$ such that $\omega \sim_{\mathcal{C}} \omega'$.*
2. *For all pairs of strategies $\sigma$ and $\theta$ in $G$, there exists a pair of strategies $\sigma'$ and $\theta'$ in $G'$ such that $\mathsf{Plays}(s,\sigma,\theta) \sim_{\mathcal{C}} \mathsf{Plays}(s',\sigma',\theta')$,*

*Proof. Assertion 1.* As the states of Player 1 and Player 2 are distinguished by the turn atomic proposition, it follows from the fact that $(s,s') \in \mathcal{C}_{\max}$, that either (i) $s \in S_1$ and $s' \in S'_1$ or (ii) $s \in S_2$ and $s' \in S'_2$.

For the first case (i) we consider a winning strategy $\sigma^{\mathcal{C}}$ in $G^{\mathcal{C}}$ such that for all $(s,s') \in \mathcal{C}_{\max}$ and against all strategies $\theta^{\mathcal{C}}$ we have $\mathsf{Plays}((s,s'),\sigma^{\mathcal{C}},\theta^{\mathcal{C}}) \in [\![\Box(\neg p)]\!]_{G^{\mathcal{C}}}$. Given the Player 1 strategy $\sigma$ in $G$ we construct $\sigma'$ in $G'$ using the strategy $\sigma^{\mathcal{C}}$. Let $h$ be an arbitrary history in $G^{\mathcal{C}}$ that visits only states of type $(S \times S')$ that are in $\mathcal{C}_{\max}$ and ends in $(s,s')$. Consider a history $w \cdot s$ in $G$ and $w' \cdot s'$ in $G'$. Let $\sigma(w \cdot s) = a$, we define $\sigma'(w' \cdot s')$ as action $a' = \sigma^{\mathcal{C}}(h \cdot ((s,s'),\mathsf{Alt},2) \cdot ((s,s'),\mathsf{Alt},a,2))$, i.e., action $a'$ corresponds to the choice of the proponents winning strategy $\sigma^{\mathcal{C}}$ in response to the

adversarial choice of checking step-wise alternating-simulation followed by action $a$ in $G$. As both $s$ and $s'$ are Player-1 states we have that $|\delta(s,a)| = 1$ and $|\delta'(s',a')| = 1$. Let $(t,t')$ be the unique state reached in 2 steps from $((s,s'),\mathsf{Alt},a,a',2)$ in $G^{\mathcal{C}}$. Assume towards contradiction that $\mathcal{L}^{\mathcal{C}}((t,t')) = \{p\}$, then there exists a strategy for adversary that reaches a loosing state while the proponent plays a winning strategy $\sigma^{\mathcal{C}}$ and the contradiction follows. For the second case (ii) we have that states $s$ and $s'$ belong to Player 2, and there is a single action available for $\sigma'$.

*Assertion 2* The proof is similar to the first assertion, and instead of using the step-wise alternating-simulation gadget for strategy construction (of the first item) we use the step-wise simulation gadget from $G^{\mathcal{C}}$ to construct the strategy pairs.

**Theorem 7.** *For all alternating games $G$ and $G'$ we have $\mathcal{C}_{\max} = \preccurlyeq_C^* = \preccurlyeq_C$.*

*Proof. First implication.* We first prove the implication $\mathcal{C}_{\max} \subseteq \preccurlyeq_C^*$. We will show the following assertions:

- For all states $s$ and $s'$ such that $(s,s') \in \mathcal{C}_{\max}$, we have that every C-ATL$^*$ state formula satisfied in $s$ is also satisfied in $s'$.
- For all plays $\omega$ and $\omega'$ such that $\omega \sim_{\mathcal{C}} \omega'$, we have that every C-ATL$^*$ path formula satisfied in $\omega$ is also satisfied in $\omega'$.

We will prove the theorem by induction on the structure of the formulas. The interesting cases for the induction step are formulas $\langle\langle 1\rangle\rangle(\varphi)$ and $\langle\langle 1,2\rangle\rangle(\varphi)$, where $\varphi$ are path formulas.

- Assume $s \models \langle\langle 1\rangle\rangle(\varphi)$ and $(s,s') \in \mathcal{C}_{\max}$. It follows that there exists a strategy $\sigma \in \Sigma$ that ensures the path formula $\varphi$ from state $s$ against any strategy $\theta \in \Theta$. We want to show that $s' \models \langle\langle 1\rangle\rangle(\varphi)$. By Lemma 8(item 1) we have that there exists a strategy $\sigma'$ for Player 1 from $s'$ such that for every play $\omega' \in \mathsf{Plays}(s',\sigma')$ there exists a play $\omega \in \mathsf{Plays}(s,\sigma)$ such that $\omega \sim_{\mathcal{C}} \omega'$. By inductive hypothesis we have that $s' \models \langle\langle 1\rangle\rangle(\varphi)$.
- Assume $s \models \langle\langle 1,2\rangle\rangle(\varphi)$ and $(s,s') \in \mathcal{C}_{\max}$. It follows that there exist strategies $\sigma \in \Sigma, \theta \in \Theta$ that ensure the path formula $\varphi$ from state $s$. By Lemma 8(item 2) we have that there exist strategies $\sigma'$ and $\theta'$ such that the two plays $\omega' = \mathsf{Plays}(s',\sigma',\theta')$ and $\omega = \mathsf{Plays}(s,\sigma,\theta)$ satisfy $\omega \sim_{\mathcal{C}} \omega'$. By inductive hypothesis we have that $s' \models \langle\langle 1,2\rangle\rangle(\varphi)$.
- Consider a path formula $\varphi$. If $\omega \sim_{\mathcal{C}} \omega'$, then by inductive hypothesis for every subformula $\varphi'$ of $\varphi$ we have that if $\omega \models \varphi'$ then $\omega' \models \varphi'$. It follows that if $\omega \models \varphi$ then $\omega' \models \varphi$.

*Second implication.* It remains to prove the second implication $\preccurlyeq_C^* \subseteq \preccurlyeq_C \subseteq \mathcal{C}_{\max}$. We prove that from the assumption that $(s,s') \notin \mathcal{C}_{\max}$ we can construct a C-ATL formula $\varphi$ such that $s \models \varphi$ and $s' \not\models \varphi$. We refer to the formula $\varphi$ as a distinguishing formula. Assume that given states $s$ and $s'$ we have that $(s,s') \notin \mathcal{C}_{\max}$, then there exists a winning strategy in the corresponding combined-simulation game for the adversary from state $(s,s')$, i.e., there exists a strategy $\theta^{\mathcal{C}}$ such that against all strategies $\sigma^{\mathcal{C}}$ we have $\mathsf{Plays}((s,s'),\sigma^{\mathcal{C}},\theta^{\mathcal{C}})$ reaches a state labeled by $p$. As memoryless strategies are

sufficient for both players in $G^{\mathcal{C}}$ [24], there also exists a bound $i \in \mathbb{N}$, such that the proponent fails to match the choice of the adversary in at most $i$ turns. We construct the C-ATL formula $\varphi$ inductively:

Base case: Assume $(s, s') \notin \mathcal{C}_{\max}$ and let 0 be the number of turns the adversary needs to play in order to win. It follows that $(s, s')$ is a winning state for the adversary, i.e., $\mathcal{L}^{\mathcal{C}}((s, s')) = \{p\}$. It follows that $\mathcal{L}(s) \neq \mathcal{L}'(s')$. There are two options: (i) there exists an atomic proposition $q \in \mathsf{AP}$ that is true in $s$ and not true in $s'$ and distinguishes the two states, or (ii) there exists an atomic proposition $q \in \mathsf{AP}$ that is not true in $s$ and true in $s'$, in that case the formula $\neg q$ distinguishes the two states.

Induction step: Assume $(s, s') \notin \mathcal{C}_{\max}$ and let $n + 1$ be the number of turns the adversary needs to play in order to win. As the states of Player 1 and Player 2 are distinguished by the turn atomic proposition, it follows that either (i) $s \in S_1$ and $s' \in S'_1$ or (ii) $s \in S_2$ and $s' \in S'_2$. Otherwise the adversary could win in 0 turns from $(s, s')$.

We first consider case (i), i.e., $(s, s') \in S_1 \times S'_1$. The adversary can choose whether to verify (1) step-wise alternating-simulation ($\mathsf{Alt}$) or (2) step-wise simulation ($\mathsf{Sim}$). After that he chooses an action $a$ to be played according the adversarial strategy $\theta^{\mathcal{C}}$ in state $(s, s')$, such that no matter what the proponent plays, the adversary will win in $n$ turns. We consider two cases: (1) the adversary checks for step-wise alternating-simulation relation ($\mathsf{Alt}$), or (2) the adversary checks for step-wise simulation relation ($\mathsf{Sim}$). For case (1) we have that there exists an action $a$ for the adversary such that for all actions $a'$ of the proponent the adversary can win in $n$ turns from the unique successor $(t, t')$ of $(s, s')$ given $\mathsf{Alt}$ and $a$ was played by the adversary and $a'$ by the proponent. From the induction hypothesis there exists a C-ATL formula $\varphi_n$ such that $t \models \varphi_n$ and $t' \not\models \varphi_n$. We define the formula $\varphi_{n+1}$ that distinguishes states $s$ and $s'$ as $\langle\!\langle 1 \rangle\!\rangle(\bigcirc \varphi_n)$. For case (2), where the adversary plays $\mathsf{Sim}$ the proof is exactly the same, as step-wise simulation turn from Player 1 states coincides with step-wise alternating-simulation turn.

Next we first consider case (ii), i.e., $(s, s') \in S_2 \times S'_2$. The adversary can choose whether to verify (1) step-wise alternating-simulation ($\mathsf{Alt}$) or(2) step-wise simulation ($\mathsf{Sim}$). We start with first case (1): there is a unique action $a$ available to the adversary from state $((s, s'), \mathsf{Alt}, 2)$ and similarly a unique action $a'$ for the proponent from $((s, s'), a, \mathsf{Alt}, 1)$. The adversary chooses an action $t'$ from the $((s, s'), a, a', \mathsf{Alt}, 2)$ according to the winning strategy and the proponent chooses some action $t_i$ from a set of available successor $(t_1, t_2, \ldots, t_m)$. As the adversary follows a winning strategy $\theta^{\mathcal{C}}$ we have that it wins from all states $(t_i, t')$ for $1 \leq i \leq m$ in at most $n$ turns. From the induction hypothesis there exist C-ATL formulas $\varphi^i_n$ such that $t_i \models \varphi^i_n$ and $t' \not\models \varphi^i_n$. We define the formula $\varphi_{n+1}$ that distinguishes states $s$ and $s'$ as $\langle\!\langle 1 \rangle\!\rangle(\bigcirc(\bigvee_{1 \leq i \leq m} \varphi^i_n)$. For case (2) where the adversary verifies the step-wise simulation step, the proof is analogous. The formula that distinguishes states $s$ and $s'$ is $\langle\!\langle 1, 2 \rangle\!\rangle((\bigcirc \bigvee_{1 \leq i \leq m} \varphi^i_n))$.

The desired result follows. □