Formal Methods in Computer-Aided Design 2026

Property-driven Causal Abstractions for Markov Decision Processes

Jule Schmidt[1], Maximilian Weininger[1], Clemens Dubslaff[2], David Parker[3], Nils Jansen[1,4]
{jule.schmidt, maximilian.weininger, n.jansen}@rub.de, c.dubslaff@tue.nl, david.parker@cs.ox.ac.uk
[1]Ruhr-University Bochum, Germany, [2]Eindhoven University of Technology, The Netherlands,
[3]Oxford University, UK, [4]Radboud University, Nijmegen, The Netherlands

***Abstract*—Markov Decision Processes (MDPs) are widely used as decision-making models, commonly specified over factored state spaces through state variables and their valuations. The exponential blowup in the number of states renders many reasoning tasks in MDPs challenging. Abstractions are promising techniques to reduce MDPs and thus mitigate scalability issues. In this work, we introduce a notion of causality on factored MDPs and a novel property-driven causal abstraction technique that retains many characteristics of the original MDP model. For this, we rely on causal relations over state variable predicates and identify those states that share the same reasons for fulfilling or violating a given abstraction property. We theoretically and empirically compare various causal MDP abstractions using different model types such as MDPs, interval MDPs, or stochastic games. Our evaluation demonstrates the potential of our approach: For several standard benchmarks, we obtain small abstractions that allow us to compute near-optimal policies for the original MDP. Furthermore, our causal abstractions often generalize to related large-scale MDP models.**

I. Introduction

Markov Decision Processes (MDPs) are a common model for sequential decision-making under uncertainty. A standard way to specify an MDP is by factoring its state space, where each state corresponds to a valuation of a vector of state variables [1]. These variables inherently carry meaning, capturing structural properties of the environment and providing a compact description of the system dynamics. Probabilistic model checking tools like PRISM [2] or Storm [3] have input languages that follow this principle and provide mature implementations for formally analyzing MDPs.

Abstractions. As the number of variables increases, the induced state space quickly grows beyond the reach of exact analysis, which motivates the use of abstractions. Abstractions reduce the size of the original MDP model, simplifying the solving while preserving the information relevant to the property of interest. Predicate abstraction [4] groups states according to the truth values of predicates over the state variables. However, choosing suitable predicates is difficult, domain-dependent, and may result in abstractions that are either still too large to be solved or too coarse to be informative.

Example 1. *Throughout this paper, we use the electric taxi MDP [5], visualized in Fig. 1, as our illustrative running example. In brief, the taxi must navigate a 2D grid-world, to bring a passenger to a destination, while ensuring that it does not run out of battery. Passing by a charging station allows the battery to be recharged. Already for a* 3×3 *grid, this MDP consists of* 93 *states. A grid of size* 10×10 *has ca.* 8200 *states, a* 100×100 *grid has ca.* 10^7*, and even larger state spaces are to be expected in real-world scenarios. Many states, however, carry similar information: For example, the agent's grid position can be identical across different battery levels and regardless of the passenger's status. Slightly different position valuations may also induce the same level of safety criticality. An abstraction only focused on safety can merge these states, leaving a smaller model for exact analysis.*

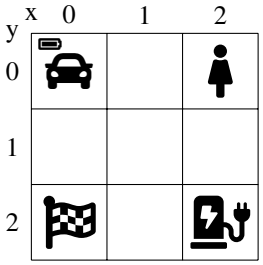


Fig. 1: The electric taxi MDP model, used as our running example throughout the paper.

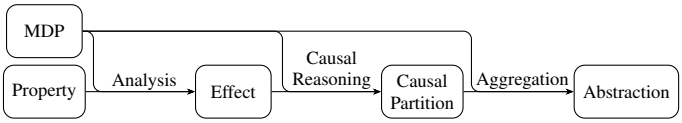


Fig. 2: The property-driven causal abstraction workflow.

Causality. The example illustrates that property-relevant information often depends on concrete valuations of specific state variables only. To reveal such information, we employ approaches from causal inference [6], [7]. In our context of MDPs, such reasoning helps identify state variables and valuations that cause the satisfaction or violation of a property.

Our approach: Property-driven causal abstractions. We provide formal concepts of causality for factored MDPs, establishing feature causality [8] over state variable predicates. We develop abstraction methods that (1) find causes for a given property of interest in a factored MDP, (2) use these causes to obtain state-space partitions, from which we (3) construct abstract models that provide a suitable tradeoff between model size and preservation of property-relevant information. We

https://doi.org/

now detail the steps of our approach as visualized in Fig. 2.

Effect sets and predicates. In the first step, we analyze the input MDP regarding the property at hand, such as reaching certain unsafe states. Based on that, we build so-called effect sets that distinguish states with respect to (potentially multiple different) probabilities to satisfy ("good effects") or violate ("bad effects") the property, or more fine-grained notions such as nearly satisfying or violating. Then, we employ a Boolean encoding of the factored state towards a predicate abstraction describing the membership of states to the various effect sets.

Causes. The second step introduces the novel concept of feature causality for factored MDPs. Our notion of *causes* takes into account specific valuations of state variables and not just the states, exploiting the inherent meaning of variables. Using causal inference, we determine such causes as minimal sets of predicates sufficient to show an effect.

Causal partition. We then partition the state space based on causally relevant parts. In particular, we consider three approaches towards such a partition: *one-shot*, which groups states that satisfy the same causes; *iterative*, which iteratively considers multiple effect sets; and *causal graph*, a standard way to aggregate states that neglect specific state valuations.

Abstraction. Given a partition and the original MDP, we finally build an abstraction by aggregating states in the same subset of the partition into one abstract state. To achieve a thorough evaluation of the approach, we compare three methods for abstracting an MDP's transition function: (1) using a weighted average, resulting in an MDP [5], (2) constructing an interval MDP where the intervals capture all possible transition probabilities [9], [10], [11], and (3) using a stochastic game where an opponent chooses original states from the partition [12].

Results. Our paper introduces property-driven causal abstractions and demonstrates their potential: Our experimental evaluation shows that, on many models, causal abstractions reduce model size, while hardly compromising the quality of optimal policies. Further, causes can generalize from small model variants to larger ones. Our detailed analysis of different choices in analyzing the property, performing causal reasoning, and aggregating states reveals that usually, focusing on states that are close to satisfying the property leads to the best partitions, and that using SG-based abstractions leads to the best performance. Overall, our contributions are the following:

- A new notion of causality exploiting the meaning of state variables in factored MDPs.
- Property-driven causal abstractions, including interval MDPs, and their theoretical analysis.
- An empirical evaluation of the tradeoff between abstraction and policy performance on standard benchmarks.

Limitations. Currently, our approach requires expensive steps, namely analyzing the whole MDP to find effect sets and computing the causes. However, our main goal is to achieve a fundamental understanding of feature causality and its potential for abstracting MDPs. Then, given this knowledge, the next step is to develop efficient ways to approximate the effect set and its causes, for example by reinforcement learning or by solving small MDPs precisely to obtain causes and then using them to abstract larger model variants. We remark that no automated abstraction method can be expected to work for every MDP, independent of property and structure. We view our approach as a step on the route towards scalability instead of a complete practical solution.

II. Related Work

Causality has been extensively studied in philosophy, social sciences, and artificial intelligence [13], [6], [14]. Seminal work by Halpern and Pearl led to *actual causality* [15] establishing cause-effect relationship-based structural causal models (SCMs). *Feature causality* [16] is a form of actual causality that does not rely on learning SCMs [8] and can serve as abductive explanations under constraints [17]. Our approach relates most to work on causal reasoning in Markovian models, recently considered in formal verification and reinforcement learning (RL). *Probabilistic causes* were defined as state sets for which the probability of an effect increases [18], [19], [20]. Other approaches rely on hyperproperties [21], [22], using counterfactual logic [23], or smallest prefix paths [24]. Due to their path dependency, they are not directly suitable for local causal state abstractions as we present here. In the context of RL, causal information is mainly used to improve the performance – cf. surveys on causal RL [25], [26]. Most approaches there rely on given SCMs and adapt existing model-free [27], [28] and model-based methods [29], [30].

Abstracting Markovian models. Various state-abstraction and refinement methods tackle the state-space-explosion problem in MDP-based analysis [31]. Probabilistic *bisimulation* [32] is prevalent, which has been extended to MDPs with factored state spaces [33] or using bisimulation metrics [34], [35]. PrIC3 [36] generates ad-hoc and local on-the-fly abstractions to find inductive invariants, but does not generate a-priori sufficient or necessary conditions for these abstractions. Simão et al. considered abstractions in constrained factored MDPs during RL at the level of variables [5]. Our abstraction method is more fine-grained, relying on variable valuations and their impact.

Causal abstractions. Lally et al. [37] consider causal abstractions towards interval MDP with theoretical guarantees, however relying on SCMs compatible with an underlying MDP. Zhang et al. [38] identify causal state features in so-called *block MDPs* using invariant causal prediction to improve RL but rely on different notions of causality depending on conditional probabilities. Another line of work learns causal graphs over state variables to derive abstractions [39], [40]. Clustering states in an MDP with similar causal importance for an RL objective has shown to improve learning [41]. Similarly, we propose a property-driven abstraction method lumping explanatory equivalent states.

In contrast to the aforementioned abstraction approaches, we provide a foundational formal framework for property-driven causal abstraction in factored MDPs that does not rely on SCM learning, statistical data, nor is limited to the application of RL.

III. Preliminaries

We recall Markov decision processes (MDPs, Section III-A) and their extensions to interval MDPs and stochastic games (Section III-B) that we use for abstractions. Finally, we provide the basic notions of feature causality (Section III-C).

For bounds $\ell, u \in \mathbb{N}$, let us denote by $[\ell..u]$ the interval $\{n \in \mathbb{N} \mid \ell \leq n \leq u\}$. Given a finite set Y, a *partition* of Y is a set $P = \{P_1, \dots, P_k \subseteq Y\}$ of pairwise disjoint sets such that $\biguplus_{i \in [1,k]} P_i = Y$. A *probability distribution* over Y is a function $\mathsf{d} \colon Y \to [0,1]$ where $\sum_{y \in Y} \mathsf{d}(y) = 1$. The set of all probability distributions over Y is denoted by $D(Y)$, and Y^ω is the set of all infinite sequences of elements in Y.

A. Markov Decision Processes

Definition 1 (Markov decision process (MDP))**.** An MDP [42] is a tuple $\mathsf{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ with finite sets of states $\mathcal{S}$ and actions $\mathcal{A}$, and a (partial) transition function $\mathcal{P} \colon \mathcal{S} \times \mathcal{A} \to D(\mathcal{S})$.

We say that an MDP $\mathsf{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ is *factored* if every state $s \in \mathcal{S}$ is a vector of n state valuations $(x_1, \dots, x_n) \in X_1 \times \dots \times X_n$ over state variables X_i associated with a finite domain of m_i elements $x_{i,j}$ for $i \leq n$, $j < m_i$. We write $\mathcal{A}(s)$ for the set of *available actions* in state $s \in \mathcal{S}$, i.e., those where $\mathcal{P}$ is defined, and assume $\mathcal{A}(s) \neq \emptyset$ for all $s \in \mathcal{S}$. We write $\mathcal{P}(s, a, s')$ for $\mathcal{P}(s, a)(s')$.

Example 2. *We model the taxi example from the introduction (see Fig. 1) as the following MDP: The state space is made up of four state variables $X_1 \times X_2 \times X_3 \times X_4 \in [0..2] \times [0..2] \times [0..5] \times \{0,1\}$, with two variables for the position, one for the battery, and one for the passenger's status. The actions are $\mathcal{A} = \{u, d, l, r, e\}$, where the first four move the taxi **u**p, **d**own, **l**eft, or **r**ight, and e allows the passenger to **e**nter or **e**xit the taxi. Action e is only available in states with battery > 0 at the pickup location ($x = 0, y = 0$). For a moving action, the transition function $\mathcal{P}(s, a)$ assigns probability 0.9 to the state in the chosen direction and probability 0.1 of getting stuck in traffic; in both cases, the battery decharges. States with battery $= 0$ are self-looping sink states, and the taxi resets battery to 5 by passing the charging station at $x = 2, y = 2$.*

The *semantics* of MDPs is defined as usual through policies and paths. A *policy* is a function $\pi \colon \mathcal{S} \to \mathcal{A}$ assigning to each state s an available action $a \in \mathcal{A}(s)$. We restrict to memoryless deterministic policies , as these suffice for optimality under the objectives we consider [43]. A *path* is an infinite sequence $\rho = s_0 a_0 s_1 a_1 \dots \in (\mathcal{S} \times \mathcal{A})^\omega$ where $\mathcal{P}(s_i, a_i, s_{i+1}) > 0$ for all $i \in \mathbb{N}_0$. We write ρ_i for the state s_i of a path ρ. Π is the set of all policies and $\mathrm{Paths}_\mathsf{M}$ the set of all paths. Applying a policy π to an MDP M induces for each state s a unique probability measure over paths $\mathbb{P}^\pi_{s,\mathsf{M}}$ [43, Chapter 10.1].

Property, objective and value. A *reachability* property combines a set of target states $\mathsf{T} \subseteq \mathcal{S}$ with an optimization objective $\mathrm{opt} \in \{\min, \max\}$ that captures the optimal probability to reach T, denoted $\Diamond \mathsf{T} := \{\rho \in \mathrm{Paths}_\mathsf{M} \mid \exists i \in \mathbb{N}_0 \colon \rho_i \in \mathsf{T}\}$. The *value of a state* s is $\mathsf{V}_\mathsf{M}(s) := \mathrm{opt}_{\pi \in \Pi} \mathbb{P}^\pi_{s,\mathsf{M}}[\Diamond \mathsf{T}]$.

Example 3. *An example property for the taxi model uses* $\mathrm{opt} = \min$ *and* $\mathsf{T} = \{(x, y, \mathit{battery}, \mathit{passenger}) \mid \mathit{battery} = 0\}$*. Intuitively, the value of a state captures the minimum probability for it to run out of battery.*

B. Interval Markov Decision Processes and Stochastic Games

Definition 2 (Interval MDP (IMDP))**.** An IMDP [9] is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_L, \mathcal{P}_U)$, where $\mathcal{S}$ and $\mathcal{A}$ are as for MDPs and $\mathcal{P}_L, \mathcal{P}_U \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ are functions providing lower and upper bounds on transition probabilities.

We write $\mathsf{M} \in \mathcal{M}$ to indicate that an MDP M is *consistent* with an IMDP $\mathcal{M}$. That is, they share state and action spaces, and for all states s, s' and actions a we have $\mathcal{P}_L(s, a, s') \leq \mathcal{P}(s, a, s') \leq \mathcal{P}_U(s, a, s')$ Note that if $\mathcal{P}_L(s, a, s') > \mathcal{P}_U(s, a, s')$, there exists no consistent MDP.

Example 4. *Recall from Example 2 that the chances of being stuck in traffic are precisely known to be 0.1. In an IMDP, we can instead have, e.g., $\mathcal{P}(s_0, l, s_1) = [0.8, 0.95]$ and $\mathcal{P}(s_0, l, s_2) = [0.05, 0.2]$, adding uncertainty to the transition probability. In Section V, we will abstract IMDPs from MDPs by grouping states. Then, the bounds on the transition probabilities are determined by the extremal transition probabilities in the group of states.*

IMDP semantics and value. The semantics of an IMDP is the set of all consistent MDPs. We focus on consistent MDPs with best- and worst-case value: For a given optimization direction $\mathrm{opt} \in \{\min, \max\}$, we write $\overline{\mathrm{opt}}$ for its complement, i.e., $\overline{\mathrm{opt}} := \min$ if $\mathrm{opt} = \max$ and $\overline{\mathrm{opt}} := \max$ otherwise. Then, $\mathsf{B}_\mathcal{M}(s) := \mathrm{opt}_{\mathsf{M} \in \mathcal{M}} \mathsf{V}_\mathsf{M}(s)$ and $\mathsf{W}_\mathcal{M}(s) := \overline{\mathrm{opt}}_{\mathsf{M} \in \mathcal{M}} \mathsf{V}_\mathsf{M}(s)$ are the best- and worst-case value of a state s, respectively.

Definition 3 (Stochastic game (SG))**.** An SG [44] is a tuple $\mathcal{G} = (\mathcal{S}, \mathcal{S}_\triangle, \mathcal{S}_\triangledown, \mathcal{A}, \mathcal{P})$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{P}$ are as for MDPs and $\mathcal{S} = \mathcal{S}_\triangle \uplus \mathcal{S}_\triangledown$, i.e., the state space is partitioned into states of the agent ($\triangle$) and opponent ($\triangledown$).

SG semantics and value. In SGs, we separately consider the sets of policies $\Pi_\triangle$ and $\Pi_\triangledown$ of the agent and opponent, where $\pi_\triangle \in \Pi_\triangle$ is a function $\mathcal{S}_\triangle \to \mathcal{A}$ and analogously for the opponent. Fixing a pair of policies $(\pi_\triangle, \pi_\triangledown)$ induces a probability measure $\mathbb{P}^{\pi_\triangle, \pi_\triangledown}_{s,\mathcal{G}}$ as in MDPs. The value of a state is $\mathsf{V}_\mathcal{G}(s) := \mathrm{opt}_{\pi_\triangle \in \Pi_\triangle} \overline{\mathrm{opt}}_{\pi_\triangledown \in \Pi_\triangledown} \mathbb{P}^{\pi_\triangle, \pi_\triangledown}_{s,\mathcal{G}}[\Diamond \mathsf{T}]$. We additionally define an analogue of the best-case value of IMDPs, where the opponent states also play in favour of the agent: $\mathsf{B}_\mathcal{G}(s) := \mathrm{opt}_{\pi_\triangle \in \Pi_\triangle} \mathrm{opt}_{\pi_\triangledown \in \Pi_\triangledown} \mathbb{P}^{\pi_\triangle, \pi_\triangledown}_{s,\mathcal{G}}[\Diamond \mathsf{T}]$.

C. Feature Causality

Feature causality [8] reasons about Boolean *features* and their influence on properties. For features F, let $\mathsf{F} = \{a, \overline{a} \mid a \in F\}$ denote the set of literals indicating that a feature a is active (a) or inactive ($\overline{a}$). An *assignment* $\delta \subseteq \mathsf{F}$ is a set of literals where no feature occurs twice, called *total* if every feature occurs exactly once. We denote by $\Delta(F)$ and $\Theta(F)$ the set of assignments and total assignments, respectively. The *semantics* $[\![\delta]\!]$ of an assignment δ is the set of consistent

total assignments, i.e., $[\![\delta]\!] := \{\theta \in \Theta(F) \mid \delta \subseteq \theta\}$. Let $\mathsf{Valid} \subseteq \Theta(F)$ be the set of *valid* assignments satisfying domain constraints, and $\mathsf{Effect} \subseteq \mathsf{Valid}$ a set of *effect* assignments. The goal is now to determine those assignments that are *sufficient to guarantee an effect* among those that are valid.

Definition 4 (Cause [8])**.** Let $\mathsf{Effect} \subseteq \mathsf{Valid} \subseteq \Theta(F)$. A *cause* of Effect w.r.t. Valid is an assignment $\gamma \in \Delta(F)$, where
(C1) $\emptyset \neq [\![\gamma]\!] \cap \mathsf{Valid} \subseteq \mathsf{Effect}$ (**sufficiency**)
(C2) $[\![\delta]\!] \cap \mathsf{Valid} \not\subseteq \mathsf{Effect}$ for all $\delta \subset \gamma$ (**subset minimality**).

Inspired by actual causality [7], **(C1)** implements the sufficiency condition, ensuring that all total assignments from γ are in Effect, and **(C2)** ensures subset minimality. A cause γ thus can serve as explanation for any assignment $\theta \in [\![\gamma]\!] \cap \mathsf{Effect}$. Causes is the set of all causes of Effect w.r.t Valid.

Cause-effect covers. A *cause-effect cover* is a set of causes $C \subseteq \mathsf{Causes}$ that explain the whole set Effect, i.e., $\mathsf{Effect} \subseteq \bigcup_{\gamma \in C} [\![\gamma]\!]$. We call C a *minimal cover* if there is no cause-effect cover C' where $|C'| < |C|$ [8, Section 4.2]. Computing an exact minimal cover is expensive, therefore heuristics based on *most general causes* [8, Definition 4] are used to establish small cause-effect covers.

IV. Causal Partitions for Factored MDPs

In this section, we establish several approaches to derive partitions of MDP state spaces using property-driven causal predicates. We thereby define the concept of feature causality for MDPs. We fix a factored MDP $\mathsf{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ and introduce predicates over its state variables X_i as a basis for predicate abstractions of its state space $\mathcal{S}$. We assume that state variables X_i are either *ordinal* or *categorical*. Ordinal variables are interpretable with a meaningful order, like a coordinate of a grid position. Let $O \subseteq [1..n]$ be the index set of all ordinal state variables in M. Categorical state variables represent distinct categories like the color of a signal.

A. Predicate Abstraction for Factored MDPs

We define a set of Boolean features F_{M}, which includes one Boolean for each state variable valuation except the smallest one in ordinal variables. Then, every state $s \in \mathcal{S}$ uniquely corresponds to a total assignment $\theta_s \in \Theta(F_{\mathsf{M}})$. Hence, the semantics of an assignment over the features F_{M} corresponds to a set of states in M. We refer to the Boolean features as $f_{i,k}$ where i is the index of the state variable X_i and $k \in [0..|X_i|]$ its k-th valuation. For an ordinal state variable X_i we assume the order $x_{i,j} \leq x_{i,k}$ iff $j \leq k$. We now describe sets of states through assignments over F_{M} modulo a theory over ordinal and categorical predicates.

$$F_{\mathsf{M}} \;=\; \bigcup_{i \in O} \{\, f_{i,k} \mid k \in [0..m_i] \,\} \setminus \{\, f_{i,0} \mid i \in O \,\}.$$

Let $s = (\mathfrak{s}_0, \ldots, \mathfrak{s}_n)$ be a concrete state in M. The interpretation of state s, θ_s, distinguishes ordinal and categorical features: If X_i is ordinal, then we define $f_{i,k} = 1$ if $\mathfrak{s}_i \geq x_{i,k}$ and $f_{i,k} = 0$ otherwise. That means that $f_{i,k}$ is interpreted as a predicate $X_i \geq x_{i,k}$. Note that binary variables X_i with domain $\{0, 1\}$ are covered by our definition of ordinal features. In this case, we simplify notation and write X_i instead of $X_i \geq 1$. If X_i is categorical, then $f_{i,k} = 1$ iff $\mathfrak{s}_i = x_{i,k}$, i.e., $f_{i,k}$ is interpreted as a predicate $X_i = x_{i,k}$. Given an assignment $\delta \in \Delta(F_{\mathsf{M}})$ and a state s, we write $\delta(s) = 1$ if θ_s is in the semantics of δ, and otherwise $\delta(s) = 0$.

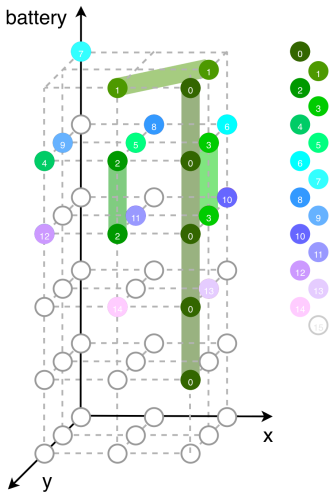


Fig. 3: Predicate partition of the taxi running example.

Example 5. *In our taxi MDP running example, all state variables are ordinal. We focus here on the battery: if* $battery = 3$*, that implies* $battery \geq 2$*,* $battery \geq 1$*, and* $battery \geq 0$*. In our definition of taxi, the domain of this variable is* $battery \in [5]$*. Therefore, we get five boolean variables:* $f_{battery,k}, k \in [1..5]$*. For states in which* $battery = 3$*, these are assigned as follows:* $f_{battery,k} = 1, k \in \{1, 2, 3\}$ *and* $f_{battery,k} = 0, k \in \{4, 5\}$*.*

B. One-shot Causal Partition

We propose a partition technique that is conceptually simple yet powerful. For an MDP M and a set of predicate assignments $C \subseteq \Delta(F_{\mathsf{M}})$, our partition groups states that are in the semantics of the same subset of assignments. We refer to the partition algorithm as PredPart, as it is well-defined independent of the method used to generate C. As we use causal predicates, and the input is exactly one set of predicates C, we call partitions generated using PredPart *one-shot causal partitions*, abbreviated to `C-1` in the evaluation. We define:

$$\mathsf{PredPart}(\mathsf{M}, C) = \{\{s' \in \mathcal{S} \mid \forall \delta \in C \colon \delta(s') = \delta(s)\} \mid s \in \mathcal{S}\}$$

Example 6. *Fig. 3 shows a predicate partition of the taxi MDP, in three dimensions (x, y, and battery) as the passenger is causally irrelevant. Circles represent states in the original MDP, and colors indicate partition subsets. For subset 0, the predicate is* $(battery \geq 1) \wedge (x \geq 2) \wedge (y \geq 2)$*, which covers all states where* x *and* y *are* 2*, and the battery level is* ≥ 1*.*

When we use the partition as input for aggregations (Section V), its size determines the size of the abstract model. There are at most $2^{|C|}$ groups in a PredPart partition, if the semantics of every combination of assignments are not

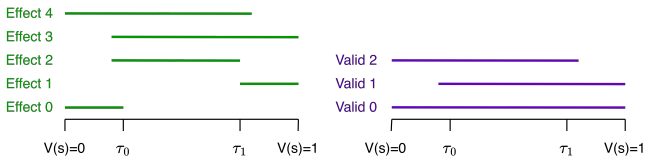


Fig. 4: Different settings of Effect and Valid for PredPart.

empty. If C is a minimal cause-effect cover, the size of the partition is minimal and still fully covers Effect. That means when PredPart is used with a minimal cause-effect cover, it is optimal in size and thus serves as a theoretical milestone of what causal partitions can achieve in terms of size. It is, however, expensive to compute, since it requires solving the model for the Effect set and computing a minimal cover of Effect. We formalize the suggested minimality characteristic in Remark 1.

Remark 1. Given an MDP M and sets Valid and Effect, let C be a minimal cover of Effect w.r.t. Valid. Then, $\mathsf{PredPart}(\mathsf{M}, C)$ is the coarsest (i.e., lowest cardinality) causal partition induced by Valid and Effect. This follows directly from the definition of minimal covers.

Choosing assignment sets. The Valid and Effect sets determine the state space partition and are built according to the given property and objective, such as the minimal reachability probability. The property induces a value for each state, and we use different *thresholds* on those values to partition states into the Valid and Effect sets. We consider percentile and relative thresholds. For percentile thresholds, ε fixes the percentage of states to include. For example, if $\varepsilon = 0.1$, and the 10% of states with lowest value have $\mathsf{V}(s) \leq 0.25$ and the highest 10% have $\mathsf{V}(s) \geq 0.93$, then we use those values as thresholds τ_0 and τ_1. Relative thresholds depend not only on ε, but also the value of a state of interest s_0, e.g. an initial state. Then, the relative threshold is $\tau = \mathsf{V}(s_0) \cdot (1 \pm \varepsilon)$. For example, if $\mathsf{V}(s_0) = 0.5$ and $\varepsilon = 0.1$, we get lower and upper relative thresholds $\tau_0 = 0.45$ and $\tau_1 = 0.55$.

Next to the thresholds, we also consider different ways to use them to construct Valid and Effect. For Valid, we can either use all reachable states as $\mathsf{Valid} = \{\theta_s \in \Theta(F_\mathsf{M}) \mid s \in \mathcal{S} : s \text{ is reachable}\}$ (Valid 0 in Fig. 4), or exclude states with values larger (Valid 2) or smaller (Valid 1) than τ. Further, we formalize the Effect set using the predicates in F_M: Given an interval $I \subseteq [0,1]$, $\mathsf{Effect}(I) = \{\theta_s \in \Theta(F_\mathsf{M}) \mid \mathsf{V}(s) \in I\}$. Fig. 4 shows different effect sets for intervals under two thresholds τ_0 and τ_1, e.g., Effect 4 is provided by $I_4 = [0, \tau_1)$.

Example 7. *For an MDP with a reachability property and maximizing objective,* Valid *1 in Fig. 4 excludes the "worst", and* Valid *2 the "best" states.* Effect *0 and 4 capture "bad" states. The meaning of* Effect *2 depends on* Valid*. For the specific example in Fig. 3, we consider a "good"* Effect *set with* Valid *0 and* Effect *3.*

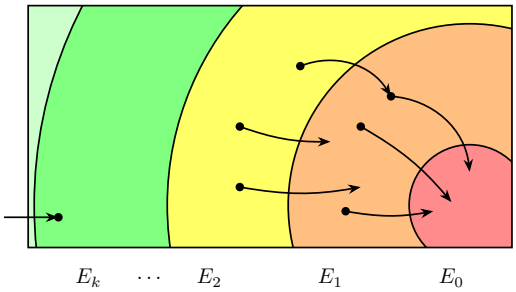


Fig. 5: Visualization of the intuition for IterPredPart.

C. Iterative Causal Partition

Abstractions built from PredPart tend to be small and to fixate on states with extreme values, motivating more elaborate approaches. We propose to iteratively consider finer slices of the state space as Effect, compute causes for each, and build a partition from the causal slices, as shown in Fig. 5. This is done by solving multiple reachability queries, where the previous iteration's Effect set becomes the next target set. Formally, for k iterations, with a threshold τ and starting from states of value 1, the initial sets are $\mathsf{Effect}_0 = \{s \in \mathcal{S} \mid \mathsf{V}(s) = 1\}$ and $\mathsf{Valid}_0 = \mathcal{S}$. Then, for the following iterations:

$$\mathsf{Effect}_i = \{s \in \mathcal{S} \mid \mathbb{P}_s(\Diamond \mathsf{Effect}_{i-1}) > \tau\}, i \in [1, k]$$
$$\mathsf{Valid}_i = \mathsf{Valid}_{i-1} \backslash \mathsf{Effect}_{i-1}$$

From the iterative Valid and Effect sets and the MDP, we obtain a list of sets of causes. We use this list in Algorithm 1 to construct the iterative partition, called `C-IT`. For each slice, it generates a partial partition like PredPart, retaining states outside Effect for future iterations. The resulting partition is more fine-grained than PredPart.

Algorithm 1 Deriving the iterative causal partition.

procedure GET `C-IT`(MDP M, cause list CS)
  $P = \emptyset$
  $\mathcal{S}' = \mathcal{S}$
  for all C in CS **do**
    $P = P \cup \{\{s' \in \mathcal{S}' \mid \forall \delta \in C\colon \delta(s') = \delta(s) \wedge \exists \delta \in C\colon \delta(s') = \text{true}\} \mid s \in \mathcal{S}'\}$
    $\mathcal{S}' = \mathcal{S}' \backslash \{s \in \mathcal{S}' \mid \exists \delta \in C\colon \delta(s) = \text{true}\}$
  end for
  if $|\mathcal{S}'| > 0$ **then** $P = P \cup \{\mathcal{S}'\}$
  end if
  return P
end procedure

Choosing iterative assignment sets. There are several ways to define iterations over Valid and Effect sets. One option is to fix a threshold as for PredPart, and re-use it for future iterations on new Effect sets. Alternatively, we predefine a desired number of k iterations, adapt thresholds dynamically, and recursively split by the median value of remaining states.

D. Causal Graph Partition

Traditionally, causality captures dependencies between features, which are visualized in causal graphs as edges between causally dependent features. These dependencies are usually assumed to be given or estimated from data using statistical causal discovery methods. Here, we can omit those estimations since we have access to the full model. Intuitively, a causal graph partition ignores features that do not cause any part of the effect. Using this information, we build a causal graph and identify the indices of irrelevant features $I_{\text{irr}} \subseteq [1..n]$. Then, CGPart groups states that only differ in irrelevant features:

$$\mathsf{CGPart}(\mathsf{M}, I_{\text{irr}}) = \{\{s' \in \mathcal{S} \mid \forall i \in [1..n] \backslash I_{\text{irr}} \colon s'_i = s_i\} \mid s \in \mathcal{S}\}$$

Given the MDP M and Effect, CGPart is always at least as large as a corresponding PredPart, formalized as follows.

Theorem 1. *Given an MDP* M *with sets* Valid *and* Effect*, a cause-effect cover C of* Effect *w.r.t.* Valid*, and I_{irr} the irrelevant features given the causal graph for* M *and* Effect*. Then, $\forall \mathcal{S}_{CG} \in \mathsf{CGPart}(\mathsf{M}, I_{irr}) \colon \exists \mathcal{S}_P \in \mathsf{PredPart}(\mathsf{M}, C) \colon \mathcal{S}_{CG} \subseteq \mathcal{S}_P$. Thus, $|\mathsf{CGPart}(\mathsf{M}, I_{irr})| \geq |\mathsf{PredPart}(\mathsf{M}, C)|$.*

Proof sketch. The first claim follows from Definition 4: An irrelevant feature cannot appear in any cause. The second claim follows from the first. [45, Appendix A] provides the full proof. □

Remark 2 (Inputs to the partition algorithms). Under the same hyperparameters, all three partitions are based on, or start from, the same set of input causes. Specifically, the set C of predicate assignments used for the one-shot partition is the set of causes that is obtained automatically using one Effect set. The same set C is used to derive the irrelevant features for the causal-graph partition. Further, C is the first iteration step for threshold-based iterative partitions.

V. From Partition to Abstraction

Given an MDP and a state space partition, we construct an abstract model via an *aggregation method*. This method maps each subset of the partition to a single abstract state and induces an abstract transition function that approximates the original. Section V-A provides three such methods: the weighted average method (WA) averages transition probabilities of aggregated states (Definition 5); the IMDP method represents all possible transition probabilities as an interval (Definition 6); and the SG method lets an opponent player choose an original state from an abstract one (Definition 7). Section V-B discusses (dis-)advantages of these methods.

A. Aggregation Methods

Example 8. *We start with an example to provide the intuition for the three aggregation methods and showcase their differences. Consider the MDP in Fig. 6a and a partition that groups $\{q, r\}$ and $\{t, u\}$, but leaves all other states as singletons. In all three abstractions, this partition forms the abstract state space, and successor distributions are aggregated by summing across concrete states in an abstract state.*

The WA abstraction *(Fig. 6b) obtains transition probabilities from an abstract state by* averaging *the probabilities of the concrete states. E.g., $\hat{\mathcal{P}}(\{q, r\}, b)(\{s\}) = \frac{1}{2} \cdot 0.1 + \frac{1}{2} \cdot 0 = 0.05$.*

The IMDP abstraction *(Fig. 6c) obtains transition probabilities via the minimum and maximum of the probabilities of its concrete states. When taking action b in $\{q, r\}$, the probability for $\{s\}$ is in $[0, 0.1]$ and for $\{t, u\}$ is in $[0.9, 0.2 + 0.8] = [0.9, 1]$. As we prove in Section V-B, the best- and worst-case value of the IMDP always contains the original MDP's value.*

The SG abstraction *(Fig. 6d) frames the abstract model as a 2-player game. Abstract states belong to the opponent and offer a choice of multiple successor states (gray squares). Each of these states corresponds to a set of original states that have equivalent outgoing distributions for each action. Here, states q and r behave differently, resulting in two choices in $\{q, r\}$.*

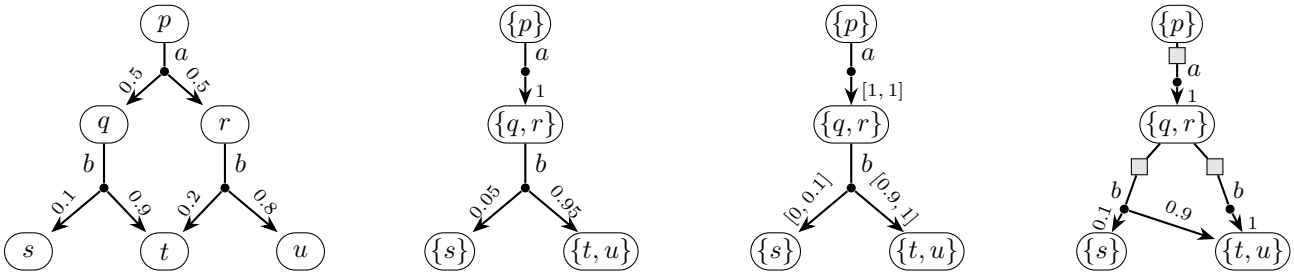
Action- and target-consistent partition. Before formally defining the aggregation methods, we provide two notions that simplify the definitions. *Action-consistency* requires that all states s in an abstract state $\hat{s}$ share the same set of available actions. This set then constitutes the set of available abstract actions $\hat{\mathcal{A}}(\hat{s})$, i.e., $\mathcal{A}(s) = \hat{\mathcal{A}}(\hat{s})$. This simplifies transferring policies between abstraction and original model. The assumption is easily satisfied by refining any given partition P to be action-consistent by splitting groups of states into new sets that agree on their available actions. Formally, for every subset $\hat{s} \in P$ and original state $s \in \hat{s}$, the refined partition contains the set $\{s' \in \hat{s} \mid \mathcal{A}(s') = \mathcal{A}(s)\}$. Related work often assumes that all actions are available in all states [46], [5].

Similarly, we require *target-consistency*, i.e., no subset of the partition contains both target and non-target states. While it is possible to soundly define the abstractions without this requirement, target-consistency simplifies the definitions (an abstract state $\hat{s}$ is a target iff $\hat{s} \subseteq \mathsf{T}$). Target-consistency is implicitly ensured in [5] and explicitly required in [12], [46].

Definition 5 (Weighted Average (WA) Abstraction)**.** For a P and MDP $\mathsf{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$, the WA abstraction MDP is $\hat{\mathsf{M}} = (\hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{P}})$ with: $\hat{\mathcal{S}} = P$, $\hat{\mathcal{A}} = \mathcal{A}$ and for all $s \in \hat{s}$, $\mathcal{A}(\hat{s}) = \mathcal{A}(s)$. For $\hat{s}, \hat{s}' \in \hat{\mathcal{S}}$ and $a \in \hat{\mathcal{A}}(\hat{s})$, the transition function is $\hat{\mathcal{P}}(\hat{s}, a, \hat{s}') = \sum_{s \in \hat{s}} \sum_{s' \in \hat{s}'} \frac{1}{|\hat{s}|} \mathcal{P}(s, a, s')$.

Definition 6 (IMDP Abstraction)**.** For P and MDP $\mathsf{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$, the IMDP abstraction is $\hat{\mathcal{M}} = (\hat{\mathcal{S}}, \hat{\mathcal{A}}, \mathcal{P}_L, \mathcal{P}_U)$ with $\hat{\mathcal{S}}$, and $\hat{\mathcal{A}}$ as in Definition 5. For $\hat{s}, \hat{s}' \in \hat{\mathcal{S}}$ and $a \in \hat{\mathcal{A}}(\hat{s})$, the lower and upper bounds on the transition probabilities are $\hat{\mathcal{P}}_L(\hat{s}, a, \hat{s}') = \min_{s \in \hat{s}} \sum_{s' \in \hat{s}'} \mathcal{P}(s, a, s')$ and $\hat{\mathcal{P}}_U(\hat{s}, a, \hat{s}') = \max_{s \in \hat{s}} \sum_{s' \in \hat{s}'} \mathcal{P}(s, a, s')$.

While the definitions of WA and IMDP abstraction follow naturally from Example 8, formally capturing the idea of the SG abstraction is more involved. We introduce the following notation: $\mathcal{P}_P(s) = \{(a, \mathsf{d}) \mid a \in \mathcal{A}(s), \mathsf{d} \in D(P) \colon \mathsf{d}(\hat{s}) = \sum_{s' \in \hat{s}} \mathcal{P}(s, a, s')\}$, i.e., $\mathcal{P}_P(s)$ is the set of all action-distribution pairs available in s, where d is the transition function lifted to the partition. For example, in Fig. 6d, we have $\mathcal{P}_P(\{q, r\}) = \{(b, [\{s\} \mapsto 0.1, \{t, u\} \mapsto 0.9]), (b, [\{t, u\} \mapsto 1]\}$. The subsets of the partition correspond



(a) MDP (b) WA (c) IMDP (d) SG

Fig. 6: Example MDP and three possible abstractions after grouping states $\{q, r\}$ and $\{t, u\}$: (a) Original MDP; (b) weighted average (WA) MDP abstraction; (c) IMDP abstraction; (d) SG abstraction.

to opponent states, where first, the opponent chooses an original state, and then the agent chooses an action (and a distribution over successor states). Thus, the agent states merge all original states that have the same action-distribution pairs.

Definition 7 (SG Abstraction)**.** For P and MDP $\mathsf{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$, the abstract SG is $\hat{\mathcal{G}} = (\hat{\mathcal{S}}, \hat{\mathcal{S}}_\triangle, \hat{\mathcal{S}}_\triangledown, \hat{\mathcal{A}}, \hat{\mathcal{P}})$ with $\hat{\mathcal{S}}_\triangledown = P$ and $\hat{\mathcal{S}}_\triangle = \{\mathcal{P}_P(s) \mid s \in \mathcal{S}\}$. $\hat{\mathcal{A}} = \mathcal{A} \cup \hat{\mathcal{S}}_\triangle$, where for $\hat{s} \in \hat{\mathcal{S}}_\triangledown$, we have $\mathcal{A}(\hat{s}) = \{\mathcal{P}_P(s) \mid s \in \hat{s}\}$, and picking the action surely leads to the selected successor, i.e., $\hat{\mathcal{P}}(\hat{s}, Y, Y) = 1$ for $Y \in \hat{\mathcal{S}}_\triangle$. For $Y \in \hat{\mathcal{S}}_\triangle$, recall that Y is a set of action-distribution pairs (a, d). Then, $\mathcal{A}(Y) = \{a \mid (a, \mathsf{d}) \in Y\}$, and for $\hat{s} \in \hat{\mathcal{S}}_\triangledown$ we have $\hat{\mathcal{P}}(Y, a, \hat{s}) = \mathsf{d}(\hat{s})$.

Definition Origins. The WA abstraction originates from [5]. We developed the IMDP abstraction independently, but it is effectively equivalent to transferring [46, Eqs. (5) and (6)] to the discrete setting. The SG abstraction adapts [12] to our notation, in particular adding action identifiers. The definitions do not require the given partition to be causal.

B. Theoretical Comparison of Aggregation Methods

We compare the aggregation methods with respect to three measures of quality: (i) *size*, (ii) *value bounds*, i.e., how the value(s) of the abstraction relate to the original MDP's value, and (iii) *policy performance*, i.e., how a worst-case optimal policy in the abstraction performs in the original MDP.

Abstraction size. Effectively, all abstractions are of the same size with an abstract state for every subset of the partition. The SG abstraction contains the additional agent states, but they need not be stored explicitly, as there is a strict alternation between states of agent and opponent [12, Section 3].

Value bounds. The WA abstraction does not provide any precision indication and the abstract MDP's value can be arbitrarily smaller or larger than the original MDP's. In contrast, the IMDP and SG abstraction yield an interval with a best- and worst-case value that is guaranteed to contain the original MDP's value. The width of the interval is then a measure of their precision. We summarize these insights in Theorem 2, where $\preceq$ indicates $\leq$ for $\max$ and $\geq$ for $\min$ objectives.

Theorem 2. *For MDP* M *and partition* P*, let* $\hat{\mathsf{M}}$*,* $\hat{\mathcal{M}}$*, and* $\hat{\mathcal{G}}$ *be the abstract MDP, IMDP, and SG according to Definitions 5 to 7. For every abstract state* $\hat{s} \in P$ *and original state* $s \in \hat{s}$*, the following hold:*

1) WA no guarantee: *It is possible that* $\mathsf{V}_\mathsf{M}(s) \preceq \mathsf{V}_{\hat{\mathsf{M}}}(\hat{s})$ *or that* $\mathsf{V}_{\hat{\mathsf{M}}}(\hat{s}) \preceq \mathsf{V}_\mathsf{M}(s)$.
2) IMDP guarantee: $\mathsf{W}_{\hat{\mathcal{M}}}(\hat{s}) \preceq \mathsf{V}_\mathsf{M}(s) \preceq \mathsf{B}_{\hat{\mathcal{M}}}(\hat{s})$.
3) SG guarantee: $\mathsf{V}_{\hat{\mathcal{G}}}(\hat{s}) \preceq \mathsf{V}_\mathsf{M}(s) \preceq \mathsf{B}_{\hat{\mathcal{G}}}(\hat{s})$.

Proof. The full proof is in [45, Appendix B]. There, for the WA abstraction, we provide concrete counterexamples where reachability probabilities in abstract states are strictly smaller or larger than in the original MDP. For the IMDP abstraction, we first show that there exist valid probability distributions giving upper and lower bounds on the values of the original MDP that are consistent with the IMDP. We then use an inductive argument to prove the relevant inequalities. For the SG abstraction, the proof from [12, Theorem 1] still applies to our definition. □

From a practical perspective, the value problem for all three models used as abstractions can be solved efficiently with dynamic programming: *value iteration* [47], [44] for MDPs or SGs; and *robust value iteration* [48] for IMDPs. Despite the differences in the underlying models, the efficiency and scalability of these solution methods are relatively similar.

Policy performance. An abstract policy can be applied in the original MDP as follows: For a policy $\hat{\pi}$ in the abstract MDP or IMDP, we obtain a policy π for the original MDP by setting $\pi(s) = \hat{\pi}(\hat{s})$ for every original state s, where $\hat{s}$ is the unique state with $s \in \hat{s}$. For a policy in the abstract SG, every agent state corresponds to a set $\mathcal{P}_P(s)$. Thus, we can transfer an abstract agent-policy in the SG to the MDP by selecting the action chosen in the unique abstract agent state corresponding to the original s. As the abstractions differ widely in how they aggregate information, the policy performance on the original MDP can also differ depending on the MDP's structure and the given partition, as empirically demonstrated in Section VI.

VI. Experimental Evaluation

Our evaluation focusses on the following questions:

(RQ1) How does the property affect performance?
(RQ2) How do different causal partitions compare?
(RQ3) How do different aggregation methods compare?
(RQ4) Do the causal partitions generalize to larger models?

A. Setup

Implementation. Our implementation builds on the model checkers PRISM [2] and Storm [3]. Specifically, we use version 1.11.1 of stormpy, the Python bindings of Storm for the general framework, and adapt PRISM for computing values of SGs and IMDPs. The overall experimental pipeline is run on a 2023 MacBook Pro with Apple M3 Pro Chip and 18GiB RAM. Cause computations were carried out with an end-to-end symbolic implementation [49] based on binary decision diagrams, and run on a workstation with Debian 12 Bookworm with AMD Ryzen Threadripper PRO 5965WX 24-Cores CPU and 8x64GiB Samsung DDR4-3200 RAM. The most expensive step of feature cause calculation uses a timeout of up to 12 hours, see [45, Appendix C] for details. Our implementation is available on Zenodo [50].

Benchmarks. We use the MDPs collected in [51] that are available in PRISM format and have unbounded reachability or safety properties. We exclude models where $\mathbb{P}_{\min}{=}1$ or $\mathbb{P}_{\max}{=}0$, as well as highly synthetic verification benchmarks with low inherent meaning in their state features. We configure the benchmark parameters to state spaces in the order of 10^4 (small), 10^5 (medium), and 10^6 (large). A full list is in [45, Appendix C].

Algorithms. We consider all variants of Effect and Valid sets introduced in Section IV, as well as the three partition types, namely causal graph (C-G), one-shot (C-1), and iterative (C-IT). We combine these with the three aggregation methods WA, IMDP, and SG, resulting in names such as C-IT-SG.

Remark 3. Our empirical evaluation focuses on reachability properties with minimizing and maximizing objectives. The former is also called *safety*, with the intuition that the probability of reaching a safety-critical state is minimized. Further, our definitions immediately includes reach-avoid properties, which we also analyse empirically. Extending our work to reward-based properties would require only minor changes to definitions and proof statements, but does not affect the derivation of causes. We exclude them, however, since checking reward-based properties on non-graph-preserving interval models is currently not supported by PRISM or Storm.

Metrics. We consider three metrics: (1) the size of the abstraction relative to the original model size, (2) the difference between the worst and best case value for the IMDP and SG aggregation method, and (3) the relative difference between the optimal policy in the MDP and the worst case abstract policy, scaled by the difference between minimal and maximal reachability probability (see [45, Appendix C]. All metrics range from 0 to 1, and small values are preferable.

(RQ1) *How does the property affect performance?*

The Effect set can vary depending on interpretation (see Fig. 4) and value threshold. We group the interpretations into those that capture "good" and "bad" effects. For maximizing properties, 3 and 1 capture "good" effects whereas 2 and 4 capture "bad" effects. Moreover, for the C-IT partition, we can pre-determine a desired number of iterations. We analyse configurations using the SG method on small models.

For the C-1 partition, results are similar across hyperparameters. The detailed analysis in [45, Appendix C], shows that the relative threshold with $\epsilon = 0.1$ achieves the best policy performance across benchmarks. We deem this the most important metric for the quality of our abstraction. For Effect choices, interpretations that find causes for large parts of the state space (3 and 4 in Fig. 4) lead to the best abstractions.

For the C-IT partition, we receive better abstractions with fixed numbers of iterations. Like with C-1, we find that results are very similar across thresholds, and also when using 6 or 10 fixed iterations. [45, Appendix C] includes detailed analysis. Between fixed thresholds or numbers of iterations, we observe that fixing the number of iterations leads to slightly larger abstractions, but with highly improved interval bounds and policy performance.

We use our findings to fix hyperparameters for the subsequent experiments. Overall we find that while some hyper-parameter values have little influence, such as the concrete threshold values, others matter greatly. In particular, we find that the semantics of the Effect set are decisive, and we observe the best performance with Effect sets covering a large part of the "good" states. Based on these findings, for the C-1 partition, we continue with relative $\epsilon = 0.1$ and use the Effect sets numbered 3 and 4 in Fig. 4. For the C-IT partition, we again observe that concrete threshold values have little impact, but fixing a larger number of iteration yields better performance. Therefore, we iterate 6 times, since that leads to the best ratio of performance and computational effort. In both cases, we observe that "good" effect sets achieve better abstractions than "bad" effects and include visualizations in the [45, Appendix C]. To still cover both interpretations, we continue with the hyperparameters on both types of effects.

(RQ2) *How do different causal partitions compare?*

We compare the C-G, C-IT, and C-1 partitions with the previously fixed hyperparameters using the SG abstraction. The C-G partition uses the same causes as C-1.

The C-G partition does not modify the model in 38% of our benchmarks. In these cases, no state variables can be completely removed since all are causally relevant to the property in some states and the abstract and original model are the same. We exclude those benchmarks from further evaluation but include details in [45, Appendix C].

C-IT partitions produce better abstractions than C-1 and C-G. In terms of model size, the C-1 partition produces the smallest abstract models on most benchmarks, retaining less than 20% of the original states on 13 out of 14 benchmarks. C-G returns the largest sized abstractions on most benchmarks. However, all three partitions are able to significantly reduce model sizes across benchmarks. On the qualitative metrics however, C-IT-based abstractions achieve the smallest value bounds and best policy performance, with small bounds on 10 out of 14 benchmarks and good policies on 5 out of 14

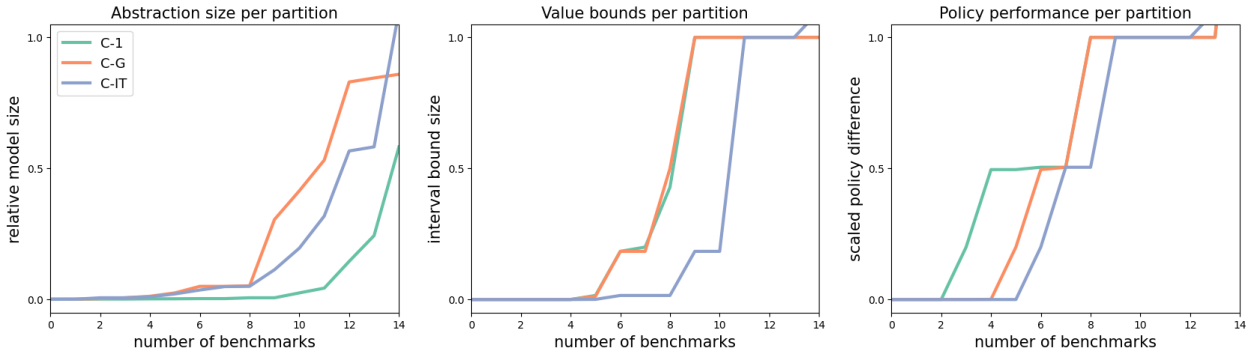


Fig. 7: Performance of different causal partitions across benchmarks.

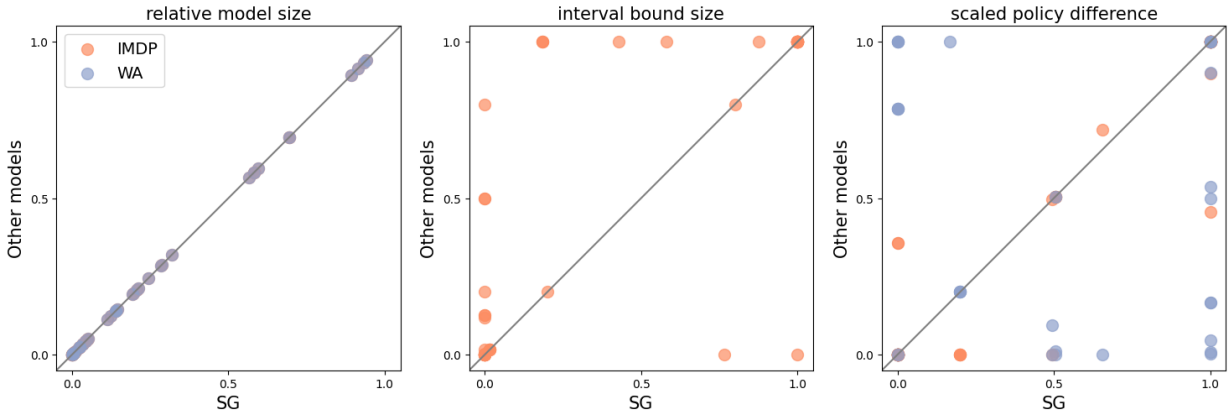


Fig. 8: Comparing the performance of different abstraction methods.

benchmarks. These results are visualized in Fig. 7. We find that `C-G`, which is usually employed by related works, is consistently outperformed. `C-IT` performs best, as it captures the most fine-grained interplay of causes. It is, however, expensive to derive. `C-1`, with lower computational cost and often reasonable performance marks a middle ground.

(RQ3) *How do different aggregation methods compare?*

We compare the performance of the aggregation methods in Fig. 8, with more details in [45, Appendix C]. Our empirical results confirm Theorem 2, i.e., the interval for the abstract IMDP and SG always contain the original value.

`SG` abstractions perform best, followed by `IMDP` and `WA`. Since all abstractions are built from the same partition, abstract model size is always the same. The `SG` abstractions mostly achieve smaller interval value bounds than the `IMDP` ones, often even achieving an interval size of 0. The policy performance however differs more strongly between the abstraction methods. `WA` abstractions often achieve a policy difference close to 0, but can also result in arbitrarily wrong policies without any guarantees. Between `IMDP`s and `SG`s, the results are often similar. As in the previous experiments, we observe that all methods perform better on effects that are interpreted as "good" than those that are "bad". Our findings here show a trade-off between performance and guarantees versus computational cost. Particularly, the `SG` aggregations offers guarantees and the best performance, as it tightly preserves uncertainty. This however comes at increased computational cost, compared to `WA`. The latter still often performs well and is easy to solve, but can be arbitrarily wrong, as the averaging may neglect specific behaviors.

(RQ4) *Do the causal partitions generalize to larger models?*

Using causality in MDPs and RL is often motivated by the potential of causal relations to generalize to scaled-up models. We test the potential of our causes to generalize to medium and large sized models . We apply causes retrieved on small models to larger settings, and report the results in Fig. 9. Additionally, [45, Appendix C] compares with medium-sized causes on the medium-sized models.

The causes generalize well in terms of size but at reduced performance. The small causes decrease the relative size of larger models to less than 20%, even in cases where the small abstraction maintained more than 60% of the original size. However, the policy difference often becomes worse, sometimes degrading to the worst possible policy. Still, on several models, the abstract policy attains a policy difference close to 0 on the large models, indicating optimal performance.

Abstract policies perform well on large models. The policy performance for large models is often closer to the original

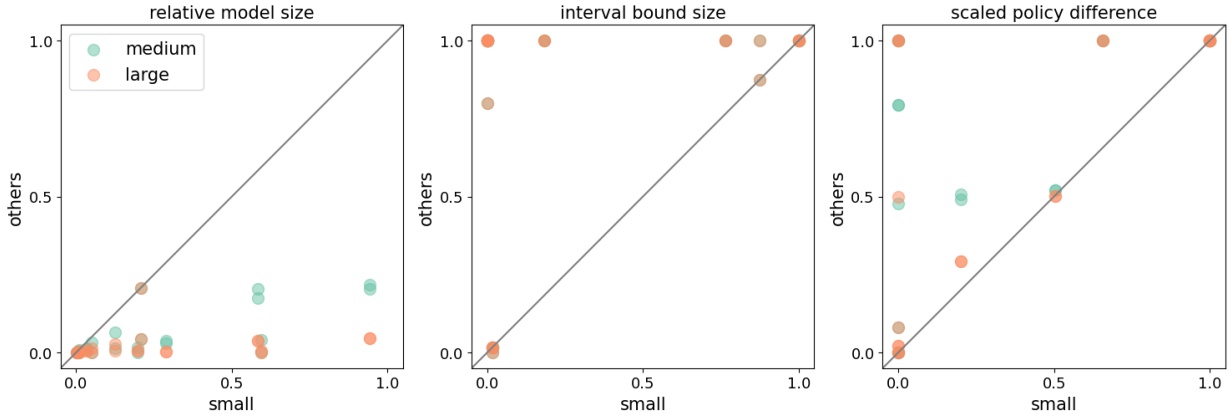


Fig. 9: Generalizing causes from the small models to larger model sizes.

model performance than for medium abstractions, even if both are derived from small causes. This is interesting, as obtaining causes for larger models is computationally more expensive. Our findings demonstrate the potential of our approach and provide an avenue to efficiently apply property-driven causal abstractions even on large models, provided they are parameterized to allow for small variants.

Causes fitted to the model size increase performance, but are computationally expensive. As we show in [45, Appendix C], those abstractions built from medium-sized causes usually receive tighter value bounds and better policy performance. The calculation of most general causes however becomes more computationally expensive on larger models, with several experimental configurations timing out even after 12 hours, as documented in [45, Appendix C]. Our straight-forward approach for generalizing causes between models shows our framework's potential to scale to models of larger sizes without further increase in computational demands.

B. General findings and practical implications

In our empirical evaluation, we observe an overall mixed performance across benchmarks, and across combinations of partitions and aggregation methods. We now provide further insights on our key takeaways from this evaluation.

Concerning benchmarks, we find that our abstractions are effective when whole state variables or large parts of their domain are irrelevant to the property of interest. In these cases, our abstractions become very small in size while maintaining good policy performance. On the other hand, handcrafted and highly artificial models where state variables do not carry much structural meaning, and where the outcome is solely dependent on a single state, do not produce effective abstractions. These benchmarks can show extreme differences in values between states, leading to a singular cause, and an abstraction consisting of only two abstract states.

Regarding the practical implications to draw from our experiments, in particular, we find a trade-off between computational expenses and the informativeness of partitions and therein the preciseness of abstractions. Specifically, we find that a combination of C-IT-SG is the most reliable yet most expensive choice. C-1-WA on the other hand offers a lightweight heuristic without guarantees on abstraction quality, that nonetheless often works well in practice. However, combinations of C-1 or C-IT with the SG aggregation method is preferable when guarantees matter.

VII. Conclusion

We formalize feature causality in MDPs to derive several types of property-driven predicate-based causal abstractions. We combine our causal state space partitions with three aggregation methods and analyze their theoretical (dis-)advantages. Empirical evaluation shows that our proposed methods can significantly reduce model size while maintaining good policy performance on several standard benchmarks. Further, the derived feature causes show potential to generalize to larger model sizes. A relevant limitation is that our approach requires to exactly analyze the full model to obtain the causal information. However, our aim is to gain a fundamental understanding of how causality can be applied to obtain abstractions that are capable of retaining relevant parts of the model. Future work aims to tackle this limitation by, among others, approximating the causal predicates from data or learned models.

Acknowledgments

This work was funded by the ERC Starting Grant 101077178 (DEUCE) and supported by the DFG project TRR 248 (see https://perspicuous-computing.science, project ID 389792660), and the NWO Veni grant VI.Veni.222.431.

References

[1] A. L. Strehl, C. Diuk, and M. L. Littman, "Efficient structure learning in factored-state mdps," in *AAAI*, pp. 645–650, AAAI Press, 2007.

[2] M. Z. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *CAV*, vol. 6806 of *Lecture Notes in Computer Science*, pp. 585–591, Springer, 2011.

[3] C. Hensel, S. Junges, J. Katoen, T. Quatmann, and M. Volk, "The probabilistic model checker storm," *Int. J. Softw. Tools Technol. Transf.*, vol. 24, no. 4, pp. 589–610, 2022.

[4] S. Graf and H. Saïdi, "Construction of abstract state graphs with PVS," in *CAV*, vol. 1254 of *Lecture Notes in Computer Science*, pp. 72–83, Springer, 1997.

[5] T. D. Simão, N. Jansen, and M. T. J. Spaan, "Alwayssafe: Reinforcement learning without safety constraint violations during training," in *AAMAS*, pp. 1226–1235, ACM, 2021.

[6] J. Pearl, *Causality*. Cambridge University Press, 2 ed., 2009.

[7] J. Y. Halpern, *Actual Causality*. The MIT Press, 08 2016.

[8] C. Dubslaff, K. Weis, C. Baier, and S. Apel, "Feature causality," *J. Syst. Softw.*, vol. 209, p. 111915, 2024.

[9] R. Givan, S. M. Leach, and T. L. Dean, "Bounded-parameter Markov decision processes," *Artif. Intell.*, vol. 122, no. 1-2, pp. 71–109, 2000.

[10] M. Suilen, T. Badings, E. M. Bovy, D. Parker, and N. Jansen, "Robust markov decision processes: A place where AI and formal methods meet," in *Principles of Verification (3)*, vol. 15262 of *Lecture Notes in Computer Science*, pp. 126–154, Springer, 2024.

[11] T. Badings, T. D. Simão, M. Suilen, and N. Jansen, "Decision-making under uncertainty: beyond probabilities," *Int. J. Softw. Tools Technol. Transf.*, vol. 25, no. 3, pp. 375–391, 2023.

[12] M. Kattenbelt, M. Z. Kwiatkowska, G. Norman, and D. Parker, "A game-based abstraction-refinement framework for markov decision processes," *Formal Methods Syst. Des.*, vol. 36, no. 3, pp. 246–280, 2010.

[13] E. Eells, *Probabilistic Causality*. Cambridge Studies in Probability, Induction and Decision Theory, Cambridge University Press, 1991.

[14] J. Peters, D. Janzing, and B. Schölkopf, *Elements of Causal Inference: Foundations and Learning Algorithms*. Cambridge, MA, USA: MIT Press, 2017.

[15] J. Y. Halpern, "A modification of the halpern-pearl definition of causality," in *IJCAI*, pp. 3022–3033, AAAI Press, 2015.

[16] C. Dubslaff, K. Weis, C. Baier, and S. Apel, "Causality in configurable software systems," in *ICSE*, pp. 325–337, ACM, 2022.

[17] N. Gorji and S. Rubin, "Sufficient reasons for classifier decisions in the presence of domain constraints," pp. 5660–5667, 2022.

[18] C. Baier, C. Dubslaff, F. Funke, S. Jantsch, R. Majumdar, J. Piribauer, and R. Ziemek, "From verification to causality-based explications (invited talk)," in *ICALP*, vol. 198 of *LIPIcs*, pp. 1:1–1:20, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[19] C. Baier, J. Piribauer, and R. Ziemek, "Foundations of probability-raising causality in markov decision processes," *Log. Methods Comput. Sci.*, vol. 20, no. 1, 2024.

[20] R. Oura and Y. Ito, "Probability-raising causality for uncertain parametric markov decision processes with PAC guarantees," in *UAI*, vol. 286 of *Proceedings of Machine Learning Research*, pp. 3300–3321, PMLR, 2025.

[21] E. Ábrahám and B. Bonakdarpour, "Hyperpctl: A temporal logic for probabilistic hyperproperties," in *QEST*, vol. 11024 of *Lecture Notes in Computer Science*, pp. 20–35, Springer, 2018.

[22] R. Dimitrova, B. Finkbeiner, and H. Torfah, "Probabilistic hyperproperties of markov decision processes," in *ATVA*, vol. 12302 of *Lecture Notes in Computer Science*, pp. 484–500, Springer, 2020.

[23] M. Kazemi, J. Lally, and N. Paoletti, "Causal temporal reasoning for markov decision processes," *Research Directions: Cyber-Physical Systems*, vol. 3, p. e3, 2025.

[24] R. Ziemek, J. Piribauer, F. Funke, S. Jantsch, and C. Baier, "Probabilistic causes in markov chains," *Innov. Syst. Softw. Eng.*, vol. 18, no. 3, pp. 347–367, 2022.

[25] Y. Zeng, R. Cai, F. Sun, L. Huang, and Z. Hao, "A survey on causal reinforcement learning," *CoRR*, vol. abs/2302.05209, 2023.

[26] Z. Deng, J. Jiang, G. Long, and C. Zhang, "Causal reinforcement learning: A survey," 2023.

[27] E. Bareinboim, A. Forney, and J. Pearl, "Bandits with unobserved confounders: A causal approach," in *NIPS*, pp. 1342–1350, 2015.

[28] A. Méndez-Molina, I. Feliciano-Avelino, E. F. Morales, and L. E. Sucar, "Causal based q-learning," *Res. Comput. Sci.*, vol. 149, no. 3, pp. 95–104, 2020.

[29] H. Sun, "Toward causal-aware RL: state-wise action-refined temporal difference," *CoRR*, vol. abs/2201.00354, 2022.

[30] M. Gasse, D. Grasset, G. Gaudron, and P. Oudeyer, "Causal reinforcement learning using observational and interventional data," *CoRR*, vol. abs/2106.14421, 2021.

[31] L. Li, T. J. Walsh, and M. L. Littman, "Towards a unified theory of state abstraction for MDPs," in *AI&M*, 2006.

[32] K. G. Larsen and A. Skou, "Bisimulation through probabilistic testing," *Information and Computation*, vol. 94, no. 1, pp. 1–28, 1991.

[33] R. Givan, T. Dean, and M. Greig, "Equivalence notions and model minimization in markov decision processes," *Artificial Intelligence*, vol. 147, no. 1, pp. 163–223, 2003. Planning with Uncertainty and Incomplete Information.

[34] N. Ferns, P. Panangaden, and D. Precup, "Metrics for finite markov decision processes," in *UAI*, pp. 162–169, AUAI Press, 2004.

[35] S. S. Ruan, G. Comanici, P. Panangaden, and D. Precup, "Representation discovery for mdps using bisimulation metrics," in *AAAI*, pp. 3578–3584, AAAI Press, 2015.

[36] K. Batz, S. Junges, B. L. Kaminski, J. Katoen, C. Matheja, and P. Schröer, "Pric3: Property directed reachability for mdps," in *CAV (2)*, vol. 12225 of *Lecture Notes in Computer Science*, pp. 512–538, Springer, 2020.

[37] J. Lally, M. Kazemi, and N. Paoletti, "Robust counterfactual inference in Markov decision processes," in *AAMAS*, 2026. to appear.

[38] A. Zhang, C. Lyle, S. Sodhani, A. Filos, M. Kwiatkowska, J. Pineau, Y. Gal, and D. Precup, "Invariant causal prediction for block mdps," in *ICML*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 11214–11224, PMLR, 2020.

[39] Z. Wang, X. Xiao, Z. Xu, Y. Zhu, and P. Stone, "Causal dynamics learning for task-independent state abstraction," in *ICML*, vol. 162 of *Proceedings of Machine Learning Research*, pp. 23151–23180, PMLR, 2022.

[40] Z. Wang, C. Wang, X. Xiao, Y. Zhu, and P. Stone, "Building minimal and reusable causal state abstractions for reinforcement learning," in *AAAI*, pp. 15778–15786, AAAI Press, 2024.

[41] S. Pranger, H. Chockler, M. Tappler, and B. Könighofer, "Test where decisions matter: Importance-driven testing for deep reinforcement learning," in *Advances in Neural Information Processing Systems* (A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, eds.), vol. 37, pp. 28103–28126, Curran Associates, Inc., 2024.

[42] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics, Wiley, 1994.

[43] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.

[44] A. Condon, "The complexity of stochastic games," *Inf. Comput.*, vol. 96, no. 2, pp. 203–224, 1992.

[45] J. Schmidt, M. Weininger, C. Dubslaff, D. Parker, and N. Jansen, "Property-driven causal abstractions for markov decision processes," *CoRR*, vol. abs/2607.26787, 2026.

[46] J. Jackson, L. Laurenti, E. W. Frew, and M. Lahijanian, "Strategy synthesis for partially-known switched stochastic systems," in *HSCC*, pp. 6:1–6:11, ACM, 2021.

[47] K. Chatterjee and T. A. Henzinger, "Value iteration," in *25 Years of Model Checking*, vol. 5000 of *Lecture Notes in Computer Science*, pp. 107–138, Springer, 2008.

[48] G. N. Iyengar, "Robust dynamic programming," *Mathematics of Operations Research*, vol. 30(2), pp. 257–280, 2005.

[49] E. Liem and C. Dubslaff, "Implicit computation of filtered prime implicants," *CoRR*, vol. abs/2608.05943, 2026.

[50] J. Schmidt, M. Weininger, C. Dubslaff, D. Parker, and N. Jansen, "Property-driven causal abstractions for Markov decision processes (FM-CAD 2026 artifact)," 2026. https://doi.org/10.5281/zenodo.21825827.

[51] A. Hartmanns, S. Junges, T. Quatmann, and M. Weininger, "The revised practitioner's guide to MDP model checking algorithms," *Int J Softw Tools Technol Transfer*, 2026.