Counterexample-Guided Policy Improvement for Parameterized Markov Decision Processes

Muqsit Azeem[1], **Debraj Chakraborty**[2], **Kush Grover**[1], **Sudeep Kanav** [2], **Jan Křetínský**[2,1]
[1]Technical University of Munich, Germany
[2]Masaryk University, Brno
muqsit.azeem@tum.de, chakraborty@fi.muni.cz, kush.grover@tum.de, jan.kretinsky@fi.muni.cz

Abstract

We focus on the qualitative analysis in Markov Decision Processes (MDPs), e.g. almost-sure reachability (with probability 1), which are fundamental in sequential decision-making systems. Counterexample-based methods have been proposed as a means to refine policies for such properties. Recently, our work (appeared at VMCAI 2025) introduced a scalable approach, called "1–2–3–Go!", which leverages decision-tree-based learning to effectively generalize policies from small instances to arbitrarily larger ones.

In this ongoing work, we propose to utilize the scalable 1–2–3–Go! approach to generate an initial "good" policy for large instances. This policy serves as a starting point for further refinement using counterexamples. By combining the scalability of 1–2–3–Go! with precise counterexample-guided refinement, this approach addresses limitations in the original framework and provides enhanced solutions for complex scenarios where smaller instances do not fully capture the dynamics of the system.

1 Introduction

Sequential decision-making under uncertainty is a fundamental problem in artificial intelligence (AI), with applications spanning robotics, autonomous driving, healthcare, finance, and more. A key mathematical framework for modeling such systems is Markov Decision Processes (MDPs) (Puterman 1994), which capture both the decision-making process and the inherent stochasticity of the environment. Policy synthesis for MDPs has been extensively studied, with established methods available for finding optimal policies (Puterman 1994). However, synthesizing policies for large, parameterized MDPs presents significant challenges due to the state-space explosion problem. Traditional approaches often either fail to scale or cannot generalize solutions to unseen parameterizations effectively.

Ensuring correctness in decision-making systems often requires satisfying qualitative properties, such as almost-sure or sure reachability and safety. These properties guarantee that desired outcomes are achieved with probability 1 or without any violations. For MDPs, Counterexample Guided Abstraction Refinement (CEGAR) (Clarke et al. 2003) has proven to be an effective method for synthesizing policies that satisfy such properties (Chatterjee, Chmelík, and Daca 2015). CEGAR iteratively refines an abstraction by analyzing counterexamples until the desired solution is achieved.

While counterexample-based improvement has been successful, its performance heavily depends on the choice of initial policies, as a well-chosen starting point can significantly reduce the number of refinement iterations required. Machine learning techniques can play a pivotal role in enhancing this performance. For instance, the 1–2–3–Go! approach (Azeem et al. 2025) employs decision-tree learning to generate policies for small, manageable instances of parameterized MDPs and then generalizes these policies to larger instances. This avoids the need to solve large models directly and leverages the generalizability of decision-tree representations. However, this generalization let alone might not be sufficient in general due to some differences in the policy because of large parameter. In the quantitative setting, verifying the correctness of the generalized policy is particularly challenging, as it requires precise knowledge of the actual model values—something achievable only through solving the model. In contrast, in the qualitative setting, we can simulate the model to identify counterexamples that violate the qualitative property, as a single trace is sufficient to demonstrate a violation.

Proposed Approach We propose a hybrid method for qualitative analysis that integrates learning-based policy synthesis with formal verification techniques to provide scalable and correct solutions for parameterized MDPs. Specifically, we extend the 1–2–3–Go! by complementing it with counterexample-guided improvement. This approach refines learned policies iteratively to ensure the satisfaction of qualitative properties, combining the strengths of machine learning and formal methods.

In this paper, we:

- Review the 1–2–3–Go! approach and its applicability.
- Present our idea for counterexample-guided policy improvement, integrating it with the 1–2–3–Go! approach to do quantitative analysis.

1.1 Related Work

The state-space explosion problem in large probabilistic models has been addressed through various methods.

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Model reduction techniques: Probabilistic bisimulation techniques for MDPs and discrete-time Markov chains (DTMCs) reduce model size while preserving temporal logic properties (Groote, Verduzco, and de Vink 2018). Other reduction methods focusing on specific temporal logic subsets have been proposed in (Kamaleson 2018), alongside high-level model reductions discussed in (Smolka et al. 2019; Lomuscio and Pirovano 2019). Efficient storage and sparse representations for extensive models are explored in (Hartmanns and Hermanns 2015), while compositional methods for verifying stochastic systems are outlined in (Feng 2014; Li and Liu 2019).

Machine Learning (ML) and Statistical Model Checking (SMC): ML methods, particularly reinforcement learning (RL), have been adapted for verification tasks with objectives such as reachability (Brázdil et al. 2014; Hahn et al. 2019). However, RL suffers from issues like sparse rewards, affecting scalability. Alternatively, SMC uses simulations to approximate desired properties but struggles with non-determinism resolution.

Counterexample-based improvments in sequential decision-making: Counterexample-based imrovements have been applied to decision-making under uncertainty in (Gangopadhyay and Dasgupta 2021), where counterexamples of a trained reinforcement learning policy are computed using Bayesian optimizations and failure points of the path are identified using gradient-based updates. For qualitative properties, CEGAR-based methods have been employed in (Chatterjee, Chmelík, and Daca 2015) to compute simulation relations for compositional analysis in two-player games and MDPs.

2 Preliminaries

Definition 1 (MDP) *A (finite)* Markov Decision Process *(MDP) is a tuple* $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta, \bar{s}, \mathsf{G})$ *where:*

- S *is a finite set of states,*
- A *is a finite set of actions, and* $\mathsf{A}(s)$ *denotes the (non-empty) set of enabled actions for each state* $s \in \mathsf{S}$*,*
- $\delta : \mathsf{S} \times \mathsf{A} \to \mathcal{D}(\mathsf{S})$ *is the probabilistic transition function mapping each state* $s \in \mathsf{S}$ *and enabled action* $a \in \mathsf{A}(s)$ *to a probability distribution over successor states,*
- $\bar{s} \in \mathsf{S}$ *is the initial state,*
- $\mathsf{G} \subseteq \mathsf{S}$ *is the set of goal states.*

A *Markov chain* (MC) can be viewed as an MDP where $|\mathsf{A}(s)| = 1$ for all $s \in \mathsf{S}$, i.e., a system exhibiting only probabilistic behavior, without non-determinism.

The **semantics** of an MDP are defined in the standard way through policies and paths in the Markov chain it induces. An infinite *path* $\rho = s_1 s_2 \ldots \in \mathsf{S}^\omega$ is a sequence of states extending indefinitely. A *policy* is a mapping $\sigma : \mathsf{S} \to \mathcal{D}(\mathsf{A})$ that determines which action to take in each state. A policy is said to be deterministic if it selects exactly one action per state, and randomized otherwise. The i-th state on a path is denoted by ρ_i, the set of all paths is represented as Paths, and the collection of all policies is denoted by Σ. Applying a policy σ to resolve nondeterminism in an MDP $\mathcal{M}$ results in a Markov chain $\mathcal{M}^\sigma$ (Baier and Katoen 2008, Definition 10.92). This Markov chain induces a unique probability measure $\mathbb{P}^\sigma_s$ over paths starting from a state s (Baier and Katoen 2008, Definition 10.10).

The **reachability objective** is integral to our MDP definition, reflected in the initial state $\bar{s}$ and the set of goal states G. The *value* V of the reachability objective represents the maximum probability of reaching a goal state from the initial state. Formally, this is expressed as:

$$\mathsf{V}(\mathcal{M}) = \mathsf{opt}_{\sigma \in \Sigma} \mathbb{P}^\sigma_{\bar{s}}[\Diamond \mathsf{G}],$$

where $\mathsf{opt} \in \{\max, \min\}$ indicates whether the goal is to maximize or minimize the probability of reaching G, and $\Diamond \mathsf{G} = \{\rho \in \mathsf{Paths} \mid \exists i.\, \rho_i \in \mathsf{G}\}$ represents the set of paths that eventually reach a goal state. This optimization can be restricted to deterministic and memoryless policies (Puterman 1994, Proposition 6.2.1). The qualitative analysis focuses on the reachability objective with probability 1.

2.1 State Space Structure and Scalable Models

For effective learning techniques, such as decision tree learning, it is crucial for the MDP's state space to be *factored*. This means that each state is defined as a tuple of values assigned to state variables, rather than being represented as a simple enumeration. In this factored representation, the state space is characterized by multiple dimensions, such as time or the state of a protocol. Each dimension corresponds to a state variable v_i with domain $\mathbb{D}_i$. Consequently, each state $s \in \mathsf{S}$ is a tuple $(v_1, v_2, \ldots, v_n)$, where $v_i \in \mathbb{D}_i$ is the value of the i-th state variable.

Definition 2 (pMDP) *A* parameterized Markov Decision Process *(pMDP) is a tuple* $\mathcal{M}_p = (\mathsf{S}, \mathsf{A}, \delta, \bar{s}, \mathsf{G}, \Theta)$*, where* Θ *is the parameter space and for each* $\theta \in \Theta$*, the tuple* $(\mathsf{S}_\theta, \mathsf{A}_\theta, \delta_\theta, \bar{s}_\theta, \mathsf{G}_\theta)$ *defines an MDP instance.*

Given a pMDP $\mathcal{M}_p$, different parameter values $\theta \in \Theta$ yield different instances of the MDP, and the parameterization can affect the state space, transition dynamics, or both.

Example 1 *Consider the pMDP in Figure 1. Each state is a tuple* (m, x) *with* $m \in \mathbb{D}_m = \{0, 1, \ldots, k\}$ *and* $x \in \mathbb{D}_x = \{0, 1, 2\}$*. The variable* m *identifies one of* $k+1$ *blocks, while* x *specifies the position within a block.*

The block $m = 0$ *is special, containing the initial state* $(0, 0)$*, the goal state* $(0, 2)$*, and a sink state* $(0, 1)$ *(not reachable to the goal). For* $m \in [1, k]$*, state* $x = 0$ *allows action* a *(progress to* $x = 1$*) or* b *(self-loop). For* $m \in [1, k-1]$*, state* $x = 1$ *permits action* a *(return to* $x = 0$*) or* b *(leave the block). Action* b *transitions with 50% chance to the sink state* $(0, 1)$ *or* $x = 0$ *in block* $m + 1$*. In block* k*, leaving progresses to the goal.*

This model scales with k*, yielding a state space of size* $2k + 3$ *and a maximum reachability probability of* 0.5^{k-1}.

2.2 Decision trees for policy representation

Knowing that the state space is a product of state-variables, a deterministic policy is a mapping $\prod_i \mathbb{D}_i \to \mathsf{A}$ from tuples of state variables to actions. By viewing the state variables as features and the actions as labels, we can employ machine learning classification techniques such as decision

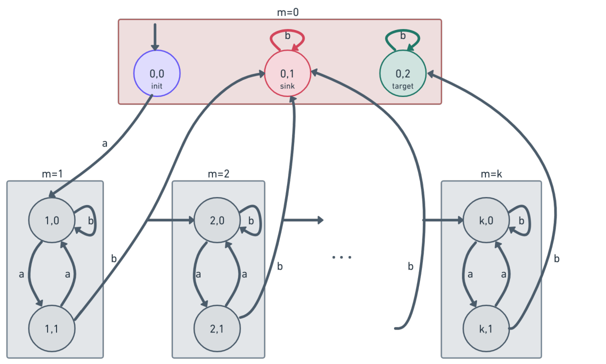
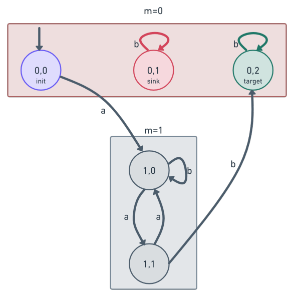
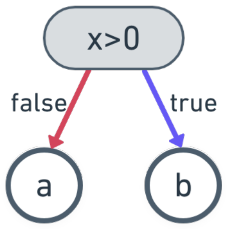


Figure 1: A parameterized, scalable MDP with $k+1$ blocks, described in Example 1.



(a) MDP instance for $m = 1$.



(b) Optimal policy represented as a decision tree.

Figure 2: Illustration of (a) the MDP instance for $m = 1$ from Figure 1 and (b) the optimal policy learned and represented as a decision tree.

trees , see e.g. (Mitchell 1997, Chapter 3), to represent a policy concisely. We refer to (Ashok et al. 2021) for an extensive description of the approach and its advantages. Here, we shortly recall the most relevant definitions in order to formally state our results.

Definition 3 *A* decision tree (DT) T *is defined as follows:*

- *T is a rooted full binary tree, meaning every node either is an* inner node *and has exactly two children or is a* leaf node *and has no children.*
- *Every inner node v is associated with a decision predicate α_v which is a boolean function $\mathsf{S} \to \{false, true\}$ (or equivalently $\prod_i \mathbb{D}_i \to \{false, true\}$).*
- *Every leaf ℓ is associated with an action $a_\ell \in A$.*

For a given state s, we use the following recursive procedure to obtain the action $\sigma(s) = a$ that a DT prescribes: Start at the root. At an inner node, evaluate the decision predicate on the given state s. Depending on whether it evaluates to false or true, recursively continue evaluating on the left or right child, respectively. At a leaf node, return the associated action a.

Example 2 *Consider again the MDP given in Figure 1. The optimal policy needs to continue towards the goal and not be stuck in any loops. This can be achieved by playing action a in states where $x = 0$ and action b in all other states. Traditionally, this policy would be represented as a lookup table, storing $2k + 3$ state-action pairs explicitly. Instead, we can condense the policy to the DT given in Figure 2b, mimicking the intuitive description of the policy: If $x > 0$, we play b, otherwise, we play a.*

We assume the MDPs are defined using high-level modeling languages such as Probmela (Baier and Katoen 2008), PRISM (Kwiatkowska, Norman, and Parker 2012) or MODEST (Hartmanns 2012).

3 Generalized policy using 1–2–3–Go! Approach

Our work (Azeem et al. 2025) introduces a novel approach 1-2-3-Go! for synthesizing policies in parameterized Markov Decision Processes (MDPs), addressing the critical challenge of scalability in large probabilistic models.

Solving small instances. By leveraging small instances of parameterized MDPs, we precisely compute optimal policies using model checking and generalize them using decision tree (DT) learning. This approach avoids explicit exploration of the state space for large instances, enabling the synthesis of scalable and interpretable policies for models that are otherwise beyond the reach of traditional verification tools.

Example 3 *Our running example has one parameter k, the number of modules. In fact, we will see that it suffices to consider $\mathcal{B} = \{b_1\}$ where $b_1 := \langle k{=}1\rangle$, i.e. only learn on the simplest instance of the MDP as depicted in Figure 2a.*

Example 4 *For our running example in Example 1, we only consider one base instance. Thus, our data is given by the function σ_1 for all reachable non-goal states in the MDP. Concretely, we have pairs of state $(0, 0)$ with action a, then $(1, 0)$ also with a, and $(1, 1)$ with b.*

Generalization. The key innovation in our work lies in recognizing and utilizing the structural regularities in parameterized MDPs. While numerical values across different parameterized instances may vary, the decision patterns often remain consistent. By learning these patterns from small, manageable instances, we encode them into decision trees that generalize effectively to arbitrarily large parameterized models. This process reduces the computational cost significantly while ensuring that the synthesized policies maintain their quality and scalability.

Example 5 *For our example data set D constructed in Example 4, the result of the DT learning is the DT depicted in Figure 2b. This policy is in fact optimal for all k; see Example 2 for an explanation of this. In addition to being optimal, it is also small and perfectly explainable.*

In contrast, if we are interested in a huge instance of this model, e.g., setting $k = 10^{15}$, already storing the resulting MDP in the memory in order to compute an optimal policy is challenging or even infeasible for a large enough k.

Additionally, the policy produced by state-of-the-art model checkers is represented as a lookup table with as many rows as there are states.

The experimentation on standard benchmarks demonstrate its ability to produce near-optimal policies for large-scale models. The results show that policies synthesized using our method perform consistently well across a wide range of parameterized instances, often achieving performance comparable to optimal policies computed for smaller instances. The decision tree representation further enhances scalability and interpretability, making our approach practical for real-world applications.

Our method offers a simple yet powerful framework for tackling the state-space explosion problem in large parameterized MDPs. By integrating learning-based synthesis with structural generalization, we provide a scalable solution that sets a new standard for policy synthesis in probabilistic models. This work not only addresses current limitations in the field but also opens avenues for further research in scalable policy generalization.

3.1 Core contributions

Scalability. The framework leverages parameterized MDPs, where the state space grows with the parameter values. By generalizing from small instances, 1-2-3-Go! avoids the computationally prohibitive task of explicitly exploring the state space for larger instances.

Decision-tree-based representation. Decision trees provide a compact, interpretable, and generalizable representation of policies. Instead of storing explicit state-action mappings, decision trees use predicates over state variables to represent policies symbolically.

Robust generalization. Policies learned from small instances often capture structural regularities that extend to larger instances. This makes 1-2-3-Go! particularly effective for parameterized MDPs where the dynamics scale predictably with the parameters.

4 Counterexample Guided Policy Improvement

Counterexample Guided Policy Improvement (CEGPI) offers a scalable and robust methodology for synthesizing policies in Markov Decision Processes (MDPs) to ensure qualitative properties like almost-sure reachability.

Policy initialization. The process begins by using the 1-2-3-Go! approach to compute “good” / “robust” initial policies for small parameterized instances, which are then generalized using decision tree representations. These trees provide both scalability and interpretability, but the initial policies may fail on larger or more complex instances due to the limitations of learning from simplified scenarios.

Counterexample detection and usage. To address these failures, counterexamples are generated during verification to identify traces where the policy does not satisfy the desired qualitative guarantees. Such counterexamples often arise from issues like loops / failure to progress toward the goal state and reaching bad states. These failures highlight critical shortcomings in the learned policy and guide targeted improvements. The refinement process involves incorporating additional data derived from failed traces and simulations, ensuring that the policy captures the nuances of the system dynamics. For example, if a counterexample reveals a loop caused by a specific decision, the policy is modified to prioritize progress-inducing actions, informed by domain-specific heuristics or local simulations.

Iterative policy refinement. Decision trees are updated with new data while preserving consistency with previously learned behaviors. Local simulations help evaluate alternative actions, preferring those that guarantee progress toward the goal. The refined policy is then tested on larger instances, ensuring it generalizes effectively. If further counterexamples are identified, the process repeats until the policy achieves the desired qualitative properties across the huge intance under consideration.

Benefits of the extension. By leveraging systematic learning and robust generalization policies, the approach ensures scalability with added completeness for qualitative verification. Initialization using 1-2-3-Go! retains the scalability by focusing refinements on critical parts of the policy. This ensures that the generalized policies satisfy qualitative properties across all instances.

5 Conclusion

The combination of counterexample-guided policy improvement with the 1–2–3–Go! approach provides a scalable and robust solution for synthesizing policies in parameterized MDPs. By leveraging the generalizability of decision-tree-based learning to generate effective initial policies and iteratively refining these policies using counterexamples, this approach ensures the satisfaction of qualitative properties. Through this work, we address key limitations in scalability and correctness, which can provide the way for more effective applications in real-world decision-making systems.

References

Ashok, P.; Jackermeier, M.; Kretínský, J.; Weinhuber, C.; Weininger, M.; and Yadav, M. 2021. dtControl 2.0: Explainable Strategy Representation via Decision Tree Learning Steered by Experts. In *TACAS (2)*, volume 12652 of *Lecture Notes in Computer Science*, 326–345. Springer.

Azeem, M.; Chakraborty, D.; Kanav, S.; Křetínský, J.; Mohagheghi, M.; Mohr, S.; and Weininger, M. 2025. 1–2–3–Go! Policy Synthesis for Parameterized Markov Decision Processes via Decision-Tree Learning and Generalization. In Shankaranarayanan, K.; Sankaranarayanan, S.; and Trivedi, A., eds., *Verification, Model Checking, and Abstract Interpretation*, 97–120. Cham: Springer Nature Switzerland. ISBN 978-3-031-82703-7.

Baier, C.; and Katoen, J. 2008. *Principles of model checking*. MIT Press. ISBN 978-0-262-02649-9.

Brázdil, T.; Chatterjee, K.; Chmelik, M.; Forejt, V.; Kretínský, J.; Kwiatkowska, M. Z.; Parker, D.; and Ujma,

M. 2014. Verification of Markov Decision Processes Using Learning Algorithms. In *ATVA*, volume 8837 of *Lecture Notes in Computer Science*, 98–114. Springer.

Chatterjee, K.; Chmelík, M.; and Daca, P. 2015. CEGAR for compositional analysis of qualitative properties in Markov decision processes. *Formal Methods in System Design*, 47(2): 230–264.

Clarke, E.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2003. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5): 752–794.

Feng, L. 2014. *On learning assumptions for compositional verification of probabilistic systems*. Ph.D. thesis, University of Oxford, UK.

Gangopadhyay, B.; and Dasgupta, P. 2021. Counterexample Guided RL Policy Refinement Using Bayesian Optimization. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 22783–22794. Curran Associates, Inc.

Groote, J. F.; Verduzco, J. R.; and de Vink, E. P. 2018. An Efficient Algorithm to Determine Probabilistic Bisimulation. *Algorithms*, 11(9): 131.

Hahn, E. M.; Perez, M.; Schewe, S.; Somenzi, F.; Trivedi, A.; and Wojtczak, D. 2019. Omega-Regular Objectives in Model-Free Reinforcement Learning. In *TACAS (1)*, volume 11427 of *Lecture Notes in Computer Science*, 395–412. Springer.

Hartmanns, A. 2012. MODEST - A unified language for quantitative models. In *FDL*, 44–51. IEEE.

Hartmanns, A.; and Hermanns, H. 2015. Explicit Model Checking of Very Large MDP Using Partitioning and Secondary Storage. In Finkbeiner, B.; Pu, G.; and Zhang, L., eds., *Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Shanghai, China, October 12-15, 2015, Proceedings*, volume 9364 of *Lecture Notes in Computer Science*, 131–147. Springer.

Kamaleson, N. 2018. *Model reduction techniques for probabilistic verification of Markov chains*. Ph.D. thesis, University of Birmingham, UK.

Kwiatkowska, M. Z.; Norman, G.; and Parker, D. 2012. The PRISM Benchmark Suite. In *QEST*, 203–204. IEEE Computer Society.

Li, R.; and Liu, Y. 2019. Compositional stochastic model checking probabilistic automata via symmetric assume-guarantee rule. In *2019 IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA)*, 110–115. IEEE.

Lomuscio, A.; and Pirovano, E. 2019. A Counter Abstraction Technique for the Verification of Probabilistic Swarm Systems. In *AAMAS*, 161–169. International Foundation for Autonomous Agents and Multiagent Systems.

Mitchell, T. 1997. *Machine learning*, volume 1. McGraw-hill New York.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley. ISBN 978-0-47161977-2.

Smolka, S.; Kumar, P.; Kahn, D. M.; Foster, N.; Hsu, J.; Kozen, D.; and Silva, A. 2019. Scalable verification of probabilistic networks. In *PLDI*, 190–203. ACM.