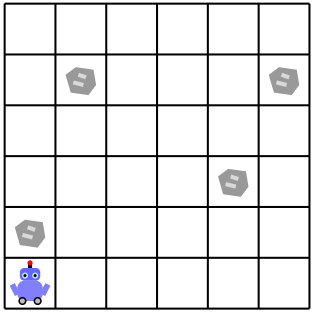
# Constrained and Robust Policy Synthesis with Satisfiability-Modulo-Probabilistic-Model-Checking

*Linus Heck*, Filip Macák, Milan Češka, Sebastian Junges

AAAI 2026
Presented at ACV 2026

Radboud Universiteit


# Example: Constrained and Robust Policies



Grid where robot slips to the left with probability 0.1

Goal: Find a robot policy that provably collects all rocks with probability ≥ 0.9:

- Wherever the rocks are (robust)
- With a simple set of rules (constrained)

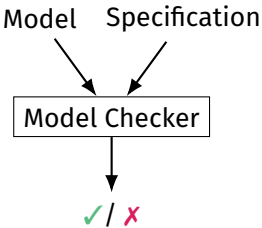
Previous work has been unable to find a robot policy that always works.

At the end of this talk, we will see how we can solve this!

# Overview

1. Why probabilistic model checking?
2. Constrained and robust policy synthesis using SMPMC
3. Solving the robot problem!

# Why Probabilistic Model Checking?




Mechanical Wear

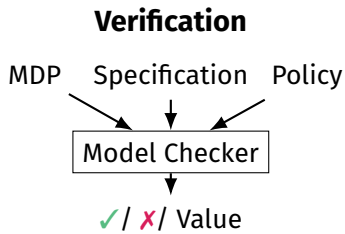
Noisy sensors

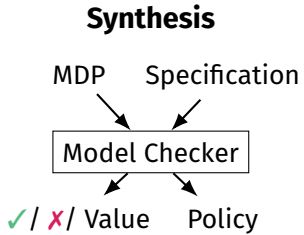
Lossy Channels

A lot of behavior possible, but unlikely.

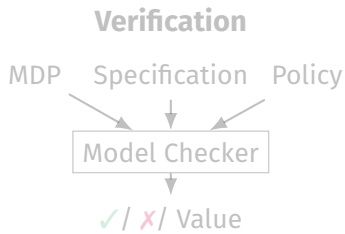
(**Classical) model checking** is insufficient:
cannot distinguish likely from unlikely.

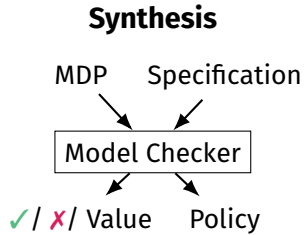
# Probabilistic Verification and Synthesis



# Probabilistic Verification and Synthesis




We want to automatically synthesize policies whose quantitative guarantees hold by design.

# Policy Synthesis for MDPs: Mature Tools

## Compatible Tools

Storm Model Checker

Prism 4.0

The Modest Toolset
Download
Modest mcsta

And many more ……..

## Recommended reading

*Annual Review of Control, Robotics, and Autonomous Systems*

Probabilistic Model Checking and Autonomy

Marta Kwiatkowska,[1] Gethin Norman,[2] and David Parker[3]

Tools at the Frontiers of Quantitative Verification*
QComp 2023 Competition Report

Roman Andriushchenko[1], Alexander Bork[2], Carlos E. Budde[3], Milan Češka[1], Kush Grover[4], Ernst Moritz Hahn[5], Arnd Hartmanns[5,8], Bryant Israelsen[6], Nils Jansen[7], Joshua Jeppson[6], Sebastian Junges[7], Maximilian A. Köhl[8], Bettina Könighofer[9], Jan Křetínský[4,10], Tobias Meggendorfer[4,11,12], David Parker[13], Stefan Pranger[9], Tim Quatmann[2], Enno Ruijters[3], Landon Taylor[6], Matthias Volk[14], Maximilian Weininger[4,11], and Zhen Zhang[8]

The Revised Practitioner's Guide to MDP Model Checking Algorithms

Arnd Hartmanns · Sebastian Junges · Tim Quatmann · Maximilian Weininger

https://sjunges.github.io/files/revised_practitioners_guide.pdf

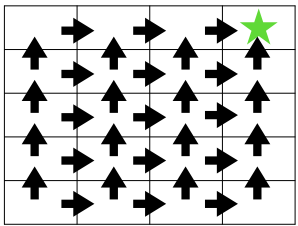
# Policies are not Concise

Grid; actions in cardinal directions, probability to slip.

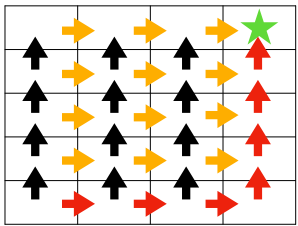
Find a policy that minimises the expected number of steps to ★

Storm

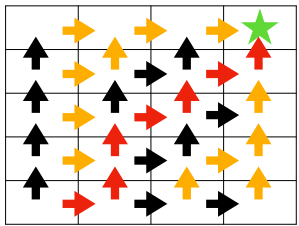
MDP with states (x,y):



Ideal outcome



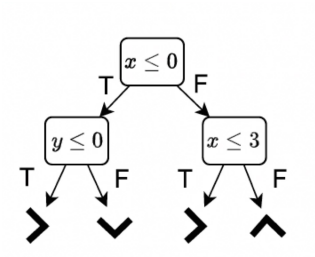
Outcome by model checker



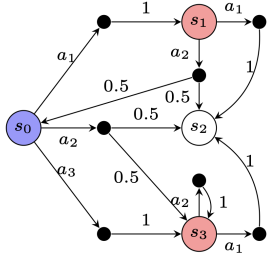
Sometimes, we want policies of a certain shape.

# Constrained Synthesis

We want to constrain our policy space to policies of a certain shape.




Decision trees




Partial observability

```
hole X either { XA is 1 cost 3, 2}
hole Y either { YA is 1, 3 }
hole Z either { 1, 2 }
constraint  !(XA && YA);
module rex
s : [0..3] init 0;
s = 0 -> 0.5: s'=X + 0.5: s'=Y;
s = 1 -> s'=s+Z;
s >= 2 -> s'=s;
endmodule
```

Configurable systems

# SMPMC: Challenges and Approach

Challenge: Constrained search space of policies and probabilities

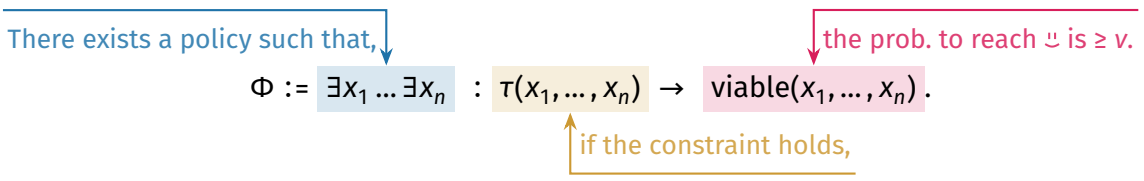
- Monolithic encodings (e.g. MILPs): Very slow at evaluating MDP policies
- Branch & Bound: Cannot reason about arbitrary constraints

Our approach: SMT Solving + Probabilistic Model Checking (SMPMC)

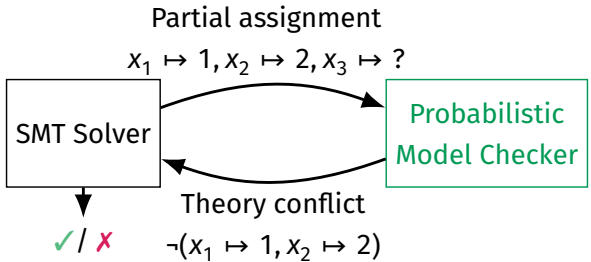
- Search space and constraints: handled by the SMT solver
- Solving underlying MDPs: handled by the probabilistic model checker (e.g. using value iteration)

# SMT Problem Formulation: Constrained Synthesis

Input formula to SMT solver:

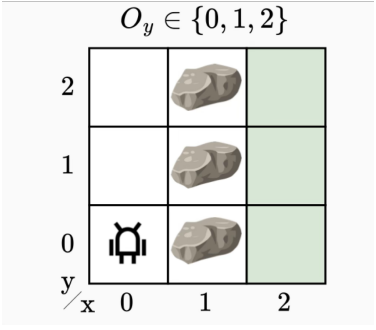


# SMT Solving Meets Probabilistic Model Checking




- Smarter than Guess-And-Check: Check partial assignments, prune early
- More flexible than Branch & Bound: SMT solver handles search space

# Robust Synthesis

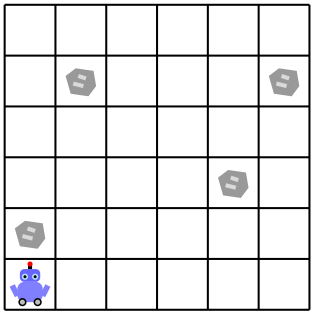



Multi-environment MDPs

```
hole X either { XA is 1 cost 3, 2}
hole Y either { YA is 1, 3 }
hole Z either { 1, 2 }
constraint  !(XA && YA);
module rex
s : [0..3] init 0;
s = 0 -> 0.5: s'=X + 0.5: s'=Y;
s = 1 -> s'=s+Z;
s >= 2 -> s'=s;
endmodule
```

Configurable Systems

# Example: Constrained and Robust Policies



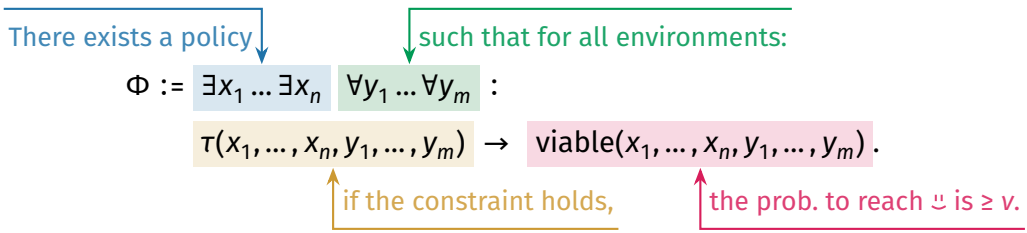
Grid where robot slips to the left with probability 0.1

Goal: Find a robot policy that provably collects all rocks with probability ≥ 0.9:

- Wherever the rocks are (robust)
- With a simple set of rules (constrained)

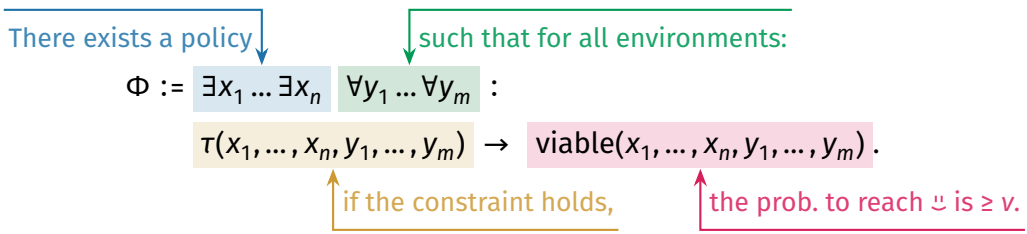
# SMT Problem Formulation: Constrained and Robust Synthesis

Input formula to SMT solver:



# SMT Problem Formulation: Constrained and Robust Synthesis

Input formula to SMT solver:




SMT solver uses model-based quantifier instantiation.

# Experiments

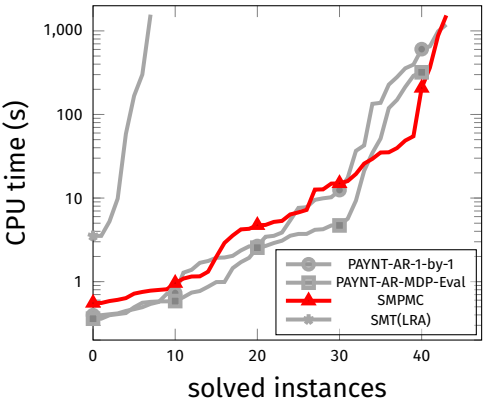
- 372 benchmarks from different domains: Sets of MCs, POMDPs with FSCs, Multi-Environment MDPs
- Comparisons: PAYNT (state-of-the-art for policy synthesis), monolithic SMT (baseline)

Q1: Is SMPMC competitive with the state of the art at policy synthesis?

Q2: Can SMPMC handle both robustness and arbitrary constraints?

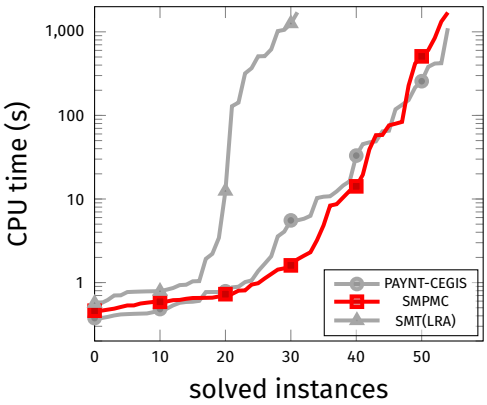
# Results: Comparisons Against Baselines

Robust synthesis:




⇒ competitive with Branch & Bound!

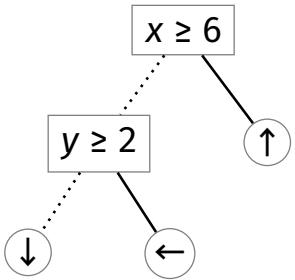
Constrained synthesis (single env.):

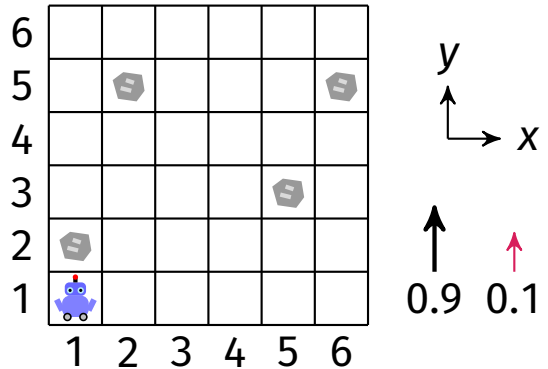



⇒ competitive with Guess-And-Check!

# Solving Rocks with Constrained and Robust Synthesis

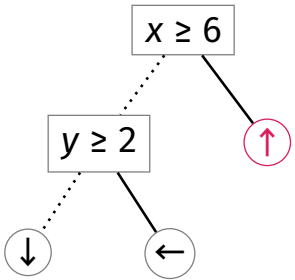
SMPMC finds a five-node decision tree that picks up all rocks in a spiral:

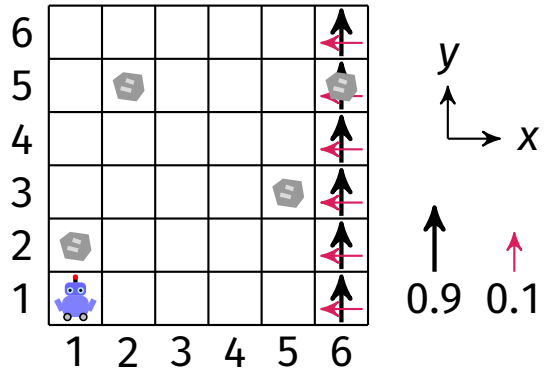


# Solving Rocks with Constrained and Robust Synthesis

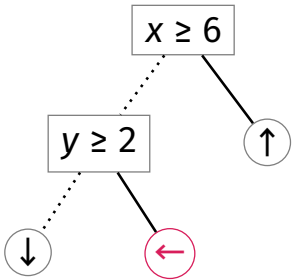
SMPMC finds a five-node decision tree that picks up all rocks in a spiral:

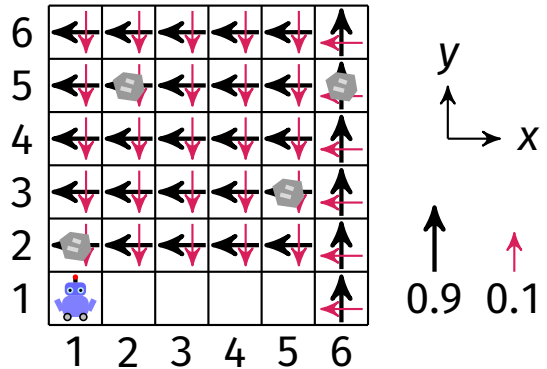


# Solving Rocks with Constrained and Robust Synthesis

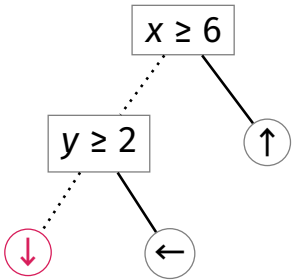
SMPMC finds a five-node decision tree that picks up all rocks in a spiral:

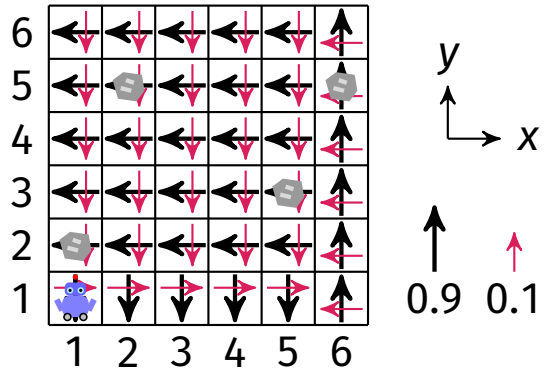


# Solving Rocks with Constrained and Robust Synthesis

SMPMC finds a five-node decision tree that picks up all rocks in a spiral:



# Contributions

- SMPMC: A novel, versatile, and efficient approach that tightly integrates satisfiability solvers with probabilistic model checking
- First effective approach that constructs robust decision tree policies
- Competitive with state-of-the-art heuristics, solves new problems


# Appendix

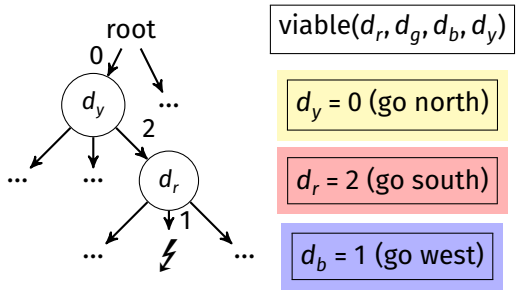



**Figure:** Sketch of CDCL.

Plain synthesis: