Verified Solution Methods for Markov Decision Processes

Maximilian Schäffeler

TU Munich, RTG ConVeY

Abstract

We formally verify executable algorithms for solving Markov decision processes (MDPs) in the interactive theorem prover Isabelle/HOL. Then, we verify executable dynamic programming algorithms and approximate algorithms using linear programming certification. Currently, we are building verified certificate checkers for quantitative model checking.

Motivation

- **Markov decision processes (MDPs)** model systems where an agent maximizes rewards under uncertainty
- Application areas are often **safety-critical** and real-world scenarios: planning, reinforcement learning, model checking, operations research

⇒ Correct and robust software is important
⇒ We use **interactive theorem provers** to develop trustworthy software for MDPs

Isabelle/HOL

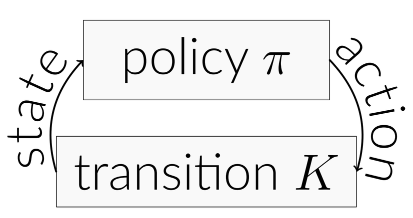
- We use **Isabelle/HOL** for our developments
- Isabelle/HOL is an interactive theorem prover
- Math library + AFP: probability theory, limits, etc.
- Powerful automation: *sledgehammer* and *auto*

- **Code extraction** ⇒ verified executables
- **Highly trustworthy**: high-level proofs ⇒ axioms

Markov Decision Processes

An MDP consists of

- sets of **states** S and **actions** A_s for each state
- **transition probabilities** $K : S \times A \to P(S)$
- **rewards** $r : S \times A \to \mathbb{R}$
- **discount factor** $\lambda \leq 1$



- **Goal**: maximize reward by optimizing a policy
- A **policy** selects an action based on the states and actions observed

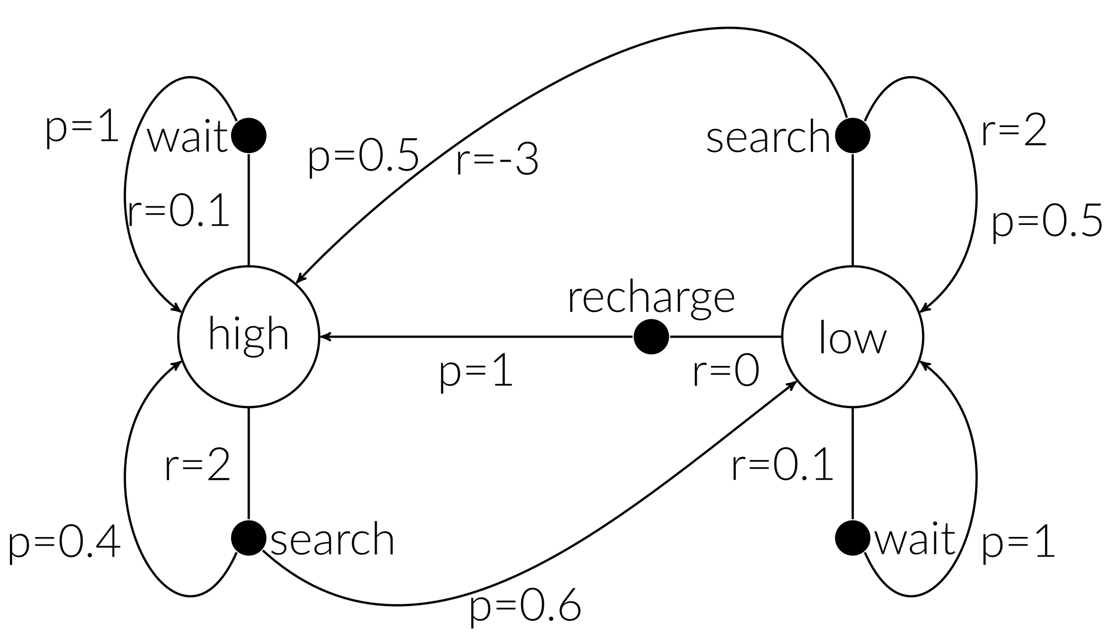


Figure 1. The robot can choose between searching for treasure, waiting, or recharging. States indicate battery levels.

Expected Total Reward

- Expected total discounted reward ν_π is the cumulative reward collected during a run:

$$\nu_\pi(s) = \lim_{n\to\infty} \mathbb{E}_{\omega\sim T_\pi(s)} \left[\sum_{i<n} \lambda^i \cdot r(\omega_i)\right]$$

- The discount factor makes the agent **short-sighted**

Dynamic Programming

- The **Bellman optimality operator**

$$\mathcal{L}(v) = \sup_\pi r_\pi + \lambda \mathcal{P}_\pi v$$

converges to the **optimal value** ν^*:

$$\lim_{i\to\infty} \mathcal{L}^i(v) = \nu^*.$$

- We formalize
  - (Gauss-Seidel) Value iteration
  - (Modified) Policy Iteration
- Refine algorithms to efficiently executable code
- Faster:
  - **begin with value iteration on floats**,
  - then one iteration with precise arithmetic

Value Iteration in Isabelle/HOL

```
function value_iteration ::
  ‹real ⇒ ('s ⇒b real) ⇒ ('s ⇒b real)›
where
  "value_iteration eps v = (
  if 2 * l * dist v (ℒb v) < eps * (1-l) ∨ eps ≤ 0
  then ℒb v
  else value_iteration eps (ℒb v))"
```

Benchmarks

- Benchmark problems:
  International Planning Competition 2018
- Precise arithmetic is expensive, certification is much more competitive

		VI			GS			Certif.	
			Verified			Verified			
Domain	Instances	Prism	$\mathbb{R}$	$\mathbb{F}$	Prism	$\mathbb{R}$	$\mathbb{F}$	VI	GS
traffic	4	4	4	4	4	2	4	4	4
elevators	8	5	2	6	5	1	6	6	6
game-of-life	3	3	–	3	3	–	3	3	–
manufacturer	2	2	–	2	2	–	2	2	2
luck	5	5	5	5	5	5	5	5	5
skill-teaching	8	6	4	7	6	3	7	6	6
tireworld	6	4	4	4	4	4	4	4	4
wildlife	8	6	5	8	6	4	8	8	8

Table 1. Table with number of instances solved by different algorithms. Columns 2,3 show the performance of PRISM vs. our implementations: $\mathbb{R}, \mathbb{F}$ indicate precise or floating-point arithmetic. Column 4 displays the performance of our certification approach.

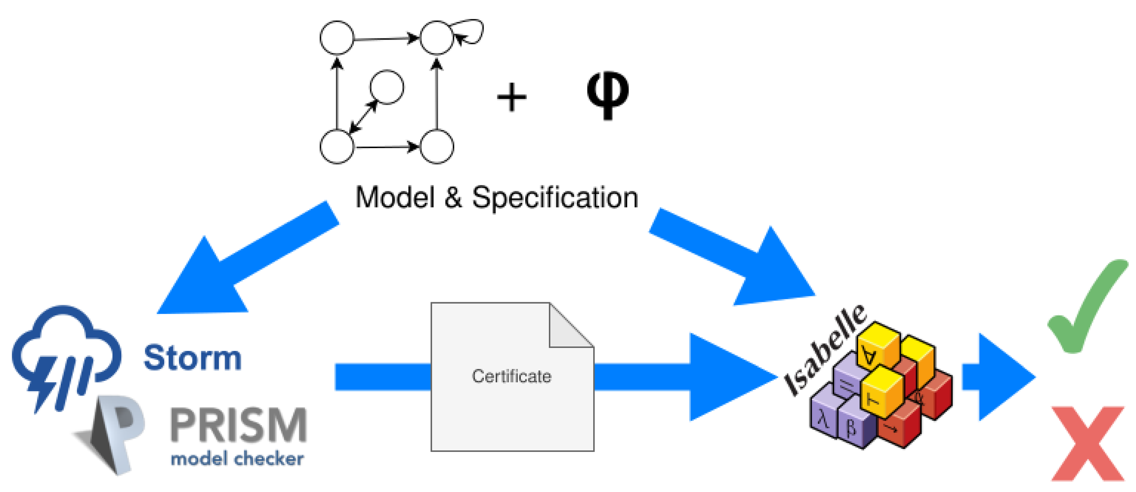
Example Proof

```
lemma contraction_L: "dist (L p v) (L p u) ≤ l * dist v u"
proof -
  have aux: "L p v s - L p u s ≤ l * dist v u"
    if lea: "(L p v s) ≥ (L p u s)" for v s u
  proof -
    have "L p v s - L p u s = l *R (𝒫₁ p v - 𝒫₁ p u) s"
      by (simp add: L_def algebra_simps)
    also have "... ≤ l * ¦𝒫₁ p (v - u) s¦"
      by (auto simp: blinfun.diff_right)
    also have "... ≤ l * norm (𝒫₁ p (v - u))"
      using abs_le_norm_bfun by fastforce
    also have "... ≤ l * dist v u"
      by (auto simp: 𝒫₁.rep_eq  dist_norm)
    finally show ?thesis
      by auto
  qed
  have "dist (L p v s) (L p u s) ≤ l * dist v u" for s
    using aux aux[of v _ u] by (cases "L p v s ≥ L p u s")
      (auto simp: dist_real_def dist_commute)
  thus "dist (L p v) (L p u) ≤ l * dist v u"
    by (simp add: dist_bound)
qed
```

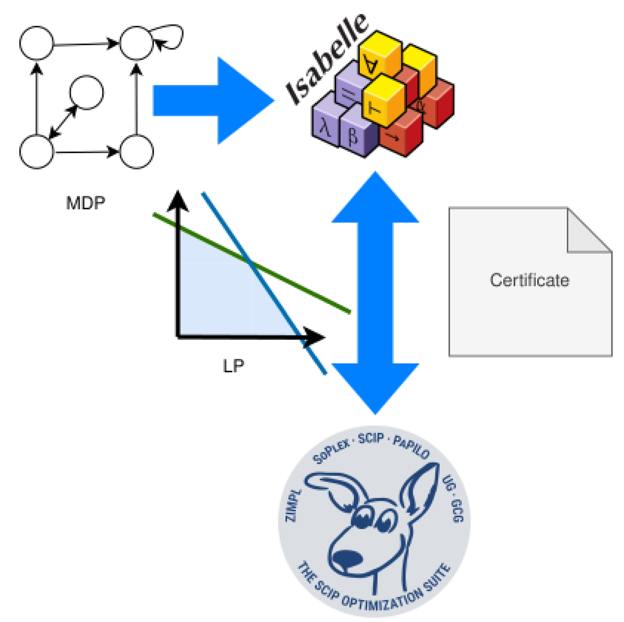
Conclusion

- ITPs are suitable for developing executable algorithms for reasoning under uncertainty
- Powerful proof automation + a large library of formal proofs were essential
- Certification is key for large state spaces
- Future work: Floating-point guarantees, Monte-Carlo algorithms

Quantitative Model Checking



Linear Programming Certification



References

[1] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored mdps. *J. Artif. Intell. Res.*, 19:399–468, 2003.

[2] Maximilian Schäffeler and Mohammad Abdulaziz. Formally verified solution methods for markov decision processes. In *AAAI 2023*.

[3] Maximilian Schäffeler and Mohammad Abdulaziz. Verified algorithms for solving markov decision processes. *Archive of Formal Proofs*, December 2021. `https://isa-afp.org/entries/MDP-Algorithms.html`, Formal proof development.

Funding This work was supported by the Research Training Group GRK 2428 CONVEY of the German Research Council (DFG)