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 \( A \) for each state
- transition probabilities \( K: S \times A \times P(S) \)
- rewards \( r: S \times A \rightarrow \mathbb{R} \)
- 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.

Dynamic Programming

- The Bellman optimality operator
  \[ L^v(s) = \min \{ r(s, a) + \lambda \sum_{s'} K(s, a, s') \mid a \in A(s) \} \]
  converges to the optimal value \( v^* \):
  \[ \lim_{n \to \infty} L^n(v) = v^* \].
- 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

\[
\text{function value_iterations :} \ (s \Rightarrow (\text{real} \Rightarrow (s \Rightarrow \text{real}))) \\
\text{where} \\
\text{case value_iterations } v \text{ of} \\
\text{if } 2 \times v \cdot \text{eps} \leq (1-\lambda) \cdot v \text{ then } v, \\
\text{else } v \\
\text{end}
\]

Benchmarks

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

<table>
<thead>
<tr>
<th>Domain</th>
<th>Value Iteration</th>
<th>Policy Iteration</th>
<th>GS Proof</th>
<th>Certif.</th>
</tr>
</thead>
<tbody>
<tr>
<td>traffic</td>
<td>4</td>
<td>4</td>
<td>F</td>
<td>4</td>
</tr>
<tr>
<td>elevator</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>game-of-life</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>manufacturer</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>luck</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>skill-teaching</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>tireworld</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>wildlife</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

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

References


Funding This work was supported by the Research Training Group GRK 2438 CONVEY of the German Research Council (DFG).

https://github.com/schaeffeler/mdps-isabelle-hol
Formally Verified Solution Methods for Markov Decision Processes
maximilian.schaeffeler@tum.de

Example Proof

Lemma contraction L "dist (L π v) v ≤ dist v v'" proof

have aux: "L π v · L π v' ≤ dist v v'
if lazy: "L π v · L π v' ≤ dist v v'" proof

have "L π v · L def algebra simp" by (simp add L_def algebra_simps)
also have "L π v · L dist v v'" by (auto simp blinfom.dist_right)
also have "L π v · L norm (v' (v · dist v v'))" using abs_norm_bfn by fastforce
also have "L π v · L dist v v'" by (auto simp: L_def algebra_simps)
finally show Thesis
by auto
qed

"dist (L π v) v ≤ dist v v'" for s using aux of (v u) by cases "L π v · L π v'" (auto simp dist_def dist commute) thus "dist (L π v) v ≤ dist v v'" by (simp add: dist_bound)

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