Who Verifies the Verifier? *Certificates for Probabilistic Model Checking*

Tobias Winkler UnRAVeL Spring WS 2025





Parts of this presentation are based on a TACAS '25 paper with Krishnendu Chatterjee, Tim Quatmann, Maximilian Schäffeler, Maximilian Weininger, and Daniel Zilken.





Part I: Certifying Algorithms

The Implementation Problem

f: $X \rightarrow Y$ specification or abstract algorithm.

The Implementation Problem

$f: X \rightarrow Y$ specification or abstract algorithm.



provides

You or some other programmer(s)



1 1 1

The Implementation Problem





• Bugs (wrong implementation of a right idea)



- Bugs (wrong implementation of a right idea)
- Right implementation of a wrong idea



- Bugs (wrong implementation of a right idea)
- Right implementation of a wrong idea
- Errors due to inexact floating point arithmetic



- Bugs (wrong implementation of a right idea)
- Right implementation of a wrong idea
- Errors due to inexact floating point arithmetic
- Unsound 3rd party software (e.g. LP solver, compiler, ...)



- Bugs (wrong implementation of a right idea)
- Right implementation of a wrong idea
- Errors due to inexact floating point arithmetic
- Unsound 3rd party software (e.g. LP solver, compiler, ...)
- Testing can only cover small number of instances and is costly.





Idea: *w* is an easy-to-check proof that y = f(x).









Requirements: $\forall x, y$:





Requirements: $\forall x, y$:

• $(\exists w : c(x, y, w) = True) \implies f(x) = y$



(soundness)



Requirements: $\forall x, y$:

• $(\exists w: c(x, y, w) = True) \implies f(x) = y$

• $(\exists w: c(x, y, w) = True) \iff f(x) = y$



(soundness)

(completeness)



Requirements: $\forall x, y$:

- $(\exists w: c(x, y, w) = True) \implies f(x) = y$
- $(\exists w: c(x, y, w) = True) \iff f(x) = y$

• *w* is "small" and *c* is "trivial/easy to implement"



(soundness)

(completeness)

(informal requirements)

Example: Certificates for Shortest Paths

- Input: Directed graph (V, E) with two special vertices $s, t \in V$.



• Output: Length d(s, t) of shortest path from s to t, or ∞ if no such path exists.

d(s,t) = 2

Example: Certificates for Shortest Paths

- Input: Directed graph (V, E) with two special vertices $s, t \in V$.
- Output: Length d(s, t) of shortest path from s to t, or ∞ if no such path exists.
- Certificate:
 - ♦ Data: For *every* $v \in V$ a number d(v, t)
 - \checkmark Verification condition: Check that d(t, t) = 0 and for all $v \in V \setminus \{t\}$

 - $d(v, t) = 1 + \min d(v', t)$ $(v,v') \in E$

Example: Certificates for Shortest Paths



• Certificate:

♦ Data: For *every* $v \in V$ a number d(v, t)

✓ Verification condition: Check that d(t, t) = 0 and for all $v \in V \setminus \{t\}$

$d(v, t) = 1 + \min d(v', t)$ $(v,v') \in E$





• Input: Undirected graph G = (V, E)





- Input: Undirected graph G = (V, E)
- **Output:** *True* if *G* is bipartite, otherwise *False*.





- Input: Undirected graph G = (V, E)
- **Output:** *True* if *G* is bipartite, otherwise *False*.
- **Hint:** *G* is bipartite \iff *G* does *not* have an odd-length cycle.





- Input: Undirected graph G = (V, E)
- **Output:** *True* if *G* is bipartite, otherwise *False*.
- **Hint:** *G* is bipartite \iff *G* does *not* have an odd-length cycle.
- Certificate:





- Input: Undirected graph G = (V, E)
- **Output:** *True* if *G* is bipartite, otherwise *False*.
- **Hint:** G is bipartite \iff G does *not* have an odd-length cycle.
- Certificate:
 - ♦ Data: For output *True* a vertex color assignment $\kappa: V \rightarrow \{r, b\}$, for output *False* a vertex sequence $v_1 \dots v_n$.





- Input: Undirected graph G = (V, E)
- **Output:** *True* if *G* is bipartite, otherwise *False*.
- **Hint:** G is bipartite \iff G does *not* have an odd-length cycle.
- Certificate:
 - ♦ Data: For output *True* a vertex color assignment $\kappa: V \rightarrow \{r, b\}$, for output *False* a vertex sequence $v_1 \dots v_n$.



✓ Verification condition: If output *True* check κ is valid 2-coloring, if output *False* check $v_1 \dots v_n$ is an odd-length cycle.



• Enable automated testing.

- Enable automated testing.
- Prove correctness of closed-source content of checkers, ...) to customers.

• Prove correctness of closed-source commercial software (SAT-solvers, model

- Enable automated testing.
- checkers, ...) to customers.
- Provide reference results for standardized benchmark sets.

• Prove correctness of closed-source commercial software (SAT-solvers, model

Part II: Certificates for Probabilistic Model Checking

• MDP \approx transition system + probabilities

- MDP \approx transition system + probabilities
- **Example:** Writing a PhD thesis with p = 100 pages:

- MDP \approx transition system + probabilities
- **Example:** Writing a PhD thesis with p = 100 pages:



- MDP \approx transition system + probabilities
- **Example:** Writing a PhD thesis with p = 100 pages:



 ≈ 0.819 , trust me \bigcirc

- Input: MDP with states S, initial state $s_0 \in S$, target state $t \in S$
- Output: $Pr_{s_0}^{\max}(\langle t \rangle)$, the maximal probability to reach *t* from s_0

- Input: MDP with states S, initial state $s_0 \in S$, target state $t \in S$
- Output: $Pr_{s_0}^{\max}(\langle t \rangle)$, the maximal probability to reach *t* from s_0
- Certificate:

- Input: MDP with states S, initial state $s_0 \in S$, target state $t \in S$
- Output: $Pr_{s_0}^{\max}(\langle t \rangle)$, the maximal probability to reach *t* from s_0
- Certificate:

♦ Data: For every $s \in S$ the probability $Pr_s^{\max}(\Diamond t)$ & shortest distance d(s, t)

- Input: MDP with states S, initial state $s_0 \in S$, target state $t \in S$
- **Output:** $Pr_{s_0}^{\max}(\Diamond t)$, the maximal probability to reach *t* from s_0
- Certificate:

 - ♦ Data: For every $s \in S$ the probability $Pr_s^{\max}(\Diamond t)$ & shortest distance d(s, t)✓ Verification condition: Check that $Pr_t^{\max}(\diamondsuit t) = 1$ and for all $s \in S \setminus \{t\}$

- Input: MDP with states S, initial state $s_0 \in S$, target state $t \in S$
- Output: $Pr_{s_0}^{\max}(\langle t \rangle)$, the maximal probability to reach *t* from s_0
- Certificate:
 - $\sum p(s, \alpha, s') \cdot Pr_{s'}^{\max}(\diamondsuit t)$ $Succ(s,\alpha)$
 - ♦ Data: For every $s \in S$ the probability $Pr_s^{\max}(\Diamond t)$ & shortest distance d(s, t)✓ Verification condition: Check that $Pr_t^{\max}(\diamondsuit t) = 1$ and for all $s \in S \setminus \{t\}$

$$\Pr_{s}^{\max}(\diamondsuit t) = \max_{\alpha} s' \in$$

- Input: MDP with states S, initial state $s_0 \in S$, target state $t \in S$
- Output: $Pr_{s_0}^{\max}(\langle t \rangle)$, the maximal probability to reach *t* from s_0
- Certificate:
 - $\sum p(s, \alpha, s') \cdot Pr_{s'}^{\max}(\diamondsuit t)$ $Succ(s,\alpha)$
 - ♦ Data: For every $s \in S$ the probability $Pr_s^{\max}(\Diamond t)$ & shortest distance d(s, t)✓ Verification condition: Check that $Pr_t^{\max}(\diamondsuit t) = 1$ and for all $s \in S \setminus \{t\}$

$$\Pr_{s}^{\max}(\diamondsuit t) = \max_{\alpha} s' \in$$

 $Pr_s^{\max}(\diamondsuit t) > 0 \implies d(s,t) < \infty$

and

Extensions Check out our TACAS '25 paper!

- Approximate probabilities
- *Expected rewards*
- Both *minimization* and maximization
- *Qualitative* properties (e.g. certify $Pr_{s_0}^{\max}(\diamondsuit t) < 1$)











Fixed Point Certificates for Reachability and Expected Rewards in MDPs^{*}

Krishnendu Chatterjee¹[®], Tim Quatmann²[®], Maximilian Schäffeler³[®], Maximilian Weininger¹, Tobias Winkler², and Daniel Zilken^{1,2}

¹ Institute of Science and Technology Austria, Klosterneuburg, Austria {Krishnendu.Chatterjee, Maximilian.Weininger}@ist.ac.at ² RWTH Aachen, Germany

{tim.quatmann, tobias.winkler, daniel.zilken}@cs.rwth-aachen.de ³ Technical University of Munich, Germany · maximilian.schaeffeler@tum.de

Abstract. The possibility of errors in human-engineered formal verification software, such as model checkers, poses a serious threat to the purpose of these tools. An established approach to mitigate this problem are *certificates*—lightweight, easy-to-check proofs of the verification results. In this paper, we develop novel certificates for model checking of Markov decision processes (MDPs) with quantitative reachability and expected reward properties. Our approach is conceptually simple and relies almost exclusively on elementary fixed point theory. Our certificates work for arbitrary finite MDPs and can be readily computed with little overhead using standard algorithms. We formalize the soundness of our certificates in Isabelle/HOL and provide a formally verified certificate checker. Moreover, we augment existing algorithms in the probabilistic model checker Storm with the ability to produce certificates and demonstrate practical applicability by conducting the first formal certification of the reference results in the Quantitative Verification Benchmark Set.

Keywords: Probabilistic model checking · Markov decision processes · Certificates · Reachability · Expected rewards · Proof assistant

1 Introduction

Markov decision processes (MDPs) [48,7,5] are cision making in probabilistic environments. The frequently require computing reachability probabili system state, as well as the *expected rewards* (or ing so. MDP model checking amounts to computin

^{*} This project has received funding from the ERC the Austrian Science Fund (FWF) 10.55776/COE1 Ministerium für Kultur und Wissenschaft NRW, th VeY), the EU's Horizon 2020 research and innovation Sklodowska-Curie grant agreement Nos. 101034413 (MISSION), and the DFG RTG 2236 (UnRAVeL). Ex computing resources granted by RWTH Aachen Uni



• "What if there is an *error in the theory* and validity of the verification conditions does not actually imply correctness of the result?"

- "What if there is an *error in the theory* and validity of the verification conditions does not actually imply correctness of the result?"
- "What if there is a *bug in the certificate checker*" implementation?"

- "What if there is an *error in the theory* and validity of the verification conditions does not actually imply correctness of the result?"
- "What if there is a bug in the certificate checker implementation?"
- → We have formalized all the theory in *Isabelle/HOL*. Based on that, we have generated a *formally verified* certificate checker.

- "What if there is an *error in the theory* and validity of the verification conditions does not actually imply correctness of the result?"
- "What if there is a *bug in the certificate checker* implementation?"
- → We have formalized all the theory in *Isabelle/HOL*. Based on that, we have generated a *formally verified* certificate checker.



• \approx 450 MDPs from the literature

- \approx 450 MDPs from the literature
- Most instances have 1K 10M states.

- ≈ 450 MDPs from the literature
- Most instances have 1K 10M states.
- Most benchmarks model practically relevant problems (communication protocols, scheduling problems, ...).

- ≈ 450 MDPs from the literature
- Most instances have 1K 10M states.
- Most benchmarks model practically relevant problems (communication protocols, scheduling problems, ...).

• For each instance we certify a reachability probability or an expected reward.



Compute result + certificate



• Runtime overhead often within factor 2

Compute result + certificate



• Runtime overhead often within factor 2



Compute result + certificate



- Runtime overhead often within factor 2
- Generated certificates for $\approx 350/450$ benchmarks within 900s time limit

Empirical Evaluation Scalability of the formally verified certificate checker?



- Reasonable performance up to $\approx 1M$ states
- *Parsing* the input MDP and the certificate is currently a bottleneck.

Empirical Evaluation What is the total overhead of certified MDP verification?



• Often within factor ≈ 4 times slower, but some additional timeouts

Lessons Learned and Takeaways

• Trustworthiness is a spectrum; reaching 100% is extremely difficult.

nothing testing certificates

- But uncertainties remain: Bugs in formalization, compilers, hardware, etc.
- Consider certificates for your next algorithm.
 - If infeasible, consider certifying individual modules, subroutines, etc.

ed Point Certificates for Reachability an xpected Rewards in MDPs



Lessons Learned and Takeaways

• Trustworthiness is a spectrum; reaching 100% is extremely difficult.

nothing testing certificates

- But uncertainties remain: Bugs in formalization, compilers, hardware, etc.
- Consider certificates for your next algorithm.
 - If infeasible, consider certifying individual modules, subroutines, etc.

ed Point Certificates for Reachability an xpected Rewards in MDPs



Thanks for listening!