Paper Review 2:
The probabilistic model checker STORM

Ivo Melse, s1088677

April 24, 2025

1 Objective

Markov models are used to model state-like systems where (statistical) uncertainty plays a role. Once we have such a model, it becomes relevant to ask whether the model satisfies a certain property. For example, we can verify if the system is never able to reach a state where a safety hazard occurs. Over the last decades, various tools have been proposed that allow us to perform this model checking. One of these tools is STORM, which was introduced in a previous paper in 2017 [2]. In the current paper [3], the authors of STORM provide an introduction to STORM, and then evaluate and compare it to other model checkers.

Before STORM was introduced, the state of the art was the PRISM model checker. PRISM provided superior performance to its competitors, and continues to be highly influential. However, the performance of PRISM is limited due to the fact that it was written in the Java programming language.

2 Proposal

The authors set out to implement a model checker that would be more efficient than PRISM. A secondary goal was to make the new tool modular and extendable, so that it would be relatively easy to maintain and extend.

C++ was chosen as the programming language because of its reduced overhead and access to low-level functionality that can be used for optimizations. In brief, STORM has a number of *engines*, which are units that can solve model checking problems. Some of these are *explicit*, some are *symbolic* and others are a combination. In explicit engines all transitions are explicitly loaded in memory as transition matrices, while in symbolic engines BDD-like structures are used to represent models.

Each engine depends on a number of *solvers* for smaller sub-problems. The relation between engines and solvers is summarized in Figure 1.

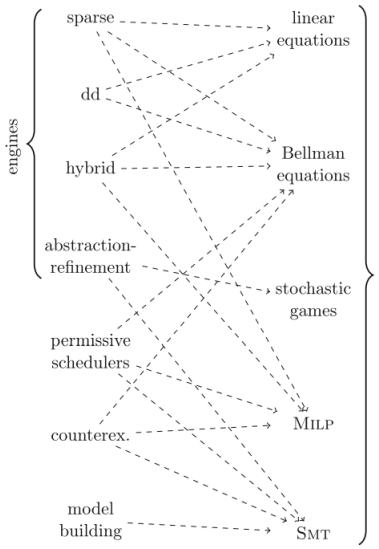


Figure 1: Summary of engines and solvers in STORM

STORM can be used as a library in C++ or a CLI application. The authors also provide Python bindings for STORM, called StormPy. Models can be specified using PRISM or Jani files, or using C++ or Python code.

3 Evidence

The paper presents two kinds of empirical evidence. First, the results of QComp 2019 are presented. Qcomp is a competition for probabilistic model checkers. In 2019, they tested four model checkers (EPMC, MCSTA, PRISM, STORM) across 100 benchmarks. Not all instances were supported by all model checkers, but STORM supports the most instances (96/100), see table 1.

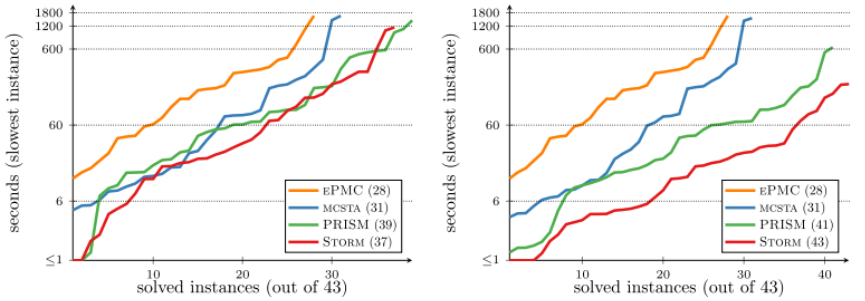
Model checker	Supported instances (out of 100)
STORM	96
EPMC	63
PRISM	58
MCSTA	86
All	43

Table 1: Supported instances per model checker in QComp 2019

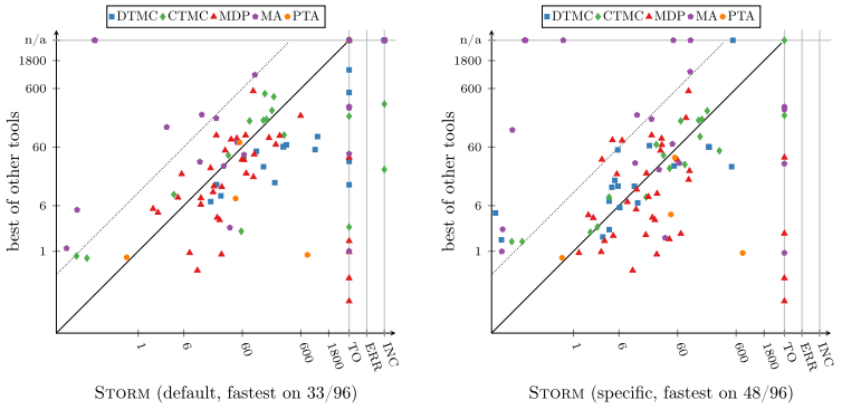
The 43 models that were supported by all four model checkers were executed and their run time was measured, see Figure 2. At the top of the Figure (a and b), the run times of the four participating model checkers are compared. On the horizontal axis, we have the fastest instances that were solved by each model checker. Note that the 20 fastest instances for one model checker might be

different from the 20 fastest instances for another model checker. In Figure 2 a, the measurements were done with default settings while in 2 b, the developers were asked to manually set the best settings in order to optimize the performance for this particular instance. We can see that STORM performs similar to PRISM with default settings, but outperforms the other participants significantly with optimized settings.

At the bottom of the figure (b and c), the performance of STORM is compared to the best other participating model checkers for each instance. This means that if an instance is left of the line, STORM performs better than PRISM, EPMC and MCSTA. Likewise, if an instance is right of this line, that means that there was another model checker that performed better than STORM for this instance. Once again, in Figure 2 c, default settings are used while in Figure 2 d, optimized settings are used.



(a) Quantile plots for the general-purpose model checkers (default) **(b)** Quantile plots for the general-purpose model checkers (specific). Repeats Figure 1.



(c) Runtime of Storm (default) compared with the best results. **(d)** Runtime of Storm (best config) compared with the best results.

Figure 2: Results of QComp 2019

4 Shoulders of giants

The theoretical basis of probabilistic Markov models can be attributed to the Russian mathematician Andrey Andreyevich Markov [4] who first described finite state models with probabilistic transitions in the early 1900s. Model checking as a technique was independently developed by Clarke and Emerson and Queille and Sifakis in the early 1980s. The technique has been broadly successful. Clark and Wang attribute this success to its automated nature as opposed to theorem proving, and the fact that its relatively easy to use for non-experts. Over the last decades, there has been a lot of research on model checking, and a myriad of techniques and optimizations for different settings have been developed [1].

The authors build on their own previous work where they introduce STORM for the first time in 2017 [2]. STORM is by no means the first probabilistic model checker. Its most important competitors and predecessors are PRISM, MRMC, YMER, EPMC, and MCSTA. These were a source of inspiration during the development of STORM. In particular, the most common input format is adopted from PRISM.

5 Impact

The paper seems very influential. It has been cited 284 times since its publication in 2022. In the field of model checking, STORM is widely considered the best model checker that is available in terms of performance. STORM is widely used by computer science researchers worldwide, including at Radboud University. Most of the citations seem to use STORM for verifying a system in a specific (theoretical or practical) setting. In other citations, STORM is used as a solver to accomplish a specific task within a bigger system, or STORM is extended.

6 Writing

The writing of this paper is clear and accessible. The paper is divided up quite logically into sections which each have a sensible title. The authors provide a short background section about probabilistic model checking which is nice for readers who are not already familiar with the theoretical background.

7 Unresolved issues

The paper does not mention any unresolved issues. This could be considered a shortcoming. I could come up with some unsolved issues myself.

First, the performance of STORM can be further improved. It might even be possible to leverage GPUs for this in future iterations.

Second, currently STORM is quite hard to use for new and inexperienced users. It can be quite challenging to learn how to specify models in the required input formats. It would be great if there was some educational tool on top of STORM that is more accessible.

References

[1] Edmund M Clarke and Qinsi Wang. "Years of Model Checking". In: *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*. Springer. 2014, pp. 26–40.

[2] Christian Dehnert et al. "A storm is coming: A modern probabilistic model checker". In: *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II 30*. Springer. 2017, pp. 592–600.

[3] Christian Hensel et al. "The probabilistic model checker Storm". In: *International Journal on Software Tools for Technology Transfer* (2022), pp. 1–22.

[4] Guy Leonard Kouemou and Dr Przemyslaw Dymarski. "History and theoretical basics of hidden Markov models". In: *Hidden Markov models, theory and applications* 1 (2011).