BEEM: Benchmarks for Explicit Model Checkers

Radek Pelánek [⋆]

Department of Information Technologies, Faculty of Informatics
Masaryk University Brno, Czech Republic
xpelanek@fi.muni.cz

Abstract. We present Beem — BEnchmarks for Explicit Model checkers. This benchmark set includes more than 50 parametrized models (300 concrete instances) together with their correctness properties (both safety and liveness). The benchmark set is accompanied by an comprehensive web portal, which provides detailed information about all models. The web portal also includes information about state spaces and facilities for selection of models for experiments.
The address of the web portal is `http://anna.fi.muni.cz/models`.

1 Introduction

The model checking field underwent a rapid development during last years. Several new, sophisticated techniques have been developed, e.g., symbolic methods, bounded model checking, or automatic abstraction refinement. However, for several important application domains we cannot do much better than the basic explicit model checking approach — brute force exhaustive state space search. This technique is used by several of the most well-known model checkers (e.g., Spin, Murphi). The application scope of the explicit technique has been extended significantly by progress in computer speed and algorithmic improvements and many realistic case studies showed practical usability of the method. Even some of the software model checkers (e.g., Java PathFinder, Zing) are based on the explicit search.

There is also a significant body of research work devoted to the improvement of explicit model checking. Unfortunately, many papers fail to convincingly demonstrate the usefulness of newly presented techniques. In order to perform high quality experimental evaluation, researchers need to have access to:

- tool in which they can implement model checking techniques,
- benchmark set of models which can be used for comparisons.

At the moment, there is a large number of model checking tools (see [4]), but the availability of benchmark sets is rather poor. The aim of this work is to contribute to the progress in this direction. We present Beem — a new benchmark set with a web portal.

[⋆] Partially supported by GA ČR grant no. 201/07/P035.

This short paper presents the main rationale and design choices behind Beem. Detailed documentation is given in a technical report [10], which presents description of the modeling language and used models, functionality and realization of the web portal, and an example of an experimental application over the set.

2 Experimental Work in Model Checking

In order to support the need for benchmarks, we present an evaluation of experiments in model checking papers. We have used a sample of model checking publications; experiments in each of these publications were classified into one of the following five categories:

Q1 Random inputs or few toy models.
Q2 Several toy models (possibly parametrized) or few simple models.
Q3 Several simple models (possibly parametrized) or one large case study.
Q4 An exhaustive study of parametrized simple models or several case studies.
Q5 An exhaustive study with the use of several case studies.

Table 1. presents the quality of experiments in papers from our sample (detailed description of the classification and list of all used papers and their classification is given in [10]). Although the classification is slightly subjective, it is clear from Table 1. that there is nearly no progress in time towards higher quality of used models. This is rather disappointing, because more and more case studies are available. Low experimental standards make it hard to assess newly proposed techniques; the practical impact of many techniques can be quite different from claims made in publications. This obstructs the progress of the research in the field. Clearly, a good benchmark set is missing.

The need for benchmarking, better experiments, and thorough evaluation of tools and algorithms is well recognized, e.g., experimentation is a key part of Hoare's proposal for a "Grand Challenge of Verified Software" [6]. There is also significant interest in benchmarks in the model checking community (see e.g., Corbett [3], Avrunin et al. [5], Atiya et al. [1], Jones et al. [8]). Nevertheless,

Table 1. Quality of experiments reported in model checking papers. We have used a sample of 80 publications which are concerned with explicit model checking and contain an experimental section (for details see [10]). For each quality category, we report number of published papers in years 1994-2006.

	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006
Q1	-	-	1	1	1	1	1	3	2	4	2	1	1
Q2	-	-	3	3	2	3	3	1	2	2	2	1	-
Q3	-	2	1	3	1	2	2	1	3	2	2	4	1
Q4	1	-	-	-	1	-	1	4	1	1	2	-	2
Q5	-	-	-	1	-	-	-	-	1	-	1	-	-

the progress up to date has been rather slow. The main obstacle in developing model checking benchmarks is the absence of a common modeling language — each model checking tool is tailored towards its own modeling language and even verification results over the same example are often incomparable.

Although the development of benchmarks is difficult and the model checking community will probably never have a universal benchmark set, we should try to build benchmarks as applicable as possible and steadily improve our experimental analysis. This is the goal of this work.

3 BEEM

Modeling Language Models are implemented in a *low-level modeling language* based on communicating extended finite state machines (DVE language, see [10] for syntax and semantics). The adoption of a low-level language makes the manual specification of models hard, but it has several advantages. The language has a simple and straightforward semantics; it is not difficult to write own parser and state generator. Models can be automatically translated into other modeling languages — at the moment, the benchmark set includes also *Promela* models which were automatically generated from DVE sources.

Models and Properties Most of the models are *well-known* examples and case studies. Models span several *different application areas* (e.g., mutual exclusion algorithms, communication protocols, controllers, leader election algorithms, planning and scheduling, puzzles). In order to make the set organized, models are *classified* into different types and categories. The benchmark set is *large* and still growing (at the moment it contains 57 parametrized models with 300 specified instances). *Source codes* of all models are publicly available. Models are briefly described and include *pointers to sources* (e.g., paper describing the case study), i.e., BEEMalso serves as an information portal.

The benchmark set includes also *correctness properties* of models. Safety properties are expressed as reachability of a predicate, liveness properties are expressed in Linear temporal logic. Since an important part of model checking is error detection, the benchmark set includes also *models with errors* (presence of an error is a parameter of a model).

Tool Support The modeling language is supported by an *extensible model checking environment* — The Distributed Verification Environment (DiVinE) [2]. DiVinE is both a model checking tool and a open and extensible library for a development of model checking algorithms. Researchers can use this extensible environment to implement their own algorithms, easily perform experiments over the benchmark set, and directly compare with other algorithms in DiVinE. Promela models can be used for comparison with the well-known model checker Spin [7].

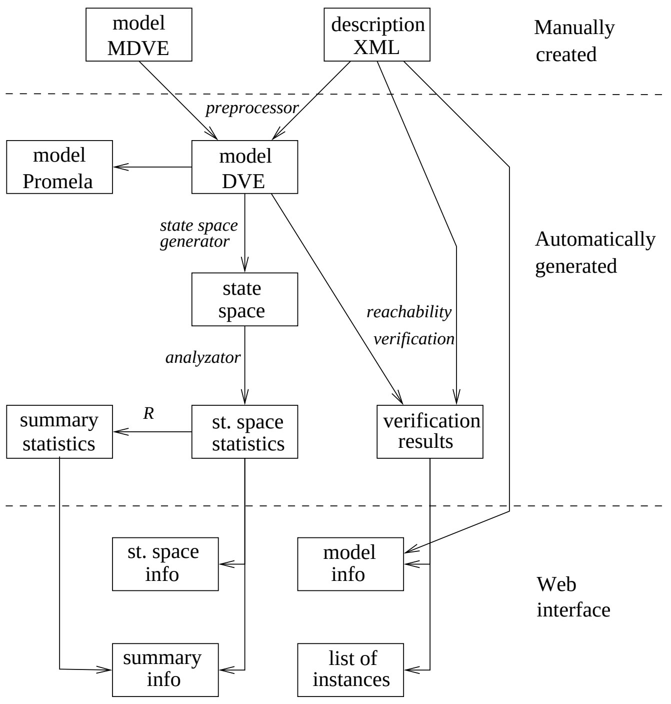

Fig. 1. Overview of the realization of the web portal. The user provides two files: parametrized model and its description. All other information is automatically generated.

Web Portal The benchmark set is accompanied by an comprehensive web portal, accessible at `http://anna.fi.muni.cz/models`, which *facilitates the experimental work*. The web provides (see Fig 1. for overview of realization):

- presentation of all information about models, their parameters, and correctness properties,
- detailed information about properties of state spaces of models [9] including summary information,
- verification results,
- web form for selection of suitable model instances according to a given criteria,
- instance generator, which can generate both DVE models and Promela models for given parameter values.

All data can be downloaded. Since model descriptions are systematic (XML file), it is easy to write own scripts for manipulation with models and automation of experiments.

4 Summary

The aim of this paper is not to present “the ultimate benchmark set” but rather:

– to provide a ready-made set for those who want to compare different model checking techniques and to facilitate experimental research,
– to encourage higher standards in model checking experiments,
– to stimulate the discussion about benchmarks in the model checking community.

Detailed description of the benchmarks set, example of an experimental application, and direction for the future work can be found in the technical report [10].

Acknowledgement I thank Pavel Krčál and to members of the DiVinE group, particularly Ivana Černá, Pavel Šimeček and Jiří Barnat, for collaboration, discussions, and feedback.

References

1. D. A. Atiya, N. Catano, and G. Lüettgen. Towards a benchmark for model checkers of asynchronous concurrent systems. In *Fifth International Workshop on Automated Verification of Critical Systems: AVOCs*, University of Warwick, United Kingdom, Sept. 12–13 2005.
2. J. Barnat, L. Brim, I. Černá, P. Moravec, P. Rockai, and P. Šimeček. Divine - a tool for distributed verification. In *Proc. of Computer Aided Verification (CAV'06)*, volume 4144 of *LNCS*, pages 278–281. Springer, 2006. The tool is available at `http://anna.fi.muni.cz/divine`.
3. J. C. Corbett. Evaluating deadlock detection methods for concurrent software. *IEEE Trans. Softw. Eng.*, 22(3):161–180, 1996.
4. J. Crhová, P. Krčál, J. Strejček, D. Šafránek, and P. Šimeček. Yahoda: the database of verification tools. In *Proc. of TOOLSDAY affiliated to CONCUR 2002*. FI MU report series, 2002. Accessible at `http://anna.fi.muni.cz/yahoda/`.
5. M. B. Dwyer G. S. Avrunin, J. C. Corbett. Benchmarking finite-state verifiers. *International Journal on Software Tools for Technology Transfer (STTT)*, 2(4):317–320, 2000.
6. T. Hoare. The ideal of verified software. In *Proc. of Computer Aided Verification (CAV'06)*, volume 4144 of *LNCS*, pages 5–16. Springer, 2006.
7. G. J. Holzmann. *The Spin Model Checker, Primer and Reference Manual.* Addison-Wesley, Reading, Massachusetts, 2003.
8. M. Jones, E. Mercer, T. Bao, R. Kumar, and P. Lamborn. Benchmarking explicit state parallel model checkers. In *Proc. of Workshop on Parallel and Distributed Model Checking (PDMC'03)*, volume 89 of *ENTCS*. Elsevier, 2003.
9. R. Pelánek. Typical structural properties of state spaces. In *Proc. of SPIN Workshop*, volume 2989 of *LNCS*, pages 5–22. Springer, 2004.
10. R. Pelánek. Web portal for benchmarking explicit model checkers. Technical Report FIMU-RS-2006-03, Masaryk University Brno, 2006.