Summary of Dagstuhl Seminar 06172 on Directed Model Checking

Stefan Edelkamp[1], Stefan Leue[2], Willem Visser[3]

[1] Department of Computer Science, University of Dortmund
D-44227 Dortmund, Germany
stefan.edelkamp@cs.uni-dortmund.de
[2] Department of Computer and Information Science, University of Konstanz
D-78457 Konstanz, Germany
Stefan.Leue@uni-konstanz.de
[3] SEVEN Networks
Redwood City, CA 94063, USA
willem@gmail.com

Abstract. This is a summary of the Dagstuhl Seminar 06172 *Directed Model Checking* that was held 26 - 29 April 2006 at Schloss Dagstuhl, Germany. Directed Model Checking is a software and hardware verification technique that performs a systematic, heuristics guided search of the state space of the model to be analyzed. It hence reconciles classical model checking technology with intelligent, heuristics driven search that has a long tradition in artificial intelligence, in particular in the area of action planning. The benefits are short or even optimally short error trails, in some instances a more efficient exploration of the state space, and the applicability of state space search in some application areas in which unintelligent search would not yield useful results.
The seminar brought together researchers from the system verification and the artificial intelligence domain in order to discuss the current state of the art, and to elicit and discuss research challenges and future directions.

Keywords. Model checking, heuristics, state space search, software and hardware verification.

1 Introduction

Model checking is an increasingly popular verification technology in software and hardware systems analysis. While model checking was originally devised as a complete state space exploration technique aiming at proving models correct, its current primary use is in debugging systems by locating errors in the state space of the model. In particular in explicit-state model checking, which is often used in model checking software systems, debugging is aided by the ease with which offending system traces, also referred to as counterexamples, can be made available. Counterexamples which are system traces leading from the initial state to a property-violating state and their analysis often hints at the causes of errors.

Dagstuhl Seminar Proceedings 06172
Directed Model Checking
http://drops.dagstuhl.de/opus/volltexte/2007/944

The sheer size of the reachable state space of realistic models imposes tremendous challenges on the algorithmics underlying model checking technology. A complete exploration of the state space is often infeasible and approximations are needed. This has lead to a large body of research on optimizing the performance of model checking, including abstraction techniques, state space reduction techniques and improvements in the algorithmics of model checking.

To this end, a recent development was to reconcile model checking with heuristics guided search strategies well investigated in the area of artificial intelligence, in particular in action planning. In standard model checking, the selection of a successor node is performed in a naive fashion without taking knowledge about the problem structure into account. In action planning there is a long tradition of using heuristics guided informed search algorithms, such as A*, in the search for a state in which a planning goal has been reached. The artificial intelligence community has a long and impressive line of research in developing and improving search algorithms over very large state spaces under a broad range of assumptions. It has therefore been observed that there are many similarities between model checking and action planning as well as a large potential for synergies when reconciling these domains, see for instance the discussions at the Dagstuhl seminar 01451 *Exploration of Large State Spaces*. Even though directed model checking approaches have been developed for symbolic and explicit-state model checking, the most natural match seems to be between heuristics guided state space search and explicit state model checking.

Looking back at approximately three quarters of a decade of research on Directed Model Checking the following main benefits of pursuing this approach seem to stand out:

- Uninformed explicit state model checking, which is commonly based on variants of memory efficient depth-first search, generates error trails that are exceedingly lengthy, thus greatly hampering error analysis. Heuristics guided search in Directed Model Checking leads to short, under some conditions even optimally *short counterexamples*, while remaining memory and run-time efficient for most practical examples. Challenges lie in the elicitation of informed and, if possible, admissible heuristics so that the solution quality is good, and the run-time penalty of heuristic search is acceptable.
- For some models Directed Model Checking approaches *permit finding errors* where standard uninformed model checking approaches would fail, either because they exceed memory boundaries or execution time limits. It is an open question what features characterize a state space that would lend itself to Directed Model Checking analysis.
- Directed Model Checking has *extended the applicability* of explicit state space search to some application domains in which classical model checking would not yield useful results. This applies, for instance, to models in which transitions are labeled with quantitative rewards or costs. The analysis of those models requires a judicious choice of the transitions taken along some path to a goal state. Examples include scheduling problems and counterexample generation in stochastic model checking.

The seminar brought together researchers from the system verification and the artificial intelligence domains in order to discuss the current state of the art, and to elicit and discuss research challenges and future directions. The current state of the art was documented by presentations given by the participants, which we briefly summarize in Section 2. The definition of research challenges was the goal of working groups that met during off-hour breakout sessions. The research challenges addressed, and in part the results that were elicited, are summarized in Section 3.

2 Presentations by Participants

The seminar was opened by two tutorials. Stefan Edelkamp introduced into the algorithmic foundations of Directed Model Checking, and Willem Visser focused on applications of Directed Model Checking in the analysis of software code. Particularly interesting was the relationship between abstraction and search heuristics in software model checking. Willem Visser suggested a possible heuristic strategy in abstract model checking, namely to find a counterexample with the least number of nondeterministic choices, since this would reduce the number of predicates necessary to perform abstraction refinement.

The presentation by Matt Dwyer was directed at methodological aspects in the assessment of the cost-effectiveness of Directed Model Checking. The comparison of classical and Directed Model Checking needs to be put on a solid methodological basis in order to lead to meaningful results. A central requirement was to always compare Directed Model Checking to a random search order in classical model checking, since the implemented default order in a classical model checker may be highly biased.

Cyrille Artho presented work on model checking networked application.

Dragan Bosnachki led us back in history to the year 1978 when Jan Hayek presented what is frequently cited as the first formulated approach to systematic statespace exploration. It was interesting to see that Hayek's work already proposed the use of a priority queue to store open states, together with an ad hoc priority function. Hence, this is arguably the first approach towards Directed Model Checking. Dragan concluded in outlining possible applications of Directed Model Checking in test case generation and biological systems analysis.

Husain Aljazzar prosed the use of heuristics guided search in the generation of counterexamples for stochastic model checking. Since those counterexamples consist of sets of traces which must carry a high probability mass only heuristics guided search that takes the probability annotations along transitions into account is able to deliver meaningful results.

Charles Pecheur described the use of heuristics guided search in diagnosis applications for spacecrafts within the Livingstone system of NASA Ames. The heuristics in use is state-based and assesses the number of diagnosis candidates, where a lower count is given priority. During simulation based verification of the diagnosis component the heuristic search strategy could greatly reduce the

number of states that needed to be considered while reducing the per state processing time only minimally.

The optimization of model executions under cost considerations is at the center of the work of Theo Ruys. He presented an extension of the model checker SPIN by a branch and bound algorithm and illustrated its application to a scheduling problem.

Viktor Schuppan talked about the construction of shortest counterexamples for liveness properties in symbolic model checking. The approach is based on verifying liveness as safety and the notion of tightness for Büchi-automata.

The presentation of Alessandro Cimatti reiterated the close relationship between conformant planning and model checking. In particular, he pointed out that a plan has the structure of a finite state machine, i.e., it includes branches and iterations and is not just a linear trace.

Nina Amla presented a hybrid refinement approach that reconciles proof-based and counterexample-based abstraction refinement.

The work presented by Maria Luisa Villani uses heuristics guided search in order to reduce the size of the state space that needs to be traversed in model checking process algebra specifications, in particular Selective Hennessy-Milner logic. The authors propose specific heuristics for deadlock detection as well as general purpose heuristics that apply to arbitrary properties to be verified.

Devising time-optimal schedules using an untimed process algebra (μCRL) is the objective of the work presented by Anton Wijs. On the algorithmics side, this work suggests a particular breadth-first strategy that takes time ticks into account.

The work presented by Ansgar Fehnker proposes a technique for performing counterexamples guided abstraction refinement for hybrid systems. Central to the idea is the selection of suitable fragments of the counterexamples that lead to an efficient refinement.

Tilman Mehler presented a C++ model checker called StEA. The focus of this work lies on efficient state storing techniques as well as on efficient hashing strategies.

The work of Eric Mercer focuses on the efficient computation of distance heuristics in Directed Model Checking. As he pointed out, computing these distances is particularly difficult in the presence of function calls, and he presented an approach to dealing with this problem.

Shahid Jabbar presented an approach that reconciles Directed Model Checking with external model checking, i.e., the storing of states on external storage rather than in main memory.

Sebastian Kupferschmid talked about the adaption of an abstraction-based technique to model checking. The central idea of this work is that in every exploration step an abstraction problem is solved in order to compute a heuristics estimate.

Wolfgang Grieskamp discussed the potential use of search heuristics for test case generation in symbolic execution environments. He illustrated the potential of combining both approaches, but hinted that dealing with portions of the

state vector that are represented symbolically constitutes a major challenge. He also suggested that heuristics should provide "feedback" on the quality of their performance.

3 Working Groups

The organizers of the seminar proposed a number of working groups in order to document the state of the art and to elicit challenges for future research. Unfortunately, due to the tight timing of the seminar the collection of the results of the working groups was rather incomplete and hence the following documentation of these results is to a certain degree incomplete.

3.1 Applications of DMC in Software and Real-Time Systems Analysis, and Other Fields

The working group defined Directed Model Checking as the application of systematic heuristics to influence the state space search order in model checking. It remained a disputed issue whether state space pruning techniques were to be subsumed under Directed Model Checking.

The working group agreed to classify Directed Model Checking applications into two groups. The first group are applications in particular domains of engineering and system design, such as in protocol analysis, the analysis of real-time and stochastic systems, in verifying middleware software and in hardware analysis. It was also predicted that directed model checking would eventually play a rôle in the analysis of biological processes and business workflows. The second class comprises the use of Directed Model Checking as an auxiliary tool in the pursuit of certain application domain goals. This includes program analysis, planning problems, test case generation and eventually also abstraction refinement and error trace analysis.

The participants in the working group elicited a number of research challenges. There was agreement that the evaluation of heuristics based methods is currently lacking due to the absence of suitable benchmark problems. There is also a need to develop metrics that support the classification of models according to properties that influence the performance/cost-effectiveness of certain heuristics. Further challenges were seen in generalizing heuristics, in particular in light of the fact that heuristics can be both domain and problem specific. There also appears to be a need to better understand the use of inadmissible search algorithms, unsound pruning techniques as well as unsound and inadmissible heuristics since these promise advantages in terms of computational costs. Test case generation and error trace analysis were identified as promising application areas, although it is unclear which goals Directed Model Checking is to pursue in these domains. Finally, it was considered important to develop adaptive heuristics that would permit a change in the search strategy as the search is progressing. This coincides with the requirement that heuristics provide "feedback" to the user, an idea that was also voiced by other working groups.

3.2 Algorithmic Foundations of DMC

The group was mainly concerned with the algorithmic foundations of directed model checking. The discussions were driven by the question how to design a most general directed model checking algorithm. The members of the group are currently finalizing a survey article on this topic of which we quote the abstract:

> This article surveys the algorithmic essentials of directed model checking, a promising bug-hunting technique to mitigate the state explosion problem. In the enumeration process, successor selection is prioritized by an evaluation function, called a heuristic. We discuss existing heuristics and methods to automatically generate them by exploiting system abstractions. We extend the algorithm(s) to perform cost-bounded heuristic search and to feature partial-order reduction. For model checking liveness properties we show how these problems can be adapted by lifting the search space. We consider 3 model checking scenarios. For deterministic, finite domains we instantiate the basic model checking algorithms to directed symbolic, external and distributed search. For real-time domains we discuss the adaption of the algorithms and heuristics to timed automata. For probabilistic domains we discuss how how heuristic search can be used to facilitate debugging and diagnostics of large probabilistic models. Last but not least, we show how directed model checking helps to accelerate finding solutions to scheduling problems.

3.3 Synthesis of Heuristics (using Planning and Other Techniques)

No outcome of this working group has been documented.

3.4 Symbolic Techniques for DMC

The working group focused on the use of heuristics during symbolic model checking. Symbolic model checking was taken to include what is commonly referred to as symbolic model checking (using Binary Decision Diagrams), bounded model checking as well as model checking using symbolic execution. Recent work by Qian is a good introduction to the use of heuristics during BDD based symbolic model checking. The group however believed the use of heuristics during SAT based bounded model checking is an interesting research challenge that has yet to be adequately explored. For example, heuristics to guide a SAT solver and methods to prune the state sets during model checking were highlighted as challenges that are not currently being addressed. The use of symbolic execution during model checking is becoming widespread and since it is mostly based on keeping control-flow explicit and data symbolic the group felt that many of the classic heuristic approaches to explicit-state model checking will be applicable also in this area. In addition using heuristics for improving the efficiency of decision procedures as historically been an active area of research that will benefit symbolic execution-based model checking as well.

3.5 Usability of Heuristics used in DMC

(Compiled with the help of Eric Mercer.)

The work group discussion focused on three specific areas: real-time reporting of heuristic performance with user feedback mechanisms, standardized benchmarks for empirical studies, and characterization of heuristics.

The group discussed the question how heuristics could be defined such that the user has some control over the exploration process at run-time. This would involve mechanisms that provide information to the user about how well the heuristics is steering the exploration. The availability of such a feedback loop would allow for a dynamic modification of the heuristics at run-time.

Heuristics for directed model checking critically rely on empirical studies to show performance gains. As such, it is important to have a characterized set of benchmarks that is accepted and understood by the larger community. The benchmarks, whether synthetic or real, should be effective in differentiating performance between heuristics in classes of systems. The benchmarks should also be characterized and understood from a basic set of metrics that convey the complexity of the benchmark and something about the structure of its underlying reachable state space. The benchmarks should be described by general mathematical properties such as branching factor, diameter, etc. rather than a specific language such a Promela or Java, and the benchmarks must contain a characterization of the error as well. The error characterization is as important as the characterization of the benchmark structure since the type of error can affect heuristic performance.

Empirical analysis for directed model checking needs a set of "best practices" to standardize experimental frameworks and and the reporting reporting facilities that they provide. This is essential to characterize heuristic performance. Specifically, as researchers develop heuristics appropriate for use in a guided search algorithm, there is a need to understand the intended problem domain for the heuristic. In other words, we need to characterize heuristics in terms of those problems for which it is expected to be effective. Without this characterization, it is not obvious which heuristic to use for any given property and model in a practical verification setting. A standard benchmark suite is needed for this characterization, but we also need language and metrics to characterize heuristics and their intended problem domains. An interesting thought is to use something similar to the "Patterns" categorization for specifications. Is it possible to characterize, for example, heuristics that are effective for finding violations of repeated chain properties? Or absence properties? Regardless, a heuristic is less usable without a basic understanding of its intended problem domain. Furthermore, it is less usable without a mechanism to automate heuristic selection and tuning for a specific problem domain and property.

4 Conclusion

The seminar succeeded in documenting the state of the art in Directed Model Checking as well as defining challenges for future research. A number of pub-

lications have since emerged from the seminar. Amongst others, a paper by Dwyer, Person and Elbaum that was based a presentation during the seminar was awarded the Best Paper Award at the Foundations of Software Engineering conference held in Portland in the Fall of 2006.

The seminar was held in the fabulous atmosphere that Dagstuhl has offered to the scientific community for more than a decade. We thank the Dagstuhl board and staff for their hospitality. The format of the seminar was that of a more unusual 3 day duration. The participants regretted the relative scarcity of time and agreed that a 5 day duration would have facilitated and enhanced the work of the working groups.

Stefan Edelkamp, Stefan Leue and Willem Visser
Dortmund, Konstanz and San Jose
March 2007