Sindh Univ. Res. Jour. (Sci. Ser.) Vol. 48 (1) 117-120 (2016)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

On Designing Heuristics for Formal Verification Approaches

Q.U.A. NIZAMANI, H. A. NIZAMANI, N. CHANA*, N. J. RAJPER

Institute of Mathematics and Computer Science, University of Sindh, Jamshoro

Received 22nd April 2015 and Revised 26th February 2016

Abstract: Model checking is one of the most widely used verification technique. One of the issues with model checking is the state space explosion which has been studied in greater detail by researchers. Recently, heuristics have been used as a potential remedy for state space explosion problem. This paper, therefore, focusses on identifying the characteristics of heuristics that are important for verification frameworks. To this purpose, we have studied various verification frameworks, identified certain criteria, and have discussed how a particular heuristic characteristic can contribute to efficiency. Thus, designers of verification frameworks have various characteristics of heuristics and their advantages in front of them. Hence, they can design appropriate heuristics for their frameworks yielding maximum efficiency.

Keywords: Verification, Heuristics, Model Checking

1. INTRODUCTION

Software verification ensures that software conforms to its specifications. Many techniques such as testing (Myers, 1979). and formal verification (Clarke, *et al.,* 2000) are used to assess the designed systems. Formal verification approaches such as model checking (Clarke, *et al.,* 2000) theorem proving (Paulson, 1998) static analysis (Bodei, *et al.,* 2001), path analysis (Boreale, *et al.,* 2001), have been rigorously used to confirm the correctness of systems and programs. Though, the application and usage of formal verification tools are numerous, they suffer from various problems too. Paramount to all these problems is the issue of state space explosion i.e., tools fail to verify the given model within available physical resources (such as memory) and usually crash before yielding any output. More precisely, the state space generated for the given model is too large to be effectively examined by the searching algorithms. Many researchers have suggested different techniques to abate the state space explosion problem such as symbolic model checking (Amadio, *et al.,* 2003), partial order reduction (Clarke, *et al.,* 2000), abstractions (Clarke, *et al.,* 1994), and directed model checking to name a few.

Directed Model Checking (*DMC*) (Edelkamp, *et al.,* 2004). is a promising technique, which aims at integrating heuristic techniques into model checking algorithms thereby directing the tool to explore the specific portions of the state space containing errors. It has been shown that use of heuristics can significantly alleviate the problem of state space explosion (Paulson, 1998). In fact, many software systems that could not be verified earlier with conventional techniques can now be analyzed using heuristic inspired verification frameworks. Keeping in view the potential of DMC, it is vital to know the various design patterns for heuristics and how they influence the searching algorithms. In this paper, various verification frameworks are analyzed to isolate the design of heuristics from them. This allows us to identify the characteristics of heuristics that contribute most towards achieving efficiency. The study will be helpful for the developers to understand the various heuristic design methods and to choose the one suitable to their verification tool. In particular, we have chosen (*i)* Lazy infinite state analysis of security protocols (Basin, 1999) (*ii)* Directed explicit state model checking of communication protocols (Edelkamp, *et al.,* 2004) (*iii)* Symbolic model checking (Nizamani, and Tuosto, 2009) to be studied in this study with focusing on the characteristics of the heuristics used in them.

2. BACKGROUND

This section provides the necessary ingredients to understand the work presented in this paper.

2.1 Lazy Infinite State Analysis

David Basin in his seminal work (Basin, *1999*). represented protocols as infinite trees. Thus verification becomes the task of checking the property at every node of the tree. This allows the searching technique to look for security protocol breaches by searching through the infinite trees. The tree is generated in the lazy way and heuristics are used for re-ordering the order of exploring nodes in the tree.

The approach has been illustrated via Needham-Schroeder protocol whereby it has been modeled as infinite set of traces and computation of such traces is

++Corresponding author: QN@usindh.edu.pk

* University of Sindh, Jamshoro.

done using Haskell. Exponential growth of the infinite tree with large branching factor makes the conventional search infeasible. Therefore, an alternative strategy, i.e., intelligent searching is employed for efficient manipulation of the state space. The first heuristic is about those traces that are generated as a result of invalid actions i.e., actions generated by the intruder but are not valid for protocol. Such traces are pruned. The second heuristic is about re-ordering where priority is given to those traces where intruder is involved and also to the events which are as a result of first step of protocol followed by second and third step of the protocol.

2.2 Explicit State Model Checking approach

In (Edelkamp, *et al.,* 2004). authors have used explicit state model checking for the analysis of the communication protocols. Explicit state model checking is where every possible state that can be generated while developing the solution is enumerated and then the state space is analyzed for possible faults. Since every state is generated and no reduction techniques are used, therefore state space explosion is the major problem in such analysis. Edelkamp *et al*., employed the heuristic flavored algorithms for searching and coined the term *DMC* for model checking approached underpinned by intelligent search.

One particular contribution of this approach is shorter error trails which help the verifier to easily analyze and track the problem containing areas of the model. The authors have used spin model checker to demonstrate their approach and modified algorithm is called HSF-spin.

2.3 Symbolic Model Checking

The symbolic Model checker (Nizamani and Tuosto, 2009) is specially formulated for security protocol analysis and is not Binary Decision Diagrams (BDD) based. The model checker partly explores the state space heuristically and partly with conventional Depth First Search (*DFS*) Algorithm. The heuristic approach is successful in reducing the state space and directing the search towards attack containing portions of the state space.

The approach allows a number of protocols to be easily verified when they failed to be checked by the conventional search. Heuristic is based on the idea of the formal property to be verified. This allows the framework to be rather general and applicable to other security protocol verification approaches also. The heuristics allows pruning as well as re-ordering of the nodes however since the heuristic function is evaluation based, therefore it not necessarily results into shorter error trails.

3. HEURISTICS

A heuristic is a problem solving strategy practically tested methods that may not guarantee to be optimal or best, but usually generates solutions that are acceptable. Heuristics evaluate the states and assign them with a number (called weights) that corresponds to the chance of that state leading to a goal/target state. Finally the search has to explore the states according to increasing/decreasing weights of the states.

3.1 Criteria for Designing Heuristics

This section elaborates the criteria that will be used throughout the paper for evaluating the heuristics.

3.1.1 Evaluation/Estimation

A heuristic function can be evaluation function or estimation function. In former case, the nodes are weighted merely based on their chance of leading to goal states. While in the latter case, the nodes estimate the number of steps that will be required to reach the goal state.

3.1.2 Admissibility

Admissibility is defined as that the estimated cost of reaching the goal node is always less or equal to the actual cost. An admissible heuristic when used with A*search algorithm yields an optimal solution.

3.1.3 Simple/Composite

A simple heuristic evaluates or estimates every node in the graph using same metric. In contrast to that a composite heuristic function can combine two or more than two heuristics so that the heuristic function has the freedom to apply the most appropriate heuristic on the node in question according to its type.

3.1.4 Re-ordering/Pruning

Usually the heuristics are intended for the purpose of re-arranging the order of exploration of nodes. This will be refereed in this paper as re-ordering. However, in some cases heuristics can additionally cut-off a certain portion of the state space such that enclosed nodes are never explored. This feature of heuristics is referred to as pruning**.**

3.1.5 Error Trails

Model checking is used mainly to locate errors in the design of software. However, the error trace generated by the model checking tools is usually too lengthy and requires time and efforts to analyze it. If heuristics can guide the search towards error states whose error trails are shorter, it allows the verifier to easily comprehend and rectify the error. Thus shorter error trails is a characteristics in which verifiers can be interested.

3.1.6 Searching Algorithm

Though heuristics are designed in number of different ways and their design contributes to the actual efficiency that you achieve. It is equally important to consider the pros and cons of the searching algorithm that is going to utilize the heuristic. The algorithm must suit the heuristic and should exploit the heuristic in best possible manner. Therefore, a number of intelligent search algorithms are available that can be used to this purpose such as A*, IDS, etc.

4. RESULTS AND DISCUSSIONS

The analysis of heuristics in various verification frameworks is presented in **(Table 1).** In the first column of the **(Table 1),** we enlist the criteria that are used as a means to measure the efficiency of the heuristic. The second column is used for specifying each of three verification frameworks that we have considered in this paper. As seen in (**Table 1).** Lazy infinite state analysis and explicit state model checking are using the heuristics that are simple in nature. However, symbolic model checking is using both kinds of heuristics viz., simple as well composite.

A heuristic can contribute to efficiency by re-ordering as well as via pruning. As evident from **(Table 1),** infinite state and symbolic model checking both are using heuristics that are capable of re-ordering as well as pruning. Explicit state model checking is on the other hand using a heuristic that is only performing re-ordering. As a general rule, we consider pruning to be contributing to efficiency when whole state space has to be explored for the purpose of verification.

Also, Lazy infinite state analysis and symbolic model checking both are using evaluation-based functions. Contrary to that, explicit state model checking is employing estimation-based function. Therefore, it has also been mentioned in the table that heuristic function used by explicit state model checking is also admissible.

Finally due to the use of estimation function, explicit state model checking generates shorter error trails. However, symbolic model checking and Lazy infinite state analysis are lacking on this feature. Shorter error trails are easier to comprehend and verifier can easily locate the bug in the model.

5. CONCLUSION

Since directed model checking is proving to be a successful verification technique, this work aims at providing various characteristics that should be kept in mind while designing heuristics. The study of various frameworks shows that pruning can additionally help in reducing state spaces if the intention of verification is to remove all bugs. It is also evident that if heuristic functions that are estimation based can be used for verification of those systems where shorter error trails are required. However, it must be kept in mind that with the estimation function, heuristics must be admissible in order to contribute to efficiency.

Table-1. Analysis of heuristics in various verification frameworks.

Characteristics of Heuristics		Verification Approaches		
		Lazy Infinite state analysis approach	Explicit State Model Checking	Symbolic Model Checking
Design of Heuristics	Simple	✓	✓	✓
	Composite	×	×	✓
Efficiency	Re-ordering	✓	✓	✓
	Pruning	✓	✓	✓
Heuristic Function	Estimation-based	×	✓ Admissible	×
	Evaluation-based	✓	×	✓
Shorter Error Trails		✓	✓	×
Search Algorithm		IDS	A*	Hybrid(Blind+Intelligent)

REFERENCES:

Amadio, R., D. Lugiez, V. Vanackere (2003). On the symbolic reduction of processes with cryptographic functions. Theoretical Computer Science, vol. 290 (1): 695–740.

Basin, D. (1999) Lazy Infinite-State Analysis of Security Protocols. In Proceedings of the International Exhibition and Congress on Secure Networking-CQRE (Secure), 30–42. Springer-Verlag, London, UK,

Bodei, C., P. Degano, F. Nielson, and H. R.Nielson (2001). Static analysis for secrecy and non-interference in networks of processes. Lecture Notes in Computer Science, 21(27): 27–34.

Boreale, M., (2001). Symbolic trace analysis of cryptographic protocols In Lecture Notes in Computer Science,Springer-Verlagvol,2076:667-681.

Clarke, E. C., S. Jha, W. R. Marrero (2000). Partial Order Reductions for Security Protocol Verification. In Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems, (TACAS '00), 503–518. Springer-Verlag.

Clarke, E. M., O. Grumberg, and D. E. Long (1994). Model Checking and Abstraction. ACM Transactions on Programming Languages and Systems, 16: 1512-1542.

Clarke, E. M., O. Grumberg, and D. A. Peled. (2000). Model Checking. MIT Press, Cambridge, MA, USA.

Edelkamp, S., S. Leue, and A. Lluch-Lafuente (2004). Directed Explicit state Model Checking in the Validation of Communication Protocols. International Journal of Software Tools and Technologies Transfer, 5(2): 247–267.

Myers, G. J. (1979). Art of software Testing, John Wiley and Sons, Inc. New York, NY, USA. Transactions on Programming Languages and Systems, 6:1512–1542

Nizamani, Q., and E. Tuosto (2009). Heuristic Methods for Security Protocols. Electronic Proceedings in Theoretical Computer Science, 7: 334-3348.

Paulson, L (1998). The Inductive Approach to Verifying Cryptographic Protocols. Journal of Computer Security, 6: 85–128.