Master Thesis Proposal

Consistency-based Software Fault Diagnosis with Multiple Observations

Lukas Graussam, BSc

Advisor: Univ.Prof. Dipl.-Ing. D.Phil. Georg Weissenbacher

April 24, 2023

CONTENTS

1 MOTIVATION & PROBLEM STATEMENT

Manually finding bug sources in programs with ever growing complexity is a tedious and expensive task for developers. With the rapid advancements of automated verification and testing in software, also the discipline of fault-localization has become of great importance in recent years. Given a bug, the goal is to automatically derive a subset of locations in the program which are candidates for being altered in order to prevent the bug. There is a broad spectrum of techniques tackling this task, though many popular approaches depend on model-based diagnosis (MBD) [13].

State-of-the-art implementations of model-based software fault-localization [9] [3] [5] [1] [11] [12] have strong limitations in terms of their fault model, with candidate locations mostly being restricted to a simple subset of the programming language ANSI-C. A single observation, i.e., a single configuration of inputs or a trace leading to a bug, is processed by the majority of currently available solutions. However, in order to make software fault-localization applicable for real-world scenarios, current solutions mainly lack in fulfilling the following demands.

- A fault model which also supports complex constructs of a programming language
- The result set, i.e., reported fault-locations, needs to be minimal in order to truly aid the debugging process

The latest advances [14] [15] [6] [10] [8] [7] [11] [12] in the discipline of model-based diagnosis include algorithms which process multiple observations efficiently. Their improvements are usually shown on simple hardware circuits with injected errors. However, recent MBD algorithms have not been implemented for software fault-localization yet. With continuous-integration tools and the already commonly large amount of tests in software verification, there are usually many observations at hand in practical environments. Thus, MDB with multiple observations is expected to be well suited for software fault-localization, and might potentially deliver improved results, i.e., a more precise set of locations and improved performance compared to the state-of-the-art.

2 AIM OF THE WORK

The aim of this work is to deploy the recent MBD algorithm *HSD* [8] in order to perform software fault-localization on realistic code examples and comparable benchmark programs of the language ANSI-C. Simple fault-models of existing solutions shall be lifted to work with more complex programming constructs, for example pointers. The outcome shall be evaluated and compared against relevant implementations. Therefore the main contributions of this thesis shall be

- A well defined fault model for the ANSI-C language
- A tool for modeling actual C programs accordingly, such that state-of-the-art MBD algorithms can be applied and multiple observations processed.
- Evaluation results of the *HSD* algorithm and the derived fault model on realistic software programs.

The following main research questions shall be addressed.

- How can complex programming constructs be modeled for fault-localization and how does it affect results?
- To what degree can state-of-the-art MBD algorithms, especially *HSD*, with multiple observations and an enhanced fault-model improve results and limitations of existing approaches in software fault-localization?

3 METHODOLOGICAL APPROACH

1. **Literature Review**
   The literature review shall focus on model-based software fault-localization and general model-based diagnosis. Furthermore, testing, verification and software-model-checking background might be beneficial for design decisions.

2. **Design & Implementation**
   A viable design for the following components shall be determined. Thereafter, a functioning prototype shall be implemented.

   - A fault model for different ANSI-C features needs to be derived, i.e., what does it mean if a certain programming-construct is declared faulty?
   - A tool for parsing and transforming an input program into a model which can be used by MBD algorithms is required. Components (possibly faulty) in the model need to be mapped back to locations in the input program. The well-known model-checking tool *CBMC* [2] shall be modified to realize this task.
   - Observations, i.e. input configurations, need to be parsed and processed.
   - Deployment of the actual MBD algorithm with the derived models must be achieved.

3. **Evaluation**
   The prototype shall be applied on different realistic and comparable programs with injected faults. Statistics of quality and performance need to be created for evaluating the prototype and comparing it to other approaches.

4 STATE-OF-THE-ART

The art of software fault-localization has become a popular field of research in recent years. While there is a broad spectrum of techniques [13], the focus in this thesis lies on model-based software fault-localization.

Alex Groce [5] introduces Error Explanation using a distance metrics between program executions on a model created by *CBMC* [2]. It requires a successful execution which is close to a failed execution and reports components that differ in the executions.

Griesmayer et al. [3] [4] again use *CBMC* to create a model and compute fault locations based on a given counter-example to a specification. They iteratively run the model checker on different instrumented versions of the program where one or multiple components are altered in order to derive possible fault locations. The original algorithm is running the model checker for all possible locations, which is very inefficient. Thus, the approach has been refined several times, most recently by Birch et al. [1] claiming improved performance and operating on single assignments as possible fault locations.

Also in [9] the tool *CBMC* is used to create a model of the input program. A maximum-satisfiability (MAX-SAT) solver is then deployed in their formula-based algorithm in order to derive a set of faulty components, working only with single observations.

The most recent MBD algorithm with multiple observations (DC) which has an implementation in software fault-localization is presented in [11] [12]. A weaknesses of this approach is that the algorithm can compute a potentially large number of redundant diagnoses, according to [8]. In [8] a novel algorithm (HSD) is proposes which does compute minimal diagnosis and outperforms previous algorithms in experimental results on fault-injected hardware circuits. Zhout et al. [6] [15] [15] propose further improvements which are also evaluated on fault-injected hardware circuits.

5 RELEVANCE TO THE COMPUTER ENGINEERING CURRICULA

This thesis relies on theoretical and practical background in formal verification. Many courses in the curricula of Computer Engineering match this field. The following list summarizes the most related ones to the topics of this thesis.

- 104.271 Discrete Mathematics VO
- 104.272 Discrete Mathematics UE
- 185.291 Formal Methods in Computer Science VU
- 185.A93 Formal Methods in Computer Science UE
- 181.144 Computer Aided Verification VU
- 181.145 Computer Aided Verification UE
- 184.774 Automated Deduction VU
- 192.106 Software Model Checking

REFERENCES

[1] Birch, G., Fischer, B., and Poppleton, M. (2019). Fast test suite-driven model-based fault localisation with application to pinpointing defects in student programs. *Softw. Syst. Model.*, 18(1):445–471.

[2] Clarke, E., Kroening, D., and Lerda, F. (2004). A tool for checking ansi-c programs. In *Tools and Algorithms for the Construction and Analysis of Systems: 10th International Conference, TACAS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29-April 2, 2004. Proceedings 10*, pages 168–176. Springer.

[3] Griesmayer, A., Staber, S., and Bloem, R. (2007). Automated fault localization for c programs. *Electronic Notes in Theoretical Computer Science*, 174(4):95–111. Proceedings of the Workshop on Verification and Debugging (V&D 2006).

[4] Griesmayer, A., Staber, S., and Bloem, R. (2010). Fault localization using a model checker. *Software Testing, Verification and Reliability*, 20(2):149–173.

[5] Groce, A. (2004). Error explanation with distance metrics. In Jensen, K. and Podelski, A., editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 108–122, Berlin, Heidelberg. Springer Berlin Heidelberg.

[6] Huisi, Z., Ouyang, D., Zhang, L., and Tian, N. (2022). Model-based diagnosis with improved implicit hitting set dualization. *Applied Intelligence*, 52.

[7] Ignatiev, A., Morgado, A., and Marques-Silva, J. (2017). Model based diagnosis of multiple observations with implicit hitting sets.

[8] Ignatiev, A., Morgado, A., Weissenbacher, G., and Marques-Silva, J. (2019). Model-based diagnosis with multiple observations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 1108–1115. International Joint Conferences on Artificial Intelligence Organization.

[9] Jose, M. and Majumdar, R. (2010). Cause clue clauses: Error localization using maximum satisfiability. *CoRR*, abs/1011.1589.

[10] Kalech, M., Stern, R., and Lazebnik, E. (2021). Minimal cardinality diagnosis in problems with multiple observations. *Diagnostics*, 11:780.

[11] Lamraoui, S.-M. and Nakajima, S. (2014). A formula-based approach for automatic fault localization of imperative programs. In Merz, S. and Pang, J., editors, *Formal Methods and Software Engineering*, pages 251–266, Cham. Springer International Publishing.

[12] Lamraoui, S.-M. and Nakajima, S. (2016). A formula-based approach for automatic fault localization of multi-fault programs. *Journal of Information Processing*, 24:88–98.

[13] Wong, W., Gao, R., Li, Y., Abreu, R., and Wotawa, F. (2016). A survey on software fault localization. *IEEE Transactions on Software Engineering*, 42:1–1.

[14] Zhou, H., Ouyang, D., Tian, X., and Zhang, L. (2023). Diagdo: an efficient model based diagnosis approach with multiple observations. *Frontiers of Computer Science*, 17(6):176407.

[15] Zhou, H., Ouyang, D., Zhao, X., and Zhang, L. (2022). Two compacted models for efficient model-based diagnosis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4):3885–3893.