Design Model Repair with Formal Verification

Cheng-Hao Cai(✉), Jing Sun, and Gillian Dobbie

School of Computer Science, University of Auckland, Auckland, New Zealand
{chenghao.cai,jing.sun,g.dobbie}@auckland.ac.nz

Abstract. The main research content of this topic is model repair in formal methods. Formal verification can verify the correctness of a model using rigorous mathematical methods. However, the repair of incorrect models is usually done by humans. In order to automate the model repair, we combine the B method, formal verification, probabilistic methods, satisfiability modulo theories and program synthesis, and we study various automatic model repair algorithms, which are used to fix reachability and eliminate invariant violations and deadlock states in incorrect models.

Keywords: Model repair · B method · Model checking · Refinement

1 Introduction

This work targets to an automatic model repair problem based on formal verification. Given a model described by a logical language and a set of properties that are needed to be satisfied, formal verification tools can verify whether the model satisfies these properties. If any properties are not satisfied, then the model may be incorrect. The question is: can a computer automatically fix the model?

The B method [2] is a correct-by-construction software development technique. Its core idea is to start with a highly abstract model, gradually refine the model and finally convert the refined model to complete software. During the design and refinement process, the correctness of the model is verified using formal logics several times, so that the final software is highly reliable. At present, there are efficient B model checkers such as ProB [8] and Rodin [3]. Although model checking is automated, subsequent model repair processes still require the involvement of humans. Humans need to analyse the results of the model checking, find out the errors in the model, propose possible repair solutions and manually repair the model, but this process is often inefficient. In order to improve the efficiency of model design, we have proposed a number of automatic model repair algorithms.

This work is supported by the State Scholarship Fund sponsored by the China Scholarship Council [Grant Number: 201708060334].

Y. Ait-Ameur and S. Qin (Eds.): ICFEM 2019, LNCS 11852, pp. 489–492, 2019.
https://doi.org/10.1007/978-3-030-32409-4_30

Currently, we have developed algorithms that can automatically eliminate deadlock states, invariant violations and assertion violations in B models [6]. The algorithms use model checking techniques to calculate the finite state space of a model, find error states in the state space, calculate candidate repairs with satisfiability modulo theories (SMT) and use probabilistic methods to select repairs. Moreover, we have developed an algorithm that can be used for reachability repair. It complements missing parts of a model using probabilistic methods, so that the model can reach a set of previously unreachable states. Further, we have confirmed the effectiveness of the algorithms via experiments.

2 Related Work

B model repair is currently an emerging research direction. It has been proposed in [11] and further improved in [12], where inductive programming is used to generate repairs for given I/O examples. For example, to generate a new operation, a number of instances of pre- and post-states must be given. Then a precondition that covers the pre-states and a post-condition (i.e., substitution) that transitions the pre-states to the post-states are synthesised using inductive programming, and the two conditions constitute a new operation. Additionally, a model repair approach based on refinement checking has been proposed in [4], which replaces model components that violate refinement conditions with other components that satisfy the conditions.

Recently, a number of techniques for automatic software repair have been developed, including those of imperative programming languages [7]. These techniques mainly include two parts: fault localisation and repair generation. At present, one of the most commonly used fault localisation methods is spectrum-based fault localisation [1]. Its central idea is to obtain execution paths of a program using test suites and estimate possible locations of errors by observing overlapping parts of these paths. Methods for repair synthesis include template-based repair, mutation repair, genetic programming, etc [7]. Similar to B model repair, the above automatic software repair techniques aim to improve the efficiency of finding and eliminating bugs in software development processes. According to [9], one of the key problems of automatic software (or model) repair is that the number of candidate fixes is generally huge, which results in a combinatorial explosion. To solve this problem, repair algorithms usually include evaluation functions for filtering high-quality fixes from candidate fixes.

3 Proposed Solutions

In order to achieve automatic model repair, we have proposed a semantic learning algorithm for constructing the evaluation function of filtering high-quality repairs. The core idea of semantic learning is to obtain the design intent of a model from the state space of the model using classification techniques. The model's state space is a collection of valid state transitions. Using a binary classifier, the state space can be probabilistically modelled to produce a semantic

model that predicts whether any state transition is valid. The semantic model is used to calculate scores of repair. For details on how to vectorise state spaces, train classifiers and score repairs, please refer to our GitHub repository [1].

Additionally, we proposed three general-purpose repair operators: insertion, modification and deletion. Insertion is a reachability repair operator. Given a model M and a desired state s, if s is unreachable, then inserting an additional state transition into the state space of M can make s reachable. In this process, the semantic model is used to rank candidate insertions. Modification is used to fix existing state transitions that violate given properties. Given a property P that a model M needs to satisfy, if any state transitions produced by M do not satisfy P, then a SMT solver is used to search for candidate edits to make these transitions satisfy P, and the semantic model is used to score and rank the edits. In order to apply modifications, scores of the edits need to achieve a certain level. If not, then deletion is used to remove the faulty state transitions.

The significance of the above methods is that they provide a general-purpose model repair strategy, and probabilistic machine learning techniques, especially classification algorithms, can assist in the model design process. As semantic learning and the three repair operators are based on the model's state space, they can be used not only for the B method, but also for other formal design methods based on the checking of state space. The classification algorithms allow the intent of model design to be modelled as evaluation functions, leading to more efficient repairs. At present, there are many studies related to classification algorithms, and these algorithms can be directly used for semantic learning. If we try more classification algorithms in the future, the predictive performance of the semantic model may be further improved.

4 Current Results and Future Work

We are currently developing a tool named B-repair that implements the B model repair algorithms described in Sect. 3. B-repair uses scikit-learn [10] and Silas [5] as semantic learners to support binary classifiers such as logistic regression models, support vector machines, random forests and artificial neural networks. Moreover, B-repair uses ProB [8] as a model checker and a SMT solver. Currently, B-repair can automatically eliminate invariant violations, assertion violations and deadlock states in B models using modifications and deletions, and it can use insertions to achieve simple reachability repair. Additionally, it supports batch repairs, which can fix multiple errors in a model at the same epoch. In order to improve B-repair, we are developing more complex repair functions, optimising the classifiers and extending the model repair algorithms to refinement.

We tested B-repair using a collection of representative models in the ProB Public Examples Repository. The results revealed that the semantic learning method led to 98.3% of average prediction accuracy. Moreover, we seeded faults into the models and individually used the three repair operators to repair the models. Results revealed that the deletion operator was able to eliminate all

[1] Our repository is on https://github.com/cchrewrite/B-Model-Repair.

invariant violations in the models, and average repair accuracies of insertion and modification reached 86.7% and 89.8%, respectively.

In the future, our work will include the following aspects. First, we will collect model quality criteria from past studies and use them to evaluate results of model repair. Second, we will make a benchmark dataset of B model repair and perform a comprehensive performance test on B-repair. Finally, we will combine model repair with refinement checking to achieve a complete software development process.

References

1. Abreu, R., Zoeteweij, P., Golsteijn, R., van Gemund, A.J.C.: A practical evaluation of spectrum-based fault localization. J. Syst. Softw. **82**(11), 1780–1792 (2009)
2. Abrial, J.: The B-Book - Assigning Programs to Meanings. Cambridge University Press, Cambridge (2005)
3. Abrial, J., Butler, M.J., Hallerstede, S., Hoang, T.S., Mehta, F., Voisin, L.: Rodin: an open toolset for modelling and reasoning in Event-B. Int. J. Softw. Tools Technol. Transf. **12**(6), 447–466 (2010)
4. Babin, G., Ameur, Y.A., Singh, N.K., Pantel, M.: A system substitution mechanism for hybrid systems in Event-B. In: Proceedings of Formal Methods and Software Engineering - 18th International Conference on Formal Engineering Methods, ICFEM 2016, Tokyo, Japan, 14–18 November 2016, pp. 106–121 (2016)
5. Bride, H., Dong, J., Dong, J.S., Hóu, Z.: Towards dependable and explainable machine learning using automated reasoning. In: Proceedings of Formal Methods and Software Engineering - 20th International Conference on Formal Engineering Methods, ICFEM 2018, Gold Coast, QLD, Australia, 12–16 November 2018, pp. 412–416 (2018)
6. Cai, C., Sun, J., Dobbie, G.: B-repair: repairing B-models using machine learning. In: 23rd International Conference on Engineering of Complex Computer Systems, ICECCS 2018, Melbourne, Australia, 12–14 December 2018, pp. 31–40 (2018)
7. Gazzola, L., Micucci, D., Mariani, L.: Automatic software repair: a survey. IEEE Trans. Softw. Eng. **45**(1), 34–67 (2019)
8. Leuschel, M., Butler, M.J.: ProB: an automated analysis toolset for the B method. STTT **10**(2), 185–203 (2008)
9. Mechtaev, S., Gao, X., Tan, S.H., Roychoudhury, A.: Test-equivalence analysis for automatic patch generation. ACM Trans. Softw. Eng. Methodol. **27**(4), 15:1–15:37 (2018)
10. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
11. Schmidt, J., Krings, S., Leuschel, M.: Interactive model repair by synthesis. In: Proceedings of Abstract State Machines, Alloy, B, TLA, VDM, and Z - 5th International Conference, ABZ 2016, Linz, Austria, 23–27 May 2016, pp. 303–307 (2016)
12. Schmidt, J., Krings, S., Leuschel, M.: Repair and generation of formal models using synthesis. In: Integrated Formal Methods - 14th International Conference, IFM 2018, Maynooth, Ireland, 5–7 September 2018, pp. 346–366 (2018)