Mutation-based test-case generation for Event-B*

A new plug-in for Rodin

Rupert Schlick and Thorsten Tarrach

AIT Austrian Institute of Technology

Motivation Event-B is a modelling language, which is used for safety-critical systems. The system is refined in several steps, until the last refinement is close to the actual implementation. A well-known problem is to confirm that the final implementation indeed conforms to the model. We propose to use automated test design for this purpose. The method cannot prove the absence of bugs, but in practice it can discover a wide range of problems. It offers a balanced compromise solution: it is computationally less expensive and scales better than full verification, it delivers tests that demonstrate that the full specification is implemented and it can also be used against black-box implementation, e.g. in hardware or on third party systems without access to the source code.

Testing is useful, even if the implementation is automatically generated. Unless the generation is fully proven [3], we also need to test it. If it is proven, we still need to test the deployed system in its full target environment, which can also show unexpected emergent behaviour.

In this abstract we introduce the Rodin plug-in we developed that generates tests for Event-B models.

MoMuT At AIT we develop a tool for mutation-based test-case generation (MBTCG) called MoMuT [1]. MBTCG accepts a behaviour model as input and produces test-cases as output. The model specifies the input and output behaviour of a system and the test cases are sequences of inputs and expected outputs. The tests can be run against an implementation to increase confidence that the implementation conforms to the model. Testing entails no guarantee that

*This work has been conducted within the ENABLE-S3 project that has received funding from the ECSEL joint undertaking under grant agreement no 692455. This joint undertaking receives support from the European Union's Horizon 2020 Research and Innovation Programme and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway. ENABLE-S3 is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" via FFG project number 853308 between May 2016 and April 2019.

the implementation indeed conforms to the model. Therefore, we need to select test-cases that are likely to discover incorrect implementations. MBTCG generates test cases by first mutating the input model and then generating tests that can distinguish the mutated model from the original model. Mutation here means that we change a small aspect of the model, e.g. replacing a mathematical operator. Further details regarding MBTCG can be found in [4, 6].

For this abstract MoMuT is treated merely as a black box that accepts models and produces test cases. The models are provided in our proprietary input language OOAS (Object-Oriented Action Systems), an object oriented extension of action systems [5]. MoMuT can accept UML models as input by first translating the UML model to OOAS. Our Rodin plug-in works by translating Event-B models to OOAS and thereby enables MoMuT to generate test cases for Event-B models.

Translating Event-B to OOAS We lack the space to go into detail about the translation of Event-B to OOAS, so we will report on its challenges. Both languages use the principle of Dijkstra's guarded commands [2]. Thereby, the structural transformation is in principle easy because every event in Event-B corresponds to one action in OOAS and the contexts provide us with sufficient information to derive types of variables and values of enumerations. The main problem is that sets as a construct are not (yet) supported in OOAS. We emulate them using unordered arrays and every set operation is a loop over the entire array. While this allows us to implement all set operations they are not as efficient as corresponding data structures in other languages. A confounding issue is that we need to know the maximal size of a set at translation time. This is problematic since Event-B uses the cross product as a type to represent both maps and relations, which have very different maximal sizes, and we prefer smaller arrays for performance reasons. We use a heuristical approach to find out how a set is used.

We do not yet support all Event-B operators, but implement them on the go as our benchmarks require them. Implementing a new operator takes about an hour.

Evaluation Since there could be problems both in the translation to OOAS and the test case generator itself, we cross-check the generated tests against the Event-B model. To do so, we use a test execution engine built using the Event-B capable model checker ProB. Use cases from Thales Austria and Airbus Germany, developed as part of the ENABLE-S3 Project, are used to evaluate and demonstrate the approach.

Work in progress Apart from completeness of the translation we are working on a better integration of the Rodin plugin into the GUI. Currently we generate an OOAS text file and the user may then invoke MoMuT on the command line with that file as input. We aim to automatically invoke MoMuT from Rodin and transform the tests into ProB traces by running the test with ProB. These we would show in the Rodin GUI directly.

References

[1] B. Aichernig, H. Brandl, E. Jöbstl, W. Krenn, R. Schlick, and S. Tiran. MoMuT::UML model-based mutation testing for UML. In *Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on*, pages 1–8, April 2015. https://www.momut.org/.

[2] Edsger W Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18(8):453–457, 1975.

[3] Andreas Fürst, Thai Son Hoang, David Basin, Krishnaji Desai, Naoto Sato, and Kunihiko Miyazaki. Code Generation for Event-B. In *Integrated Formal Methods*, Lecture Notes in Computer Science, pages 323–338. Springer, Cham, September 2014.

[4] Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *Software Engineering, IEEE Transactions on*, 37(5):649–678, 2011.

[5] Willibald Krenn, Rupert Schlick, and Bernhard K. Aichernig. Mapping UML to labeled transition systems for test-case generation: A translation via object-oriented action systems. In *Proceedings of the 8th International Conference on Formal Methods for Components and Objects*, FMCO, pages 186–207, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] Mike Papadakis and Nicos Malevris. Mutation based test case generation via a path selection strategy. *Information and Software Technology*, 54(9):915–932, 2012.