ParTeG - Integrating Model-Based Testing and Model Transformations

Dehla Sokenou
GEBIT Solutions
dehla.sokenou@gebit.de

Stephan Weißleider
Fraunhofer-Institut FIRST
stephan.weissleder@first.fraunhofer.de

coverage criterion is converted into a set of model-specific test goals: For instance, All-States is converted into test goals, each of which is referencing one state of the state machine. ParTeG generates paths from the state machine’s initial state to the states that are referenced by the test goals. All conditions on each path are transformed and used for concrete input parameter selection corresponding to the selected boundary-based coverage criterion. With these input parameters, the state machine paths are converted into executable test cases of one provided target language. ParTeG also supports test goal monitoring so that already covered test goals are excluded from further test case generations.

Mutation analysis is a wide-spread approach to measure the fault detection capability of a test suite. This approach is based on fault injection in a correct implementation. ParTeG supports mutation analysis by using a mutation factory provided by the tester that delivers a new mutant for each test execution and by generating JUnit code which can be used by the mutation tool Jumble [UTC^+07].


2 Model Transformations

Model transformations are means to convert a model into any other model of any other modeling language. ParTeG supports several model transformations to improve the fault detection capability of the generated test suite. It changes the structure of a state machine while preserving its semantics. Several transformations can be found in a corresponding report on an industrial cooperation [Wei09]. We present just one example: State machines can contain choice pseudostates that reference more than one incoming and more than one outgoing transition (see Figure 1(a)). The outgoing transitions also contain guards. Now we compare the effects of existing coverage criteria: Transition-based coverage criteria are focussed on transition sequences. They can enforce that all transition sequences from the states $F$ or $G$ to $H$ or $I$ are traversed. They fail, however, if the guard condition is more complicated and we are interested in condition values. Control-flow-based coverage criteria are focussed on guard conditions, but they do not necessarily cover all transition paths. As a consequence, the guard values $[X]$ and $[\text{else}]$ may be tested, e.g. just including the state $G$ – all faults that are related to state $F$ may be undiscovered. An intuitive solution seems to consist of generating two test suites for both kinds of coverage criteria. As a result, transition sequences and guards are tested. However, the guards are not properly tested for each sequence. Instead, we propose to transform the state machine, e.g. by splitting choice pseudostates according to their number of incoming transitions. Figure 1(b) shows the transformed test model for this example. Since there is only one incoming transition for each choice pseudostate, any selected control-flow-based coverage criterion now has to be satisfied for both states $F$ and $G$. In our industrial cooperation, all proposed model transformations resulted in a significant increase of the generated test suite’s fault detection capability. All proposed transformations are implemented in ParTeG. They can be automatically executed for all used state machines.

![Figure 1: Model transformation to split choice pseudostates.](image-url)

References

