Error Localization Based on the Counterexamples Generated by Model-Checker

Chen Shi (0910753)

School of Information Science,
Japan Advanced Institute of Science and Technology

May 19, 2012

Keywords: error localization, counterexample, model-checker, problem characteristics.

Today, software has been deeply involved in human life in many aspects. At the same time, software tends to be complicated with large-scale. The safety and reliability of software has become more and more important. Model checking, as an approach that can exhaustively verify the correctness of the specification, has attracted considerable attention lately. In model checking, we check a property against a model that represents a target system to be developed. Counterexamples are generated by model-checker when the model does not satisfy given properties. By analyzing the counterexamples, we can find out the errors of model and fix the defects of specification finally.

However, redundant and unreadable counterexamples often output from a model with complex behavior. Sometimes it takes a lot of time to identify an error of a model when we analyze these counterexamples by hand. To make error localization easier for model developers, analyzing the counterexamples and localizing the errors automatically are expected.

In recent years, counterexample explanation for model checking has been researched, but instead of localizing the errors location exactly, any of the existing research just stopped to provide the processed information that helps people to analyze counterexample. You need to refer to the processed information and find out the errors of the model by yourself eventually.

In model checking, a variety of counterexamples are generated when various kind of problems are contained in the model. Each kind of the problems has characteristics respectively. If we do not take those characteristics of the problems into account, it is difficult to find the errors out exactly.

In this dissertation, we proposed a method to find the errors of a specified model automatically based on the counterexamples generated by model-checker, using problem characteristics as additional information. As a result, we succeeded in proposing such method as well as the one, which provides information about how to correct the model.

There are various kinds of problems that make the counterexamples generated during model checking. In this research, we pick up a problem named "race conditions" as our analysis target, which is the problem that often occurs in a concurrent system. We use safety property described in assertion as desired properties, and use SPIN as our model-checker, which is a well-known model-checker awarded ACM System Software Award.

Race conditions problem has the following characteristics.

- Race conditions problem occurs when the access to shared resources of one process is interrupted by another.
- Race conditions problem can be fixed by executing the access operations of shared resources consecutively.

The idea of the proposed method is that we focus on the characteristics of the race conditions, which make the identification of the problem and the correction of the errors easier. In the proposed method, the problem of "error localization" is come to be the problem of "finding out the illegal interrupt locations in model", and the problem of "presentation of correction method" is come to be the problem of "determining the process scope that should be executed atomically".

Counterexamples contain the execution sequences that cause race conditions. We can determine which process was interrupted by other processes and where was the interrupt point according to the execution sequence. We can find out the answer of how to avoid the occurrence of race conditions based on the execution sequence where there is no illegal interruption exists in witnesses, that is, the correct execution sequence which does not cause race conditions.

To solve the problem of "finding out the illegal interrupt locations in model", we first extract all interrupt points from execution sequence of counterexamples. Then, identify the illegal ones by comparing the counterexamples with the witnesses. Those interrupt points that only exist in counterexamples, never appear in witness are determined as illegal ones.

To solve the problem of "determining the process scope that should be executed atomically", we select the minimum scope of continuous processes that beginning with the "illegal interrupt point" from the execution sequence of witnesses. The correction method provided as analysis result suggests the scope that should to be executed atomically.

According to the proposed method, we implemented an analysis tool, which generates all counterexamples and witnesses of a specified model first, then find out all of the errors that induce race conditions automatically, as the analysis result, the correction method of each error is provided.

In order to evaluate the proposed method, we prepared the models in various structures with race conditions problems embedded and the models implemented with the typical algorithms, which are usually mentioned when discussing race conditions problem. We carry out the evaluation experiments on these prepared models using the above analysis tool to see whether the tool can find the embedded errors and provide the correction method correctly.

The result of the experiments shows that the proposed method has the following advantages.

- The proposed method is effective. All of the embedded errors of race conditions are found automatically by the analysis tool in the experiments. It was confirmed that after modified the model according to the correction method presented by the analysis tool, counterexamples were no longer output by the model-checker.
- The correction method provided by the analysis tool is appropriate. The provided scope is just the process scope, which should be executed atomically to avoid race conditions.
- For all counterexamples are analyzed, multiple errors can be discovered at one-time.
- The proposed method is a practical method, which accepts the existing model as input, and executes the entire analysis automatically.
- By applying the proposed method, the time of error localization has been shortened in seconds, which improves the efficiency of counterexample analysis significantly.

But for the proposed method output all of the counterexamples and witnesses of a specified model, if the counterexamples or witnesses are generated from the model with a huge number, you should either tune the search depth of the verification or reduce the number of states by refining the model to shorten the analysis time.

In the future, like race conditions picked up in this research, various kinds of other problems can be handled in the same way by considering the problem characteristics of each one.