Constraints over Intervals for Specification Based Automatic Software Test Generation

Martine Ceberio[1], Angel F. Garcia Contreras[1], Clothilde Jeangoudoux[2], and Fabrice Larribe[2]

[1] University of Texas, El Paso
500 W. University, El Paso, Texas 79968, USA
[2] Safran Electronics & Defense
21 av. du Gros Chêne, 95610 Erangy sur Oise, France.
mceberio@utep.edu, afgarciacontreras@miners.utep.edu,
clothilde.jeangoudoux@lip6.fr, fabrice.larribe@safrangroup.com

Keywords: interval arithmetic, constraint programming, automatic test generation, mutation testing, software certification, software specification

Developing critical software and ensuring its compliance with lawful requirements are difficult, expensive and resource-intensive activities. In the aeronautical industry, it is required to provide some quality guarantees in terms of robustness and functional safety. Documented guidance [1] to produce certifiable software describes the software life cycle processes and verification and validation activities. Among those guidelines, we would like to draw the attention on the need for a description of the functional behavior of the software. A test set for the validation and verification of the software is designed from this functional specification.

To develop a test campaign, the test designer must ask him- or herself two questions:

- Where to test? That is, which point in the software will be more likely to be badly implemented, i.e., which *test case* can allow us to detect an incorrect behavior.
- How to reach that test case? That is, which *configuration* of the system under test will allow us to perform the verification of this test case.

In order to implement specification based automatic test generation, we first formalize the functional behavior of the software by means of constraints programming [2] over interval variables. Then, with the help of interval constraint solving techniques [3], we describe a method to automatically achieve the two steps of specification based test design.

First, we use a mutation testing approach [4] over interval constraints to evaluate the quality of the test set and generate new test cases.

Second, once a new test case is selected, we identify the configuration of the input variables allowing to reach this test case by constraint resolution over the interval variables.

We show that our method provides necessary test cases for software certification. Moreover, representing variable constraints as intervals gives guarantees on the generated results: all the generated test cases are proven to be necessary and all the generated input configurations are intervals of values that enable the system to reach with certainty the related test case. Our method has been developed for industrial specification of aeronautical software.

References

[1] RTCA Special Committee 205 (SC-205) and EUROCAE Working Group 71 (WG-71): DO–178C, ED–12C, Software Considerations in Airborne Systems and Equipment Certification, *RTCA, Inc*, (2012).

[2] R. Barták: Constraint Programming: In Pursuit of the Holy Grail, *In Proceedings of the Week of Doctoral Students (WDS99 -invited lecture*, (1999), 555–564.

[3] L. Granvilliers, and F. Benhamou: RealPaver: An Interval Solver using Constraint Satisfaction Techniques, *ACM Trans. on Mathematical Software 32(1)*, (2006), 138–156.

[4] R. A. DeMillo and A. J. Offutt: Constraint-Based Automatic Test Data Generation, *IEEE Transactions on Software Engineering*, (1991), 900–910.