Evolutionary Testing Supported by Slicing and Transformation

Mark Harman
Lin Hu
Rob Hierons
Brunel University
Uxbridge
Middlesex
UB8 3PH, UK

Chris Fox
University of Essex
Wivenhoe Park
Colchester
CO4 3SQ, UK

Sebastian Danicic
Goldsmiths College
University of London
New Cross, London
SE14 6NW, UK

Joachim Wegener
Harmen Sthamer
André Baresel
DaimlerChrysler AG
Research and Technology
Alt-Moabit 96a
D-10559 Berlin
Germany

Abstract *Evolutionary testing is a search based approach to the automated generation of systematic test data, in which the search is guided by the test data adequacy criterion.*

Two problems for evolutionary testing are the large size of the search space and structural impediments in the implementation of the program which inhibit the formulation of a suitable fitness function to guide the search.

In this paper we claim that slicing can be used to narrow the search space and transformation can be applied to the problem of structural impediments. The talk will present examples of how these two techniques have been successfully employed to make evolutionary testing both more efficient and more effective.

Evolutionary Testing

Evolutionary Testing [6] uses metaheuristic search based techniques to find good quality test data. Test data quality is defined by a test adequacy criterion. The fitness function drives the search by rewarding candidate solutions which perform better according to the criterion.

Slicing to Reduce Search Space Size

Search algorithms are particularly sensitive to the size of the search space. The size of the search space is exponential in the number of input variables to the program, so methods which reduce this size may produce exponential speed ups in the search.

Clearly not all predicates in all programs depend upon all inputs. Therefore, it makes sense to determine the input variables which may influence a given predicate before attempting to generate test data to cover it. This is essentially a question answered by dependence analyses such a slicing [2]. We use an approach based upon the algorithm of Danicic and Harman [1] which produces variable dependence information as a by product of slicing. We have built a tool which allows the test data generator to produce variable dependence information for predicates of interest to reduce search space size.

Transformation to Overcome Structural Problems

Generating test data using evolutionary test data generation has been shown to be successful, but its effectiveness is significantly reduced in the presence of programming features such as flags, unstructured control flow and side effects.

In our work we seek transformations which simply make the programs easier to test. The transformations are therefore different to conventional transformations because they are only a means to an end not an end in themselves. Also, the transformations need not preserve traditional notions of equivalence, marking a radical departure from conventional approaches to transformation.

We have experimented with flag removal [3] and plan to experiment with side effect removal [4] and restructuring transformations [5]. In the talk we present the initial results of our flag removal work which show that testability transformation has the potential to greatly improve test data generation ability.

References

[1] S. Danicic, M. Harman, and Y. Sivagurunathan. A parallel algorithm for static program slicing. *Information Processing Letters*, 56(6):307–313, Dec. 1995.

[2] M. Harman and R. M. Hierons. An overview of program slicing. *Software Focus*, 2(3):85–92, 2001.

[3] M. Harman, L. Hu, R. Hierons, A. Baresel, and H. Sthamer. Improving evolutionary testing by flag removal ('best at GECCO' award). In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1359–1366, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[4] M. Harman, L. Hu, X. Zhang, and M. Munro. Side-effect removal transformation. In 9^{th} *IEEE International Workshop on Program Comprehension (IWPC'01)*, pages 310–319, Toronto, Canada, May 2001. IEEE Computer Society Press, Los Alamitos, California, USA.

[5] L. Ramshaw. Eliminating goto's while preserving program structure. *Journal of the ACM*, 35(4):893–920, 1988.

[6] J. Wegener, A. Baresel, and H. Sthamer. Evolutionary test environment for automatic structural testing. *Information and Software Technology Special Issue on Software Engineering using Metaheuristic Innovative Algorithms*, 43(14):841–854, 2001.