

Automatic inter-procedural Test case generation

Karim-Cyril Griche

Laboratoire LSR-IMAG

BP72, 38402 Saint Martin d'Hères Cedex, FRANCE

{karim-cyril.griche@imag.fr}

Abstract

Our work is based on a new approach of the automatic structural test case generation problem defined in [GBR98]. It uses constraint logic programming (CLP) to try and solve the problem of generating test cases in order to attain the structural covering of a procedure. A test tool prototype, named Inka[GBR98], has been developed by *Thales Systèmes Aéroportés*. Inka is designed for automatic structural test case generation for C programs. The operating cycle of Inka is cut in three parts:

- The first part translate a C program into it's equivalent in constraint logic program.
- The second module of Inka is used to minimize the set of constraints representing the C program. It uses the prolog solvers to propagate constraints until it reaches a fix point.
- Then the last module use the restraint set of constraints in order to find a test case reaching a node selected in the control flow graph of the procedure.

Tests realized by Inka are generated to achieve structural covering of a procedure. Usually, when encountering functions calls during unitary test, a tester either replace the function by a predefined stub or choose to unfold the call and introduce the called function's code into the test.

For now on, Inka make the choice to systematically unfold the function call, which means replacing the call by the set of constraints representing the called function. While this method has many advantages and is quite simple to achieve, but it has a major drawback. The complete analyze of the whole unfold code is much too expensive in execution time.

Our work in the Inka project is to find a way to treat large programs. Our approach of this problem is to find an alternative between stubs and complete unfolding. The idea is to produce a simplified version of a called function intended to "work well" when structurally test the calling function. In order to produce simpler functions, we are following two axes of investigations. The first one is theoretic: we look for known techniques, which could match our idea of function simplification. The second axis is empirical: we define our needs and determine simplifying heuristics to be tested within the Inka prototype. This paper is based upon these axes and presents our work so far with the heuristics.

GBR98 A. Gotlieb, B. Botella, and M. Rueher. Automatic Test Data Generation Using Constraint Solving Techniques. *Software Engineering Notes*, 23(2):53-62, Mar. 1998. disponible at <http://www.esi.fr/~rueher>

A. Gotlieb, B. Botella, and M. Rueher. Automatic Test Data Generation Using Constraint Solving Techniques. *Software Engineering Notes*, 23(2):53-62, Mar. 1998. disponible at <http://www.esi.fr/~rueher>