08351 Abstracts Collection
Evolutionary Test Generation
— Dagstuhl Seminar —

Holger Schlingloff[1], Tanja E.J. Vos[2] and Joachim Wegener[3]

[1] Fraunhofer Institut - Berlin, Germany
hs@informatik.hu-berlin.de
[2] Universidad Politècnica de Valencia, Spain
tvos@dsic.upv.es
[3] Berner & Mattner Systemtechnik - Berlin, Germany
joachim.wegener@berner-mattner.com

Abstract. From September 24th to September 29th 2008 the Dagstuhl Seminar 08351 "Evolutionary Test Generation " was held in Schloss Dagstuhl – Leibniz Center for Informatics. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Software-testing, evolutionary algoritms, meta-heuristic search

08351 Summary – Evolutionary Test Generation

From September 24th to September 29th 2008 the Dagstuhl Seminar 08351 "Evolutionary Test Generation " was held in Schloss Dagstuhl – Leibniz Center for Informatics. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. This paper contains an executive summary of the seminar and the open problems that were found.

Keywords: Software-testing, evolutionary algoritms, meta-heuristic search

Joint work of: Schlingloff, Holger; Vos, Tanja; Wegener, Joachim

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2009/2022

SAT-based Automatic Test Pattern Generation

Due to the rapidly growing size of integrated circuits, there is a need for new algorithms for Automatic Test Pattern Generation (ATPG). While classical algorithms reach their limit, there have been recent advances in algorithms to

solve Boolean Satisfiability (SAT). Because Boolean SAT solvers are working on Conjunctive Normal Forms (CNF), the problem has to be transformed. During transformation, relevant information about the problem might get lost and therefore is not available in the solving process.

In the following we briefly motivate the problem and provide the latest developments in the field. The technique was implemented and experimental results are presented. The approach was combined with the ATPG framework of NXP Semiconductors. Significant improvements in overall performance and robustness are demonstrated.

Keywords: Circuit, ATPG, SAT, Boolean Satisfiability

Joint work of: Drechsler, Rolf; Eggersglüß, Stephan; Fey, Görschwin; Tille, Daniel

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2009/2015

See also: 1. Drechsler, R., Eggersglüß, S., Fey, G., Glowatz, A., Hapke, F., Schlöffel, J., Tille, D.: On acceleration of SAT-based ATPG for industrial designs. IEEE Transactions on Computer Aided Design of Circuits and Systems 27 (2008) 1329-1333 2. Drechsler, R., Eggersglüß, S., Fey, G., Tille, D.: Test Pattern Generation using Boolean Proof Engines. Springer (2009) to appear.

Ranking schemes for test cases in worst-case coevolution

Jürgen Branke (Universität Karlsruhe, DE)

In this paper, we consider the case of worst-case optimization, i.e., the user is interested in a solution's performance in the worst case only. If the number of possible test cases is large, it is an optimization problem by itself to determine a solution's worst case performance. In this paper, we apply coevolutionary algorithms to co-evolve the worst case test cases along with the solution candidates. We propose a number of new variants of coevolutionary algorithms, and show that these techniques outperform previously proposed coevolutionary worst-case optimizers on some simple test problems.

Keywords: Coevolution, worst-case optimization, evolutionary algorithm

Full Paper:
http://dx.doi.org/10.1007/978-3-540-87700-4_15

Stressing and Stretching

John Clark (University of York, GB)

In this brief presentation I talk about evolving input distributions and also ways of distorting programs to make hard to reach paths/states easier.

The key difference to previous work is that rather than evolve test data, we evolve strategies for generating test data. In one approach, inspired by sheet music, we can evolve a "tune" (comprising notes that are server requets) to stress the system. Another approach seeks to learn an input data distribution to maximise the coverage probability for the least covered element (based on Thevenod Fosse's statistical testing criterion). Much random testing assumes independently generated variables. Our approach does not.

I also describe briefly the concept of "program stretching", where we alter the branch predicates to make them easier to satisfy, find satisfying test data and then collapse the distorted program back to the original, attempting to "drag" the test data with us as we go. Distribution work is with Simon Poulding. Stretching work is with Kamran Ghani and Mark Harman.

Keywords: Testability Transformation, Program Stretching, Evolutionary Testing

Search-based Approaches to Interaction Testing

Myra Cohen (University of Nebraska, US)

In software systems, specific combinations of input parameters, configuration options or event sequences often trigger unexpected outcomes or faults. Testing to uncover these types of faults is called interaction testing. A common sampling technique for generating samples of inputs or configurations for interaction testing is to find a set of input or configuration combinations that tests all pairs or t-way combinations of the inputs or configurations.

Finding a minimal set is an optimization problem, which has been solved over the years through the use of greedy algorithms, hill climbing, tabu search, genetic algorithms, linear programming and simulated annealing. In this talk I will provide an overview of some of the search-based approaches that have been used for generating interaction test samples, and will discuss some limitations of the current approaches, opening some new and interesting research directions for the search-based testing community.

SAT-based Automatic Test Pattern Generation

Rolf Drechsler (Universität Bremen, DE)

The postproduction test of integrated circuits is crucial to ensure a high quality of the final product. This test is carried out by checking the correct response of the chip under predefined input stimuli - or test patterns.

These patterns are calculated by algorithms for Automatic Test Pattern Generation (ATPG). The basic concepts and algorithms for ATPG are briefly reviewed. Then, an advanced SAT-based ATPG tool is introduced and emprically evaluated.

Keywords: Test, ATPG, SAT

Mutation Testing

Yue Jia (King's College - London, GB)

Traditional mutation testing considers only first order mutants (FOM), created by the injection of a single fault. Often these FOMs denote trivial faults that are easily killed. This talk presents the idea of higher order mutation testing. In higher order mutation testing, we apply both FOMs and subsuming higher order mutants (HOMs) together. A subsuming HOM is one that harder to kill than the first order mutants from which it is constructed. By definition, subsuming HOMs denote subtle fault combinations. This talk also proposed a new approach that applies co-evolutionary idea to generate test case that can kill these subtle fault.

Keywords: Mutation testing, higher order mutant, search-based software engineering

Runtime of Search Heuristics on Software Testing Problems

Per Kristian Lehre (University of Birmingham, GB)

Evolutionary test generation methods are usually evaluated experimentally. To deepen the understanding of the working principles behind these methods, one could also consider a mathematical analysis.

This talk we will describe recent theoretical results on analysing the expected runtime and success probabilities of search heuristics on the problems of generating unique input output sequences for finite state machines, and test data for branch coverage testing of software.

Keywords: Finite state machines, evolutionary algorithms, runtime analysis, unique input output sequences

Using evolutionary algorithms to select parameters from equivalence classes

Felix Lindlar (TU Berlin, DE)

This paper presents some ideas about an approach which aims at extending existing methodologies for functional testing. Experience in automotive applications has shown that when selecting parameters for functional testing, many times a tester has equivalence classes in mind. Instead of losing valuable information in the process, support should be given to make them manageable. The proposed approach suggests evolutionary testing strategies to search for critical representatives within equivalence classes.

Keywords: Equivalence classes, evolutionary testing, functional testing, automotive industry

Joint work of: Lindlar, Felix; Marrero Peréz, Abel

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2009/2012

Co-testability Transformation Oracle

Philip McMinn (Sheffield University, GB)

This talk introduces the use of a testability transformation as a partial oracle in search-based testing.

The fitness function aims to guide the search to the discovery of differences in behaviour between the original program and a transformed counterpart.

The transformed program works alongside the original program rather than replacing it, and is thus called a 'co-testability transformation oracle'.

Co-testability transformation oracles are presented for finding bugs in code with floating point calculations.

Keywords: Search-based testing, testability transformation, partial oracle

Co-testability Transformation

Philip McMinn (Sheffield University, GB)

This paper introduces the notion of Śco-testability transformationŠ. As opposed to traditional testability trans- formations, which replace the original program in testing, co-testability transformations are designed to be used in conjunction with the original program (and any additional co-transformations as well). Until now, testability transfor- mations have only been used to improve test data genera- tion. However, co-testability transformations can function as partial oracles. This paper demonstrates practical usage of a co-testability transformation for automatically detect- ing floating-point errors in program code.

Keywords: Search-based testing, testability transformation

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2009/2013

Full Paper:
FastAbstract-11549.pdf

See also: Phil McMinn. Co-Testability Transformation. Testing: Academic & Industrial Conference - Practice And Research Techniques (TAIC PART 2008) Fast Abstract, 2008.

Issues in Model-Based Testing

Ina Schieferdecker (TU Berlin, DE)

Model-Based Testing (MBT) is a variant of testing that relies on explicit behaviour models that encode the intended behaviour of a system and/or the behaviour of its environment. MBT aims at automating test case generation. It is often combined with automated test execution.

This presentation gives an overview on concepts and principles of MBT and argues about its prospects and costs. MBT is then contrasted with recent approaches for test specification and automated test execution, which reveals gaps in the power of current MBT methods.

Open issues in MBT research are identified and selected ones being discussed: namely, finding model-based measures for test coverage and test quality, MBT for non-functional requirements, MBT for timed or continuous systems, and the generation of practical tests. For these issues, ongoing work together with relations to evolutionary test (ET) generation are presented.

This reveals ideas of e.g. guiding the ET search by the test oracle, to enhance the ET search with test pattern knowledge, to find additional test data that well spread over the SUT or to optimize traffic sets for performance tests with ET.

Keywords: Model-Based Testing, Test Automation, Evolutionary Test Generation

Structural Software Testing with Active Learning in a Graph

Michèle Sebag (LRI CNRS, FR)

Motivated by Structural Statistical Software Testing (SSST), this paper is interested in sampling the feasible execution paths in the control flow graph of the program being tested. For some complex programs, the fraction of feasible paths becomes tiny, ranging in $[10^{-10}, 10^{-5}]$. When relying on the uniform sampling of the program paths, SSST is thus hindered by the non-Markovian nature of the "feasible path" concept, due to the long-range dependencies between the program nodes.

A divide and generate approach relying on an extended Parikh Map representation is proposed to address this limitation; experimental validation on real-world and artificial problems demonstrates gains of orders of magnitude compared to the state of the art.

Keywords: Structural Statistical Software Testing, Active Learning, Control Flow Graph, Feaisble Paths, Parikh maps

Joint work of: Baskiotis, Nicolas; Sebag, Michèle; Gaudel, Marie-Claude

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2009/2014

Fuzzy Logic Based Objective Function Construction for Evolutionary Test Generation

Andrea Tettamanzi (University of Milano, IT)

The test case generation problem can be stated as an optimization problem whereby the closeness of test cases to violating the postcondition of a formal specification is maximized, subject to satisfying its precondition.

This is usually implemented by constructing an objective function which provides a real-valued estimate of how distant all of the constraints are from being violated, and then trying to minimize it.

A problem with this approach is that such objective functions may contain plateaux, which make their minimization hard. We propose a similar approach, grounded on fuzzy logic, which uses, instead of a "distance from violation" objective function, a fuzzy degree of proximity to postcondition violation and produces plateaux-free objective functions by construction.

The approach is illustrated with the help of a case study on the functional (black-box) testing of computer programs.

Keywords: Functional testing, fuzzy logic, objective function

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2009/2016

Pex - White-Box Testing for .NET

Nikolai Tillmann (Microsoft Research - Redmond, US)

Pex is a white-box test generation tool for .NET. Starting from a hand-written parameterized unit test, Pex analyzes the program-under-test to determine relevant test inputs fully automatically. To this end, Pex executes the program multiple times with different inputs while monitoring the taken execution paths. For each path, the precise path condition is constructed by monitoring. Pex uses several search strategies to select the next path to explore, with the goal of achieving high statement coverage fast. Pex uses an SMT solver to determine if the next chosen path is feasible, and if so, to determine test inputs that will exercise the path. Pex is integrated into Microsoft Visual Studio.

Keywords: Whitebox testing dynamic symbolic execution concolic smt solver .net C# instrumentation monitoring

Full Paper:
http://research.microsoft.com/Pex

Instrumenting where it hurts: an automatic concurrent debugging technique

Shmuel Ur (IBM - Haifa, IL)

As concurrent and distributive applications are becoming more common and debugging such applications is very difficult, practical tools for automatic debugging of concurrent applications are in demand. In previous work, we applied automatic debugging to noise-based testing of concurrent programs. The idea of noise-based testing is to increase the probability of observing the bugs by adding, using instrumentation, timing "noise" to the execution of the program. The technique of finding a small subset of points that causes the bug to manifest can be used as an automatic debugging technique. Previously, we showed that Delta Debugging can be used to pinpoint the bug location on some small programs.

In the work reported in this paper, we create and evaluate two algorithms for automatically pinpointing program locations that are in the vicinity of the bugs on a number of industrial programs. We discovered that the Delta Debugging algorithms do not scale due to the non-monotonic nature of the concurrent debugging problem. Instead we decided to try a machine learning feature selection algorithm. The idea is to consider each instrumentation point as a feature, execute the program many times with different instrumentations, and correlate the features (instrumentation points) with the executions in which the bug was revealed. This idea works very well when the bug is very hard to reveal using instrumentation, correlating to the case when a very specific timing window is needed to reveal the bug. However, in the more common case, when the bugs are easy to find using instrumentation points ranked high by the feature selection algorithm is not high enough. We show that for these cases, the important value is not the absolute value of the evaluation of the feature but the derivative of that value along the program execution path.

As a number of groups expressed interest in this research, we built an open infrastructure for automatic debugging algorithms for concurrent applications, based on noise injection based concurrent testing using instrumentation. The infrastructure is described in this paper.

Keywords: Concurrent testing, automatic debugging

Full Paper:
http://portal.acm.org/citation.cfm?id=1273469&dl=&coll=

Ideas on Signal Generation for Evolutionary Testing of Continuous Systems

Andreas Windisch (TU Berlin, DE)

Test case generation constitutes a critical activity in software testing that is cost-intensive, time-consuming and error-prone when done manually. Hence, an automation of this process is required. One automation approach is search-based testing for which the task of generating test data is transformed into an optimization problem which is solved using metaheuristic search techniques. However, only little work has so far been done to apply search-based testing techniques to systems that depend on continuous input signals rather than single discrete input values.

This paper proposes three novel approaches to generating input signals from within search-based testing techniques for continuous systems.

Keywords: Search-Based Testing, Optimization, Metaheuristic

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2009/2011

Combination of EAs and other techniques

Qingfu Zhang (University of Essex, GB)

In this talk, I present four combinations of EAs and other techniques:

1. Orthogonal GA: GA+experimental design
2. Guided mutation: GA+EDA
3. RM-MEDA: multiobjective EDA based on a math property
4. MOEA/D: a multiobjective population algorithm framework based on aggregation

Keywords: Estimation of distribution algorithm, multiobjective optimization, experimental design, genetic algorithm