Ideas on Signal Generation for Evolutionary Testing of Continuous Systems

Andreas Windisch[1]

Technische Universität Berlin
Daimler Center for Automotive IT Innovations
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
andreas.windisch@dcaiti.com

Abstract. 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

1 Introduction

The critical activity of testing is the systematic selection of test data, since the accomplishment of an exhaustive test is typically infeasible in practice. Since the test data generation process is very cost-intensive, time-consuming and error-prone when done manually, the automation of this process is highly aspired. While various search-based automation approaches for certain testing objectives on the code level have been developed and proven of value in recent years (e.g. [1], [2], [3]), there have been only few efforts addressing the search-based test data generation for systems depending on input signals, i.e. sequences of data values.

In order to apply the idea of search-based testing to the generation of input sequences for continuous systems, an efficient way of generating and optimizing signal sequences is needed. This way both white- and black-box testing techniques would be applicable for continuous systems.

2 Signal Generation Approaches

The creation of effective test scenarios for dynamic systems in general, e.g. the representation of system input signals by complete value arrays is hard to realize for real-world systems as these often require the signals to have a specific minimum length (number of included time steps). In combination with a small sample rate the resulting size of arrays representing the signals prevent its application to optimization engines. Accordingly, a differentiation between optimization and simulation sequences needs to be accomplished. The optimization sequence should consist of only a small number of parameters that are used as inputs for the optimization engine and can be used to be transformed into a simulation sequence by interpolation. In contrast, the simulation sequence is used as input for the execution of the system under test.

The following subchapters propose three approaches that are based on this principle.

2.1 Fourier Series

This approach generates signals by calculating its data values using trigonometric polynomials of variable degrees k. This is shown in equation 1 which represents the simulation sequence $T_k(t)$ and which is used to calculate the signal values based on parameters a_n and b_n which in turn constitute the optimization sequence.

$$T_k(t) = \frac{a_0}{2} + \sum_{n=1}^{k} \left(a_n \cdot \cos(n \cdot t) + b_n \cdot \sin(n \cdot t)\right) \tag{1}$$

The user must specify the desired degree k which leads to a certain number of parameters: $2 \cdot k + 1$. In order to optimize this set of parameters any advanced metaheuristic search technique can be used, such as for example Genetic Algorithms.

2.2 Segment Approach

The underlying idea of how to generate signals generally is based on the work of Baresel et al. [4]. It proposes a simple way of generating signals by stringing together a certain number of parameterized base signals. Therefore it makes use of further information about the input signals for the system under test that needs to be provided by the tester. This on the one hand includes general attributes, such as for example the length of the signals to be generated and their designated resolution. On the other hand it also contains separate specifications of the signals for each input of the system under test. It is thus possible to determine a list of base signals using which one particular input signal should be build up and to specify for example the signal amplitude boundaries to be kept. Figure 1 illustrates the general assembly of parameterized base signals (signal segments) into one composite signal. These signal segments can be specified using the three parameters *Amplitude*, *Transition Type* and *Width*.

One signal accordingly can be described using the parameters of the signal segments it is composed of, i.e. $3 \cdot n$ parameters with n being the specified number of signal segments. This number can be reduced if the user is able to provide knowledge about the signals to be generated. If for example only one

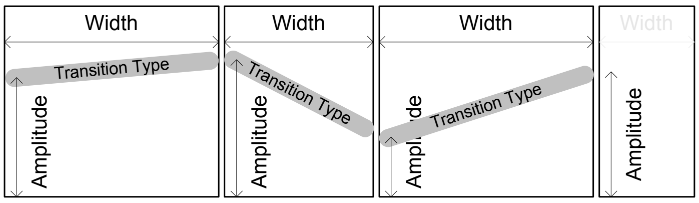

Fig. 1. General signal assembly based on a certain number of parameterized base signals. Each base signal depends on three parameters: amplitude, transition and width.

transition type is allowed for one signal, the search engine does not need to optimize this parameter. As a result the optimization sequence would reduce to $3 \cdot n$ parameters. The same applies to the width and the amplitude. However, this would further reduce the length of the optimization sequence to n parameters, if this kind of knowledge is given.

In contrast, the simulation sequence, i.e. the test data, is obviously given by the signals that are generated from the optimization sequences.

2.3 Linear Genetic Programming

This approach extends the previously introduced segment approach by applying principles of linear genetic programming (e.g. [5], [6], [7], [8]) to automatically vary the single signal segments in order to overcome the drawback of having to specify the number of signal segments.

Figure 2 illustrates the data structure the linear genetic programming algorithm operates on directly. A *population* consists of a previously specified number

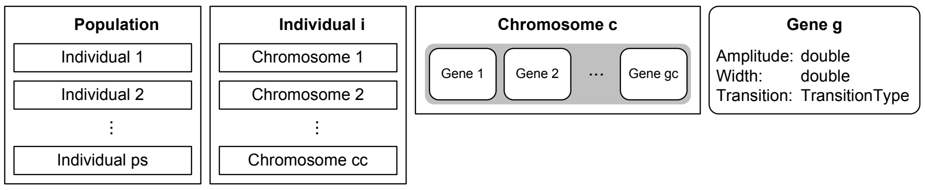

Fig. 2. Encoding of the data structures used by linear genetic programming for signal generation. A population consists of *ps* individuals, each individual consists of *cc* chromosomes, which in turn consist of *gc* genes.

of individuals which directly influences both effectiveness and efficiency of the metaheuristic search. Each *individual* is build up on a certain number of chromosomes, i.e. the particular representations of the inputs of the system under test. The number of chromosomes within each individual must be specified by the user, since this highly depends on the actual system under test. A *chromosome* contains a certain number of genes. Each chromosome is the representation of valid signals for the associated input of the system under test. Hence, input wide attributes, such as amplitude boundaries or the allowed transition types are set for the entire chromosome and thus apply to all genes included. The chromosomes represent the linear data structures the linear genetic programming operators operate on. *Genes* are representations of signal segments and accordingly feature three parameters: Amplitude, Transition Type and Width.

This data structure is evolved according to the principles of linear genetic programming using the genetic operators: selection, crossover, mutation and reinsertion.

3 Conclusion and Future Work

This paper reported on three novel approaches to generate and optimize input signals for search-based testing.

The preliminary results of ongoing experimental work already indicate that the approaches are able to approximate given signals and thus are suitable to solve problems of evolutionary testing. However, more work still needs to be done in order for the approach to be capable of reliably generating valuable signals. An example for one important extension is the incorporation of signal noise.

References

1. McMinn, P.: Search-based software test data generation: A survey. Software Testing, Verification and Reliability **14** (2004) 105–156
2. Jones, B.F., Sthamer, H., Eyres, D.E.: Automatic test data generation using genetic algorithms. Software Engineering Journal **11** (1996) 299–306
3. Wegener, J., Baresel, A., Sthamer, H.: Evolutionary test environment for automatic structural testing. Information and Software Technology **43** (2001) 841–854
4. Baresel, A., Pohlheim, H., Sadeghipour, S.: Structural and functional sequence test of dynamic and state-based software with evolutionary algorithms. In: Proceedings of Genetic and Evolutionary Computation Conference. (2003) 2428–2441
5. Banzhaf, W., Francone, F.D., Keller, R.E., Nordin, P.: Genetic programming: an introduction: on the automatic evolution of computer programs and its applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1998)
6. Brameier, M.: On Linear Genetic Programming. PhD thesis, Fachbereich Informatik, Universität Dortmund, Germany (2004)
7. Perkis, T.: Stack-based genetic programming. In: Proceedings of the 1994 IEEE World Congress on Computational Intelligence. Volume 1., Orlando, Florida, USA, IEEE Press (1994) 148–153
8. Langdon, W.B., Banzhaf, W.: Repeated sequences in linear GP genomes. In: Late Breaking Papers at the 2004 Genetic and Evolutionary Computation Conference, Seattle, Washington, USA, AAAI (2004)