Applied Soft Computing 23 (2014) 91–103

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A hybrid optimization approach to conformance testing of finite automata

Krzysztof Zaniewski [a,b,*], Witold Pedrycz [a,c,d]

[a] Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
[b] Department of Deal Life Cycle, DONG Energy, Warsaw, Poland
[c] Department of Electrical and Computer Engineering, University of Alberta, Edmonton T6R 2G7 AB, Canada
[d] Department of Electrical and Computer Engineering, Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia

ARTICLE INFO

Article history:
Received 29 December 2012
Received in revised form 17 May 2014
Accepted 19 May 2014
Available online 11 June 2014

Keywords:
Metaheuristic algorithms
Hybrid approach
Genetic Algorithm
Simulated Annealing
Finite state machine
State verification

ABSTRACT

Unique Input–Output sequences (UIOs) are quite commonly used in conformance testing. Unfortunately finding UIOs of minimal length is an NP hard problem. This study presents a hybrid approach to generate UIOs automatically on a basis of the finite state machine (FSM) specification. The proposed hybrid approach harnesses the benefits of hill climbing (Greedy search) and heuristic algorithm. Hill climbing, which exploits domain knowledge, is capable of quickly generating good result however it may get stuck in local minimum. To overcome the problem we used a set of parameters called the seed, which allows the algorithm to generate different results for a different seed. The hill climbing generates solutions implied by the seed while the Genetic Algorithm is used as the seed generator. We compared the hybrid approach with Genetic Algorithm, Simulated Annealing, Greedy Algorithm, and Random Search. The experimental evaluation shows that the proposed hybrid approach outperforms other methods. More specifically, we showed that Genetic Algorithm and Simulated Annealing exhibit similar performance while both of them outperform Greedy Algorithm. Finally, we generalize the proposed hybrid approach to seed-driven hybrid architectures and elaborate on how it can be adopted to a broad range of optimization problems.

Introduction

Testing has become more complex in the recent years given a continuously growing size and complexity of modern systems. Conformance testing is applied to verify if a designed (implemented) automata meets a series of specified requirements. Conformance testing involves a model of the finite state machine, which captures the requirements of the system to be designed [1–3]. An implemented version of a finite state machine is treated as a black box and compared with its specification. To verify if the implemented machine complies with its requirements, we generate an input for the machine and observe the resulting outputs. With this regard, various testing methods were proposed, say U-method [4,5], W-method [5,6], T-method [5,7], and D-method [5,8]. Each of these methods comes with some advantages however there are some limitations. The T-method is unable to detect state fault [5]. The W-method generates the longest test sequences [9]. The D-method generates a distinguishing sequence, which may not exist for every machine [5,9]. In fact, only 17% of FSMs have a distinguishing sequence [9].

In this study, we focus on the U-method because this method is able to detect state fault [5], produce the shortest test length [9]. Furthermore we are able to use the U-method when a distinguishing sequence does not exist for the given FSM. The U-method generates Unique Input–Output sequences (UIOs) of minimal length for each machine state to detect state fault and UIOs exists for each state on 99% of the FSM [9]. Let us recall that the Unique Input–Output (UIO) sequence is an input sequence, which once executed on the selected state produces a unique output. However generating the shortest Unique Input–Output sequences is an NP-hard problem [10] and because of this, various metaheuristic techniques may be applied. Such techniques like Genetic Algorithms [11,12], Simulated Annealing [13,14], 1+1 [15], Particle Swarm Optimization [16,17], Artificial Bee Colony [18,19] have been shown to be effective in solving NP-hard problems.

The main objective of this study is to propose, analyze, and experiment with the hybrid approach to generate minimal UIO sequences and to quantify its effectiveness. The results obtained when using the proposed approach are compared with those

* Corresponding author at: Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland. Tel.: +48 501216097.
E-mail address: krzysztof.zaniewski@gmail.com (K. Zaniewski).

http://dx.doi.org/10.1016/j.asoc.2014.05.018
1568-4946/

Terms and acronyms

FSM	finite state machine
GA	Genetic Algorithm
SA	Simulated Annealing
UIO	Unique Input–Output
UIOs	Unique Input–Output sequences

produced by a generic version of the Genetic Algorithm, Simulated Annealing, Greedy Search, and Random Search. A detailed study is provided on how the approaches are affected by the topology and size of the finite state machine. We also show how these algorithms perform over time.

In this paper, we propose a new hybrid approach called Genetic-Hill Climbing. We show that this approach leads to better results in comparison with those produced by some other techniques. The hybrid approach is composed of a hill climbing algorithm, which generates results based on knowledge about the problem and machine architecture. The hill climbing algorithm without any input parameters generates the same sequences every time. To improve the hill climbing we use a set of parameters, which guides the process of optimization. The set of parameters is referred to as a seed and the Genetic Algorithm is used as the seed generator.

This paper is structured into seven sections. We begin with offering some background by looking at the current state of affair in the field of finite state machine conformance testing. Section "The proposed hybrid algorithm" gives details on the hybrid approach and generalizes the proposed approach to seed-driven hybrid architectures. In Section "Experiments", a concise description of the testing environment and algorithm parameters selection is followed by the results of experiments and their detailed analysis. Following section provides a real-world case study. Finally, we draw the conclusions and identify future research direction.

Some preliminaries

In this section, we present some prerequisites and present a current status on conformance testing of finite state machines.

Finite state machine

The finite state machine (finite automata) is an abstract model, which describes behavior of dynamic system based on a state transformation table. Finite state machines are in common usage. They are used for describing logical circuits [20], network protocols [21] and computer applications [3].

Formally speaking, a State Machine is described in the form (I, O, S, S_s, f, g), where I: a finite non empty set of input objects; $I = \{I_1, I_2, I_3 \ldots, I_a\}$, O: a finite non empty set of output objects; $O = \{O_1, O_2, O_3 \ldots, O_o\}$, S: a finite non empty set of states; $S = \{S_1, S_2, S_3 \ldots, S_n\}$, a: a number of inputs in the fsm, o: a number of outputs in the fsm, n: a number of states in the fsm, f: a state transformation function; $f : S \times I \rightarrow S$, g: an output function; $g : S \times I \rightarrow O$, S_s: a start state; $S_s \in S$.

There are two types of finite automata, namely Mealy machines [1] and Moore machines [2].

In this paper, we consider Mealy machines because the machines of this type produce a lower number of states than the equivalent Moore machines. In the Mealy automata, each state has a list of actions, which may be executed. When being in a certain state and provided with a certain input, the machine moves to the next state (identified in the transformation table) and generate a certain output. An example of the finite Mealy state machine is shown in Fig. 1. For example, if we start from state S_1 and run a sequence of actions 'aa', the machine moves to state S_3 through state S_2 and produce the output sequence '21'.

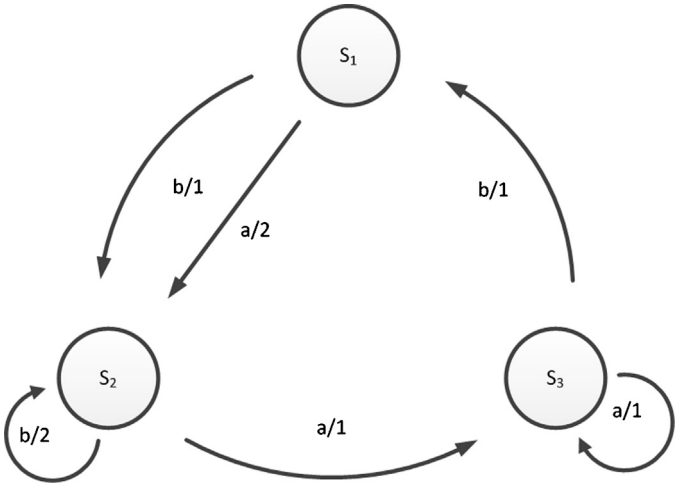

Fig. 1. An example of Mealy's machine.

Conformance testing of finite state machines

Conformance testing of a finite state machine requires comparing the machine specification with its implementation. We encounter the following types of faults [5,10]:

- Output fault
- State fault

An output error arises when the output coming from a test suite disagrees with the machine specification. Whereas a state error appears when after test execution the finite state machine is not in an expected state. The output error can be discovered easily by comparing an actual test output with an expected test output derived from the specification. The basic method for finite state machine testing is the Transition Tour (T-method) [5,7]. When applying the T-method, we move through each transition at least once and check the output. Finding the shortest sequence, which moves through all transitions is also known as The Chinese Postman Problem [22] and an effective solution for the problem exists [22]. However the T-method detects only the output error. The state error can be discovered after each transition test execution by checking if the machine is in a correct state. But this is not as easy as it looks. To check if a machine is in a proper state we need to run a special sequence, which verifies the state. There are three state verification methods:

- U-method or unique input/output sequence (UIO) [4,5]
- D-method or distinguishing sequence (DS) [5,8]
- W-method or characterization set (W-set) [5,6]

Let us briefly highlight the essence of these three methods.

D-method

The method creates a sequence called the distinguishing sequence, which generates a different output for each state. For example, the distinguishing sequence 'bb' for the graph in Fig. 1 produces the following outputs:

- For state S_1 – 12
- For state S_2 – 22
- For state S_3 – 11