# Computing UIO Sequences using Parallel GAs ⋆

Qiang Guo, John McCall and Horacio González-Vélez

IDEAS Research Institute/School of Computing
The Robert Gordon University

**Abstract.** Unique Input/Output (UIO) sequence is an important state verification technique in Finite State Machine (FSM) based testing. Computing UIOs is *NP*-hard. Genetic Algorithms (GAs) were applied to compute UIOs [1] where an objective function is defined to guide GAs to search UIOs. The technique was experimentally evaluated for its effectiveness but also shown for its high computational cost. In this paper, we look at computing UIOs using parallel GAs. By making use of multicore resources, we intend to improve computational performance. Two parallel GA models were proposed. The models explore parallel patterns from GAs and FSMs and map them to the available multicore processors.

*Keywords:* UIOs, Finite State Machine, GA; Parallel Patterns, Multicore.

## 1 Introduction

Finite state machines (FSMs) have been widely used in system modelling and testing. In FSM-based testing, Unique Input/Output (UIO) sequence is an important technique for state verification. However, computing UIOs is *NP*-hard. Guo *et al*. investigated constructing UIOs using Genetic Algorithms (GAs) [1]. Based on the patterns of State Splitting Trees (SSTs) [2], an objective function was defined to guide a GA to explore UIOs from the FSMs under test. The model was experimentally evaluated for the effectiveness in finding UIOs, but also showed high computational cost.

Deriving the SST for an input sequence $seq_i$ requires that $seq_i$ executes $M$ once for each of its $n$ states as the initial state (moving $M$ from $s_0$ to $s_i$). This leads to a high computational cost in finding UIOs from FSMs with a large number of states. Moreover, a large GA population size makes the computation even more expensive. To alleviate the computational burden and achieve a higher computational performance, it is desirable to make use of high performance resources such as multicore processors to undertake such computationally expensive tasks. To do so, the proposed model must be designed and implemented to be suitable for parallel computing.

This paper investigates computing UIOs under multicore systems. Two parallel GA models are proposed to compute UIOs. By extending existing micro-grained and coarse-grained models [3], the proposed models explore parallel patterns from both GAs and SSTs. The parallel patterns are mapped to the available multicore processors. With such parallel operations, computational performance is expected to be improved.

## 2 Backgound

### 2.1 FSM and UIOs

A (deterministic) FSM $M$ is defined as a 6-tuple $M = (I, O, S, \delta, \lambda, s_0)$ where $I$ is a non-empty set of inputs, $O$ a non-empty set of outputs, $S$ a finite and non-empty set of

⋆ Work funded by the FP7 project *ParaPhrase*, under contract no.: 288570

states, $\delta$ the transition function, $\delta : S \times I \rightarrow S$, $\lambda$ the output function, $\lambda : S \times I \rightarrow O$, and $s_0$ the initial state.

In FSM based testing, a standard testing strategy is defined in two steps: the transition Input/Outpt (I/O) check and the tail state verification. The former determines whether a transition of an implementation under test (IUT) produces the expected output while the latter checks that the IUT arrives at the specified state when a transition test is finished. UIO sequence is a very important technique applied for state verification. A UIO sequence of state $s_i$ is an I/O sequence $x/y$, that may be observed from $s_i$ , such that the output sequence produced by the machine in response to $x$ from any other state is different from $y$, i.e. $\lambda(s_i, x) = y$ and $\lambda(s_j, x) \neq y$ for any $i \neq j$.

A pre-requirement of UIO based testing is that at least one UIO sequence is available for each state $s_i$. However, computing UIOs is *NP*-hard. A State Splitting Tree (SST) based approach [2] is designed to compute UIOs. Figure 1 explains the approach with an example where an FSM has 6 states, $\{s_1, ..., s_6\}$ with an input set $\{a, b\}$ and an output set $\{x, y\}$. The tree starts with a root (N(0,0)) that contains the complete set of states. Suppose, when responding to an input $a$, states $\{s_1, s_3, s_5\}$ produce $x$

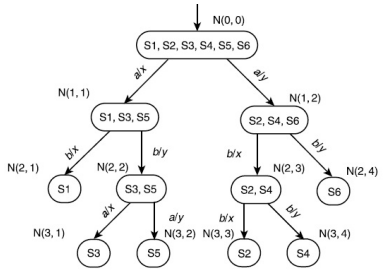


**Fig. 1.** A state splitting tree from an FSM.

while $\{s_2, s_4, s_6\}$ produce $y$, then $\{s_1, s_3, s_5\}$ and $\{s_2, s_4, s_6\}$ are distinguishable by $a$. Nodes N(1,1) and N(1,2) are created. If we then apply $b$, the state reached from $s_1$ by $a$ produces $x$, while the states reached from $\{s_3, s_5\}$ by $a$ produce $y$. Thus $ab$ distinguish $s_1$ from $\{s_3, s_5\}$. Nodes N(2,1) and N(2,2) are created. Repeating this process, we can get all discrete partitions as shown in Figure 1. A path from a discrete partition node to the root node forms a UIO for the state related to this node.

### 2.2 Computing UIOs with GAs

Finding inputs to construct SSTs is also *NP*-hard. Guo *et al*. thus investigated constructing UIOs using GAs [1]. Based upon the patterns of SSTs, an objective function is defined to guide a GA to explore UIOs. While applying an input sequence to an FSM, at each stage of a single input, the constructed SST is evaluated by Equation 1

$$f_i = \frac{x_i e^{(\delta x_i)}}{l_i^{\gamma}} + \alpha \frac{(y_i + \delta y_i)}{l_i} \tag{1}$$

where $i$ refers to the $i^{th}$ input. $x_i$ denotes the number of existing discrete partitions, $\delta x_i$ the number of new discrete partitions caused by the $i^{th}$ input, $y_i$ the number of existing separated groups, $\delta y^i$ the number of new groups, $l_i$ the length of the input sequence up to the $i^{th}$ element, while $\alpha, \gamma$ are constant settings.

Equation 1 consists of an exponential part, $f_{e(i)} = x_i e^{(\delta x_i)}/l_i^{\gamma}$, and a linear part, $f_{l(i)} = \alpha(y_i + \delta y_i)/l_i$. We can see that the occurrence of discrete partitions makes $x_i$ and $\delta x_i$ increase. This leads to $x_i e^{(\delta x_i)}$ being increased exponentially, which contributes to increasing $f_{e(i)}$ exponentially; the increment of $l_i$ makes $l_i^{\gamma}$ decrease exponentially, which reduces $f_{e(i)}$ exponentially. As long as $x_i e^{(\delta x_i)}$ has a faster dynamics than $l_i^{\gamma}$, $f_{e(i)}$ will be increased exponentially; otherwise, $f_{e(i)}$ decreases exponentially.

The exponential part encourages the early occurrence of discrete partitions and punishes the increment of an input sequence's length. The linear part rewards partitioning when discrete partitions have not been produced. This favours the discovery of more UIOs in the next partitioning. The overall fitness value is defined as $f = \frac{1}{N} \times \sum_{i=1}^{N}(f_i)$ where $N$ is the length of the sequence.

## 3 Computing UIOs with Parallel GA Frameworks

The model defined in [1] is effective in finding UIOs but computationally expensive, especially when an FSM has a large number of states. Early experiments showed, for an FSM with 12 states, with a population size of 600, it requires nearly 8 hours to complete the computation for 300 generations in a single PC. To alleviate computational burden, high performance computational resources such as multicore processors are useful. To make use of these resources, parallelisms must be defined to make the model suitable for parallel computing.

Two models have been proposed to parallelise GAs [3]. The Micro-Grained (GM) model defines a master-slave structure. A master processor is applied to execute all ge-

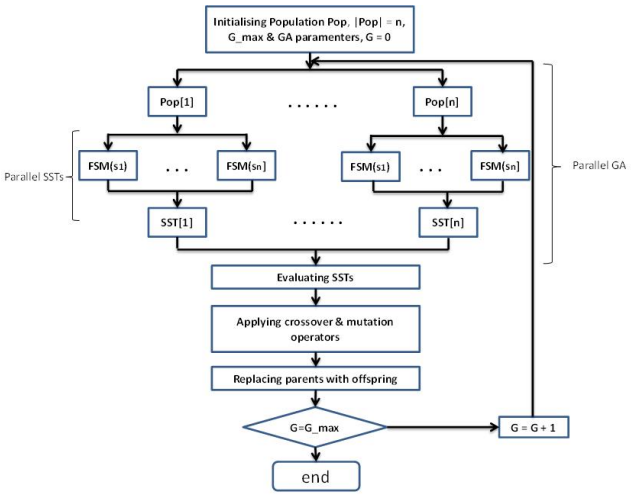


**Fig. 2.** SST based micro-grained model.

netic operations except evaluations while a set of slave processors are devised to evaluate individuals in parallel. The Coarse-Grained (CG) model is a distributed population model. This model splits the population into multiple subpopulations where individuals in a subpopulation are evaluated and optimised by a GA within this subpopulation, but can also be migrated to another subpopulation.

Both GM and CG models can be used to compute UIOs under multicore environments. It can be noted that the model defined in [1] favours the construction of SSTs

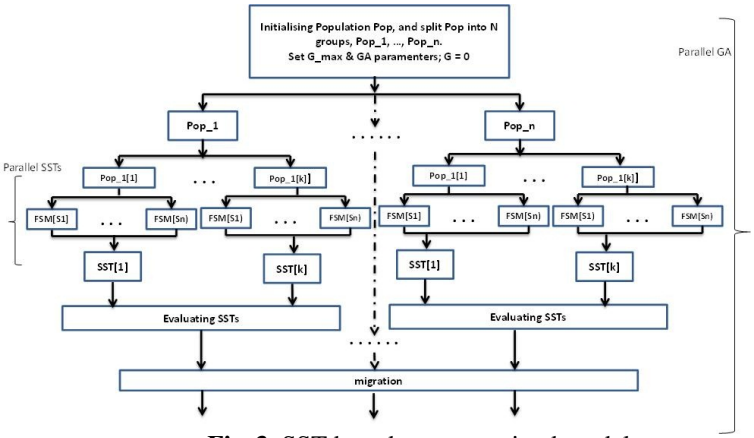


**Fig. 3.** SST based coarse-grained model.

in a parallel fashion. Thus, in this work, we extend the GM and CG models by integrating SST parallelisms. The extended models are presented in Figure 2 and 3 where both models consist of two layers of parallelism, namely, GA parallelism and SST parallelism. With such an operation, computational tasks can be further split and parallelised. This helps to maximise the use of the available multicore resources.

## 4 Summary and Future Work

This paper proposes two parallel GA models used to compute UIOs under multicore environments. Our work extends two existing parallel models, the GM model and the CG model, by integrating an SST parallelism layer. By doing so, the extended models promote maximum use of the available multicore resources for computing UIOs.

We are currently implementing the proposed models in C++ using OpenMP and MPI, and developing case studies to evaluate the computational performance. It has also been planned to integrate the models to a proven parallel framework FastFlow [4] which is specifically designed for cache-coherent shared-memory multicore systems.

## References

1. Guo, Q., Hierons, R.M., Harman, M., Derderian, K.: Constructing multiple unique input/output sequences using metaheuristic optimisation techniques. IEEE Proceedings - Software **152** (2005) 127 – 140
2. Lee, D., Yannakakis, M.: Principles and methods of testing finite state machines - a survey. Proc. IEEE **84** (1996) 1090 – 1122
3. Hiroyasu, T., Yamanaka, R., Yoshimi, M., Miki, M.: A framework for genetic algorithms in parallel environments. IPSJ SIG Notes **84** (2011) 1 – 6
4. Aldinucci, M., Danelutto, M., Kilpatrick, P., Meneghin, M., Torquati, M.: Accelerating Code on Multi-cores with Fastflow. In: Euro-Par 2011. Volume 6853 of LNCS. (2011) 170–181