Genetic Programming in automated test code generation for a multi-threaded microprocessor.

Neow Way Yuh
MSc. Machine Learning and Data Mining.
Supervised by Dr. Kerstin Eder and
Mr. Peter Hedinger
Introduction

• The complexity of modern microprocessors makes verification difficult.
• Recent research on Coverage-Directed test Generation (CDG) uses machine learning approaches.
• Genetic Programming (GP) is one of the machine learning methods used in CDG.
• Initial experiments with GP in multi-threaded microprocessor verification show potential.
• Presentation starts with introduction to GP and a case study with the XMOS multi-threaded microprocessor.
Fundamentals of Genetic Programming (GP)

- GP is based on the natural evolution process.
- Darwin Theorem: Natural selection promotes favourable heritable traits in successive generations.
- In GP, each individual in a population is given a quantitative measurement to reflect the quality of an individual relative to the environment.
GP in test code generation

- Selection algorithm samples two individuals from the population based on their fitness values (e.g. Higher fitness value individuals get sampled more often).
- The parents are merged using genetic operators to produce new individuals.
- An external simulator is used to evaluate the fitness of the new individuals.
- GP implementation in Coverage-Directed test Generation requires less expert knowledge compared to other methods (e.g. Bayesian Network).
Case study: XMOS multi-threaded microprocessor

- Software Defined Silicon (SDS).
- Up to 8 threads running simultaneously in one single core.
- Experiment focuses on channel communication.
- Channel communication design requires a number of threads to exercise the corner cases.
- However, the timing window to trigger corner cases, such as race conditions, is very narrow.
Implemented CDG loop

- The experiment extends the existing MicoGP* program.
- Expert knowledge is encoded in an instruction library.
- An individual is a graph; the nodes in a graph are constructed from the basic building blocks in the library.
- The extended MicroGP includes 2 crossover and 5 mutation operators.
- A double looped system is used to enhance the performance of MicroGP.

* MicroGP research group
http://www.cad.polito.it/research/microgp.html
Encoding multi-threaded test code in GP

- A test program can be represented by a graph; a node in a graph maps to a channel operation and a branch represents the content of a new thread.
Feedback measurements

• Improves basic coverage measurements such as line coverage and branch coverage.
• Minimises simulation cycles.
• Minimises the main graph to encourage more threads in test code.
• Balances the distribution of operations among the threads.
• Increases operation on channel IO.
• All feedback measurements are optimised together with different priority.
**Experiment Results**

- Test code generated by MicroGP method is significantly better than human engineer and randomly generated sequences.
- Actual line coverage improved to 94% and up to 50% cycle reduction.

<table>
<thead>
<tr>
<th>Random Seed</th>
<th>Line Coverage</th>
<th>Simulation Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>70</td>
<td>12</td>
</tr>
<tr>
<td>345</td>
<td>75</td>
<td>3000</td>
</tr>
<tr>
<td>678</td>
<td>80</td>
<td>5000</td>
</tr>
<tr>
<td>901</td>
<td>80</td>
<td>7000</td>
</tr>
<tr>
<td>234</td>
<td>75</td>
<td>9000</td>
</tr>
<tr>
<td>567</td>
<td>75</td>
<td>11000</td>
</tr>
<tr>
<td>890</td>
<td>70</td>
<td>13000</td>
</tr>
<tr>
<td>123</td>
<td>65</td>
<td>15000</td>
</tr>
<tr>
<td>456</td>
<td>65</td>
<td>17000</td>
</tr>
<tr>
<td>789</td>
<td>65</td>
<td>19000</td>
</tr>
</tbody>
</table>

**Graphs:**
- Line Coverage vs. Random Seed
- Simulation Cycle vs. Random Seed
Thank you.
Questions?