Finding Liveness Errors with ACO

Francisco Chicano and Enrique Alba
• Nowadays software is very complex

• An error in a software system can imply the loss of lot of money …

… and even human lifes

• Techniques for proving the correctness of the software are required

• Model checking → fully automatic
• **Objective:** Prove that model $M$ satisfies the property $\phi$: $M \models \phi$

• **HSF-SPIN:** the property $\phi$ is an LTL formula

---

```
Model $M$
```

```
LTL formula $\neg \phi$
```

```
Intersection Büchi automaton
```
• **Objective:** Prove that model $M$ satisfies the property $f$: $M \models f$

• **HSF-SPIN:** the property $f$ is an LTL formula
Objective: Prove that model $M$ satisfies the property $\varphi: M \models \varphi$

HSF-SPIN: the property $\varphi$ is an LTL formula

Model $M$

LTL formula $\neg \varphi$

Intersection Büchi automaton

Using Nested-DFS
State Explosion Problem

- Number of states very large even for small models

  Example: Dining philosophers with $n$ philosophers $\rightarrow 3^n$ states
  20 philosophers $\rightarrow 1039$ GB for storing the states

- Solutions: collapse compression, minimized automaton representation, bitstate hashing, partial order reduction, symmetry reduction

- Large models cannot be verified but errors can be found
The search for errors can be directed by using heuristic information.

Different kinds of heuristic functions have been proposed in the past:

- **Formula-based heuristics**
- **Structural heuristics**
- **Deadlock-detection heuristics**
- **State-dependent heuristics**
Safety property

\( \forall \sigma \in S^\omega : \sigma \not\models \mathcal{P} \Rightarrow (\exists i \geq 0 : \forall \beta \in S^\omega : \sigma_i \beta \not\models \mathcal{P}) \)

- Counterexample \( \equiv \) path to accepting state
- Graph exploration algorithms can be used: DFS and BFS

Liveness property

\( \forall \alpha \in S^* : \exists \beta \in S^\omega, \alpha \beta \models \mathcal{P} \)

- Counterexample \( \equiv \) path to accepting cycle
- It is not possible to apply DFS or BFS
Metaheuristic Algorithms

• Designed to solve optimization problems
  ➢ Maximize or minimize a given function: the fitness function
• They can find “good” solutions with a “reasonable” amount of resources
Metaheuristics Classification

**Single solution**
- Greedy Randomized Adaptive Search Procedure
- Iterated Local Search
- Variable Neighborhood Search
- Tabu Search
- Simulated Annealing
- Iterative Improvement
- Guided Local Search

**Population**
- Estimation of Distribution Algorithms
- Evolutionary Computation
- Scatter Search
- Ant Colony Optimization
- Particle Swarm Optimization

Metaheuristics
- ACO
- ACOhg
- ACOhg-live
**ACO: Introduction**

- Ant Colony Optimization (ACO) metaheuristic is inspired by the foraging behaviour of real ants

```
procedure ACOMetaheuristic
    ScheduleActivities
        ConstructAntsSolutions
        UpdatePheromones
    end ScheduleActivities // optional
end procedure
```
ACO: Construction Phase

- The ant selects its next node stochastically.

- The probability of selecting one node depends on the pheromone trail and the heuristic value (optional) of the edge/node.

- The ant stops when a complete solution is built.
ACO: Pheromone Update

• Pheromone update
  
  ➢ During the construction phase
  \[ \tau_{ij} \leftarrow (1 - \xi)\tau_{ij} \quad \text{with} \quad 0 \leq \xi \leq 1 \]

  ➢ After the construction phase
  \[ \tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \Delta\tau_{ij}^{bs} \quad \text{with} \quad 0 \leq \rho \leq 1 \]

• Trail limits (particular of MMAS)
  
  ➢ Pheromones are kept in the interval \([\tau_{\text{min}}, \tau_{\text{max}}]\)
  
  \[ \tau_{\text{max}} = \frac{Q}{\rho} \quad \tau_{\text{min}} = \frac{\tau_{\text{max}}}{a} \]
ACOhg: Huge Graphs Exploration

The length of the ant paths is limited by $\lambda_{\text{ant}}$

Initial node

Starting nodes for path construction change

After $\sigma_s$ steps

Second stage

Third stage

What if…?

Objective node
The search is an alternation of two phases

- **First phase**: search for accepting states
- **Second phase**: search for cycles from the accepting states

ACOhg-live Pseudocode

```
1: repeat
2:  accept = acohg1.findAcceptingStates(); {First phase}
3:  for node in accept do
4:      acohg2.findCycle(node); {Second phase}
5:  end if
6:  return acohg2.acceptingPath();
7: end for
8: acohg1.insertTabu(accept);
9: until empty(accept)
10: return null;
```
ACOhg-live

- The search is an alternation of two phases

  - **First phase**: search for accepting states
  - **Second phase**: search for cycles from the accepting states

ACOhg-live Pseudocode

```plaintext
1: repeat
2:   accpt = acohg1.findAcceptingStates(); {First phase}
3:   for node in accpt do
4:     acohg2.findCycle(node); {Second phase}
5:   if acohg2.cycleFound() then
6:     return acohg2.acceptingPath();
7:   end if
8:   end for
9:   acohg1.insertTabu(accpt);
10: until empty(accpt)
11: return null;
```
The search is an alternation of two phases

- **First phase:** search for accepting states
- **Second phase:** search for cycles from the accepting states

**ACOhg-live Pseudocode**

```
1: repeat
2:    acpt = acohg1.findAcceptingStates();  {First phase}
3:    for node in acpt do
4:       acohg2.findCycle(node);  {Second phase}
5:       if acohg2.cycleFound() then
6:          return acohg2.acceptingPath();
7:       end if
8:    end for
9:    acohg1.insertTabu(acpt);
10:   until empty(acpt)
11:   return null;
```
We selected 7 Promela models for the experiments.

<table>
<thead>
<tr>
<th>Model</th>
<th>LoC</th>
<th>Scalable</th>
<th>Processes</th>
<th>LTL formula (liveness)</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter</td>
<td>64</td>
<td>no</td>
<td>2</td>
<td>□(p → ◊q) ^ □(r → ◊s)</td>
</tr>
<tr>
<td>giopi</td>
<td>740</td>
<td>yes</td>
<td>i+3(j+1)</td>
<td>□(p → ◊q)</td>
</tr>
<tr>
<td>phi</td>
<td>57</td>
<td>yes</td>
<td>j+1</td>
<td>□(p → ◊q)</td>
</tr>
</tbody>
</table>

Parameters for ACOhg-live

<table>
<thead>
<tr>
<th>Parameter</th>
<th>msteps</th>
<th>colsize</th>
<th>λ\text{ant}</th>
<th>σ_s</th>
<th>ξ</th>
<th>a</th>
<th>ρ</th>
<th>α</th>
<th>β</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st phase</td>
<td>100</td>
<td>10</td>
<td>20</td>
<td>4</td>
<td>0.7</td>
<td>5</td>
<td>0.2</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>2nd phase</td>
<td>20</td>
<td>4</td>
<td>4</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ACOhg-live implemented in HSF-SPIN

100 independent executions
**Promela Models**

- **We selected 7 Promela models for the experiments**

<table>
<thead>
<tr>
<th>Model</th>
<th>LoC</th>
<th>Processes</th>
<th>LTL formula (liveness)</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter</td>
<td>64</td>
<td>yes</td>
<td>2 □(p → ◊q) ^ □(r → ◊s)</td>
</tr>
<tr>
<td>giopi</td>
<td>740</td>
<td>yes</td>
<td>i+3(j+1) □(p → ◊q)</td>
</tr>
<tr>
<td>phi</td>
<td>57</td>
<td>yes</td>
<td>j+1 □(p → ◊q)</td>
</tr>
</tbody>
</table>

- **Parameters for ACOhg-live**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>msteps</th>
<th>colsize</th>
<th>λant</th>
<th>σs</th>
<th>ξ</th>
<th>a</th>
<th>ρ</th>
<th>α</th>
<th>β</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st phase</td>
<td>100</td>
<td>10</td>
<td>20</td>
<td>4</td>
<td>0.7</td>
<td>5</td>
<td>0.2</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>2nd phase</td>
<td>100</td>
<td>20</td>
<td>4</td>
<td>0.7</td>
<td>0.5</td>
<td>5</td>
<td>0.2</td>
<td>1.0</td>
<td>2.0</td>
</tr>
</tbody>
</table>

- **ACOhg-live implemented in HSF-SPIN**

- **100 independent executions**
Results I: Comparison of Heuristic Information

- Comparison of $H_{ham}$ and $H_{fsm}$

<table>
<thead>
<tr>
<th>Models</th>
<th>Measure</th>
<th>$H_{ham}$</th>
<th>$H_{fsm}$</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter</td>
<td>Hit rate</td>
<td>100/100</td>
<td>100/100</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>28.54</td>
<td>30.68</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>1925.00</td>
<td>1925.00</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>88.90</td>
<td>90.00</td>
<td>-</td>
</tr>
<tr>
<td>giop2</td>
<td>Hit rate</td>
<td>100/100</td>
<td>100/100</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>43.09</td>
<td>43.76</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>2834.84</td>
<td>2953.76</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>817.50</td>
<td>747.50</td>
<td>+</td>
</tr>
<tr>
<td>giop6</td>
<td>Hit rate</td>
<td>100/100</td>
<td>100/100</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>58.41</td>
<td>58.77</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>5418.32</td>
<td>5588.04</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>16049.10</td>
<td>8733.50</td>
<td>-</td>
</tr>
<tr>
<td>giop10</td>
<td>Hit rate</td>
<td>100/100</td>
<td>100/100</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>61.10</td>
<td>62.85</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>9669.80</td>
<td>9316.67</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>87236.00</td>
<td>43059.07</td>
<td>+</td>
</tr>
<tr>
<td>ph18</td>
<td>Hit rate</td>
<td>100/100</td>
<td>100/100</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>52.38</td>
<td>51.36</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>2097.52</td>
<td>2014.32</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>2271.40</td>
<td>2126.10</td>
<td>+</td>
</tr>
<tr>
<td>ph114</td>
<td>Hit rate</td>
<td>99/100</td>
<td>99/100</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>74.68</td>
<td>76.05</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>2593.37</td>
<td>2496.07</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>9369.90</td>
<td>8070.30</td>
<td>+</td>
</tr>
<tr>
<td>ph120</td>
<td>Hit rate</td>
<td>99/100</td>
<td>98/100</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>95.28</td>
<td>97.39</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>3324.26</td>
<td>3244.67</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>21323.54</td>
<td>18064.90</td>
<td>+</td>
</tr>
</tbody>
</table>
### Results II: Comparison of ACOhg-live and NDFS

#### Comparison of ACOhg-live and NDFS

<table>
<thead>
<tr>
<th>Models</th>
<th>Measure</th>
<th>ACOhg-live</th>
<th>Nested-DFS</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter</td>
<td>Hit rate</td>
<td>100/100</td>
<td>1/1</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>30.68</td>
<td>64.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>1925.00</td>
<td>1873.00</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>90.00</td>
<td>0.00</td>
<td>+</td>
</tr>
<tr>
<td>giop2</td>
<td>Hit rate</td>
<td>100/100</td>
<td>1/1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>43.76</td>
<td>298.00</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>2953.76</td>
<td>7865.00</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>747.50</td>
<td>240.00</td>
<td>+</td>
</tr>
<tr>
<td>giop6</td>
<td>Hit rate</td>
<td>100/100</td>
<td>0/1</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>58.77</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>5588.04</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>8733.50</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>giop10</td>
<td>Hit rate</td>
<td>86/100</td>
<td>0/1</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>62.85</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>9316.67</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>43059.07</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>phi8</td>
<td>Hit rate</td>
<td>100/100</td>
<td>1/1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>51.36</td>
<td>3405.00</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>2014.32</td>
<td>4005.00</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>2126.10</td>
<td>40.00</td>
<td>+</td>
</tr>
<tr>
<td>phi14</td>
<td>Hit rate</td>
<td>99/100</td>
<td>1/1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>76.05</td>
<td>10001.00</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>2496.07</td>
<td>59392.00</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>8070.30</td>
<td>2300.00</td>
<td>+</td>
</tr>
<tr>
<td>phi20</td>
<td>Hit rate</td>
<td>98/100</td>
<td>1/1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>97.39</td>
<td>10001.00</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Mem. (KB)</td>
<td>3244.67</td>
<td>392192.00</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>Time (ms)</td>
<td>18064.90</td>
<td>17460.00</td>
<td>-</td>
</tr>
</tbody>
</table>
How to use ACOhg-live

- ACOhg-live should be used in the first/middle stages of the software development, when software errors are expected.
- ACOhg-live can also be used in other phases of the software development for testing concurrent software.
Conclusions

- **ACOhg-live** is the first algorithm based on metaheuristics (to the best of our knowledge) applied to the search for liveness errors in concurrent models.

- The heuristic function based on finite state machines is a better guide in the second phase of **ACOhg-live**.

- **ACOhg-live** is able to outperform Nested-DFS in efficacy and efficiency in the search for liveness errors.

Future Work

- Use of **Strongly Connected Components** of the never claim graph for improving the search (in progress).

- Analysis of parameterization for reducing the parameters.

- Include **ACOhg-live** into JavaPathFinder for finding liveness errors in Java programs.
Finding Liveness Errors with ACO

Thanks for your attention !!!!