A Backward Reachability Algorithm for Parameterized Systems on Weak Memory

Sylvain Conchon, David Declerck and Fatiha Zaidi

Tool Download

The algorithm described in this poster has been successfully implemented in a new version of the Cubicle model checker called Cubicle-W. This tool, as well as various examples, can be downloaded here:
https://www.lri.fr/~declerck/cubiclew/

Weak Memory Parameterized Systems

★ Weak memory:
  * order of memory accesses ≠ interleaving of memory instructions: algorithms are harder to check
  * different “flavors” of reorderings: TSO, PSO, ARM...
  * we adopt a TSO-like model, as shown on the left
  * reorderings can be prevented using fences

★ Parameterized systems:
  * concurrent systems
  * unbounded number of processes
  * process-indexed arrays

Our example: a (naive and inefficient) mutual exclusion algorithm using a process-indexed array of booleans X, and each process p executes the automaton on the right.

Cubicle-W code

The automaton on the left can be expressed as a parameterized transition system in the language of Cubicle-W.

```c
#include <assert.h>

int main()
{
  PC[proc] := Idle
  X[proc] := false

  while (true)
  {
    proc := proc + 1
    X[proc] := true
  }
}
```

Our approach

★ The base framework:
  * Model Checking Modulo Theories (MCMT)
  * checks safety properties of parameterized systems
  * assumes a sequentially consistent (SC) memory
  * uses a backward reachability algorithm

★ Our extension:
  * adds weak memory reasoning using an axiomatic model
  * maps memory instructions to read/write events
  * builds a global-happens-before relation over events

Benchmarks

The implementation was tested on some typical concurrent algorithms. Some algorithms are incorrect due to the effects of weak memory. In this case, we created fixed versions of these algorithms by adding fences. The test machine features an Intel Core i7 CPU @ 2.9 Ghz and 8GB of RAM.

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>Correct?</th>
<th>Analysis Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>naive_mutex N</td>
<td>No</td>
<td>0.05s</td>
</tr>
<tr>
<td>naive_mutex f N</td>
<td>Yes</td>
<td>0.06s</td>
</tr>
<tr>
<td>lamport N</td>
<td>No</td>
<td>0.42s</td>
</tr>
<tr>
<td>lamport f N</td>
<td>Yes</td>
<td>0.62s</td>
</tr>
<tr>
<td>spinlock N</td>
<td>Yes</td>
<td>0.07s</td>
</tr>
<tr>
<td>sense_rev N</td>
<td>Yes</td>
<td>0.15s</td>
</tr>
<tr>
<td>msi N</td>
<td>Yes</td>
<td>0.09s</td>
</tr>
<tr>
<td>moesi N</td>
<td>Yes</td>
<td>0.21s</td>
</tr>
</tbody>
</table>

Acknowledgements

This work is supported by the French ANR project PARDI (DS0703)