SMC: A Symmetry-Based Model Checker for Verification of Safety and Liveness Properties

Sistla A.P., Gyuris V., Emerson A.E.

Presented by: Avik Sinha

Model Checking
- Models: Abstractions representing system properties.
- Model Checking: Verification of the Representative System Properties (mostly automatic)

This Paper
- Model Checker for Concurrent Programs.
- Basics of Concurrent Programs.
- The tool: SMC.
- A Case Study.

Concurrent Programs
- Multiple processes execute in parallel.
- Critical resources shared between processes.
- Synchronizations between the processes

Important terms in Concurrent Programming
- Safety Property: The program never enters a bad state.
- Liveness Property: The program eventually enters a good state.
- No Deadlock Property: The program does not have a state with all blocked processes.
- Mutual Exclusion Property: At most only one process is executing in its critical section.

Fairness
- Weak Fairness:
  - A computation is called weakly fair, if every process that is continuously enabled from any point in time is executed infinitely often.
- Strong Fairness
  - A computation is called strongly fair, if every process that is enabled infinitely often in the computation is executed infinitely often.
SMC: The Features
- Checks correctness under two different notions of fairness:
  - Weak Fairness
  - Strong Fairness
- On-the-Fly model checker
- Works on principles of State Enumeration

State Enumeration
- Construction of global state graph.
- Compute the product of the global state graph and the automaton specifying the incorrect computation.
- Perform a search of the product graph, for strongly connected subgraph satisfying certain properties.

Property Check: Weak Fairness
- A strongly connected subgraph \( C \) is weakly fair if the following condition is satisfied for every process \( p \):
  - there exists a state in \( C \)
  - in which \( p \) is disabled
  - or \( C \) contains an edge caused by the execution of a transition of process \( p \).

Property Check: Strong Fairness
- A strongly connected subgraph \( C \) is strongly fair if the following condition is satisfied for every process \( p \):
  - If \( C \) contains a state in which \( p \) is enabled
  - then it also contains an edge that is caused by the execution of a transition of \( p \).

Property Check: Correctness
- The input program has a weak/strong fair incorrect computation if and only if
  - the product graph contains an accepting and weakly/strongly fair strongly connected subgraph
  - that is reachable from the initial state.

Problem with the Modeling Technique
- Exponential explosion in the size of the global state graph with the number of processes.
- SMC does Symmetry based reduction
- The symmetry existing in the concurrent program induces an equivalence relation on the global states.
- By keeping only a single representative state from each equivalence class the state space is compressed to a smaller structure called Annotated Quotient Structure (AQS).
SMC assumptions

- SMC assumes that the input program has processes that can divided into equivalent classes called modules.
- A state of the program is defined by the values of the program variables.

AQS construction

- SMC supports the following three options for the construction of the AQS:
  - In the first option the entire AQS is constructed in advance.
  - In the second option, the nodes and the edges of AQS are constructed when they are needed for the first time in the product graph construction.
  - Finally, in the most space efficient case, the nodes and edges are constructed, as in the previous case, in an on-the-fly manner, but the edges are not stored. The edges from a given AQS node are constructed as and when needed.

SMC:Input Language

- Input consists of set of commands.
- Each command can be either a
  - Module declaration
  - Variable declaration
  - Module Specification
  - Constant declaration

Example:

```
// declaration of modules
Module server 5 2;
Module client 5 3;

// declaration of program variables
busy[server] 5 0;
request[server, client] 5 0;
reply[server, client] 5 0;
lc[client] 5 0;

// index variable declaration
s of server;
c of client;

// server specification starts
s: busy[s] 5 5 0 && request[s, c] 5 5 0 -. reply[s, c] 5 1, 
   busy[s] 5 1;

// client specification starts

   c: {
      lc[c] 5 5 0 -. lc[c] 5 1, request[s, c] 5 1;
      lc[c] 5 5 1 && reply[s, c] 5 5 1 -. lc[c] 5 2, request[s, c] 5 0;
      lc[c] 5 5 2 && reply[s, c] 5 5 1 -. lc[c] 5 0, busy[s] 5 0, 
      reply[s, c] 5 0;
      lc[c] 5 5 3, 
      ...
   }  
```

Construction of Product Graph

- Automaton specifies all unwanted states and corresponding transitions.
- Product Graph is a the product of the AQS and the automaton.
- The exploration of product graph helps proving various properties

SMC:Symmetry Options

- SMC supports three types of symmetry options.
  - Option (1): No symmetry. This can be used if there is no symmetry in the system or if the number of identical processes is too small.
  - Option (2): Only process symmetry used. AQS is constructed by identifying equivalent states, and no state symmetry is invoked.
  - Option (3): This option employs both process and state symmetries.
Experiments and Case Study

- The link layer part was modeled in detail and the physical layer part was not implemented.
- Two potential problems discovered:
  - A deadlock property.
  - A liveness property.

Results: Deadlock Case

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>AQS states</td>
<td>1952/1483/</td>
<td>84752/14575/</td>
<td>98750/</td>
</tr>
<tr>
<td>Computation Time (sec.)</td>
<td>3/1</td>
<td>6/8/8</td>
<td>6/8/8</td>
</tr>
<tr>
<td>Memory Used (KB)</td>
<td>219/261/</td>
<td>20340/3498/</td>
<td>27271/</td>
</tr>
</tbody>
</table>

Results: No Deadlock Case

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>AQS states</td>
<td>2952/1483/</td>
<td>84752/14575/</td>
<td>98750/</td>
</tr>
<tr>
<td>Computation Time (sec.)</td>
<td>2/1</td>
<td>12/31/31</td>
<td>13/2/3/</td>
</tr>
<tr>
<td>Memory Used (KB)</td>
<td>219/261/</td>
<td>20340/3498/</td>
<td>27271/</td>
</tr>
</tbody>
</table>

Conclusion

- In this article a verification tool SMC is described.
- The main features of SMC are the following.
  - It is a based on state space exploration with symmetry reduction.
  - It utilizes both interstate symmetry (called process symmetry) and intra-state symmetry (called state symmetry) to reduce the explored state space.
  - It can be used to check for correctness under weak and strong fairness.

- The other features of SMC include:
  - SMC is capable of checking safety as well as all liveness properties expressible by finite-state automata.
  - It can also be used just for detecting deadlocks by not specifying the automaton component of the input.
  - It is an on-the-fly model checker which allows early termination. It constructs the state space at the same time as it builds and explores the product graph.