Applications and Limitations of the SPIN model checker

compared to TLC and CBMC

K.P. Radziwiłowicz[1] | Supervisors: B.P. Ahrens[1], A. Lukina[1]

[1] Faculty of Electrical Engineering, Mathematics and Computer Science, TU Delft

CSE3000 Research Project

Abstract

In this project, we explore the applications and limitations of the SPIN model checker as compared to TLC and CBMC checkers. We explore the checkers' expressivity and run synthetic benchmarks to compare their performance.

1 Background

1.1 Model Checking

- Formal verification method that uses a (simplified) mathematical model of a system
- Properties are commonly expressed using a temporal logic, e.g., LTL
- Example: lift moving between the ground floor and the first floor; the states are: *G* - on the ground floor; *U* - moving up; *F* - on the first floor; *D* - moving down

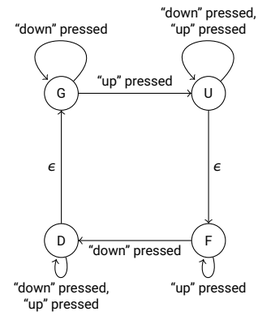

Figure: State machine modelling the lift example.

1.2 Linear Temporal Logic (LTL)

- Superset of propositional logic
- Can express statements referring to discretized time
  - Achieved through so-called "modal temporal operators"; most important ones (for us) are:
    - $\Box\varphi$ (*globally* φ)
      φ must hold at all future states
    - $\Diamond\varphi$ (*finally* φ)
      φ must hold at some future state
- Example property: $\Box\,(U \rightarrow \Diamond F)$
  - *It's always the case that if the lift is moving up, it will eventually reach the first floor*

2 SPIN

- A model checker developed in the 1980s by G.J. Holzmann
- Primarily used to verify concurrent software
- Uses an imperative modelling language PROMELA
- Properties to verify can be expressed as assertions or LTL formulae

3 Research Questions

a. How do SPIN's capabilities in terms of expressible and verifiable models and properties compare to the capabilities of TLC and CBMC?

b. How do SPIN's performance and memory usage when checking safety and liveness properties compare to the performance of TLC and CBMC?

4 SPIN vs TLC

- Both are explicit-state (i.e., visit all reachable states to verify a model)
- TLC is better at describing high-level systems
  - uses a declarative language TLA$^+$
  - supports a wider range of data types (functions, sets, multisets, . . .)
- Both support different optimisation techniques
  - SPIN supports partial order reduction (POR)
  - TLC supports symmetry reduction
    - This technique is only applicable if separate processes behave exactly in the same manner

```
proctype P() {
    a = 21;
    ...
}
```

```
proctype Q() {
    b = 37;
    ...
}
```

```
proctype P() {
    a = 21;
    ...
}
```

```
proctype Q() {
    b = 37;
    ...
}
```

Figure: Example of interleavings equivalent under partial order reduction; the first instructions of processes *P* and *Q* are independent

```
pc = [
   p1 -> "Enter",
   p2 -> "Critical",
   p3 -> "Exit"
]
```

```
pc = [
   p1 -> "Critical",
   p2 -> "Exit",
   p3 -> "Enter"
]
```

Figure: Example of process state mappings that can be equivalent under symmetry reduction

5 SPIN vs CBMC

- CBMC operates directly on C code with assertions
  - As a consequence, it doesn't support verifying liveness properties (of the form $\Box\,(p \implies \Diamond q)$; i.e., "always true that if *p*, then eventually *q*")
  - A similar effect in SPIN can be achieved by automatically extracting a model using Modex
- CBMC is a symbolic model checker - instead of traversing the state graph, it expresses the system as logic formulae and tries to satisfy them with a SAT solver
- CBMC isn't well-optimised for verifying concurrent programs
  - This limitation can be mitigated with external tools like CSeq
- CBMC automatically checks for more safety errors (e.g., integer overflows, out-of-bound accesses, double free)

6 Selected Experimental Results

Table: Performance of TLC model and full SPIN model on the filter algorithm (safety)

Model Checker	Configuration	Time Taken (s)	Memory Used (MiB)	Unique States Visited
SPIN	N, E, D	42.38	3490.24	51935282
	P, E, D	8.45	1381.88	18790666
TLC	N	11.60	955.53	315222
	S	2.00	261.57	2809

Table: Performance of TLC with symmetry reduction on the R/W lock server

Model Checker	Configuration	Time Taken (s)	Memory Used (MiB)	Unique States Visited
TLC	N	4.40	783.63	222417
	S	12.00	330.71	333

Table: Performance of CBMC and SPIN when model checking binary search algorithm

Model Checker	Configuration	Time Taken (s)	Memory Used (MiB)	Unique States Visited
SPIN	N, E, D	21.86	5278.14	54334024
CBMC	-	0.04	17.04	-

Table: Performance of CBMC and SPIN when model checking filter algorithm

Model Checker	Configuration	Time Taken (s)	Memory Used (MiB)	Unique States Visited
SPIN	N, E, D	0.02	133.86	47578
CBMC	-	43.72	90.24	-

- SPIN configurations: N/P - POR disabled/enabled; E - exhaustive state storage mode; D - DFS used as the search algorithm
- TLC configurations: N/S - symmetry reduction enabled/disabled

7 Conclusions

- SPIN is better suited for model checking low-level models, while TLC is better suited for model checking high-level models
- SPIN with POR performs better that TLC despite visiting more unique states in its models
  - With symmetry reduction, TLC can outperform SPIN
- CBMC performs better when verifying sequential programs with many possible values for a static number of variables; SPIN performs better when verifying concurrent programs with a large number of possible interleavings

8 Future Work

- Compare more model checkers, e.g., nuXmv, UPPAAL, Murphi
- Benchmark using multithreading for model checking
- Improve benchmarks
  - Introduce new test cases, e.g., models of high-level systems or protocols
    - E.g., consensus protocols like Paxos or Raft
  - Test the performance of CBMC on code sequentialised by e.g. CSeq

TU Delft Delft University of Technology