Perfomance Evaluation of Specification Types in NuSMV

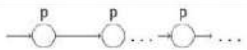
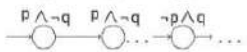
AG (CTL = theBest)?

1. Background

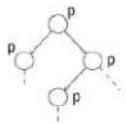
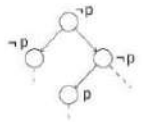
Model checkers - tools for formal verification that a system is correct (checks for safety, deadlock, invariants, etc.).

NuSMV - symbolic model checker for verifying finite-state systems against temporal logic specifications; supports LTL, CTL and PSL.

LTL (Linear Temporal Logic) - reasons over single execution paths

Temporal connectives	Formula	Trace
G - globally	Gp	
F - eventually	Fp	
X - next	Xp	
U - until	pUq	

CTL (Computation Tree Logic) - reasons over all or some execution paths

Temporal connectives	Formula	Trace
A - on all paths	AGp	
E - there exists a path	EFp	

PSL (Property Specification Language) - extension of LTL

2. Research Question

As systems scale, which **specification type** performs **best** with respect to **runtime** and **memory** consumption in **NuSMV** (2.7.1)?

Hypothesis: PSL and LTL specifications perform similarly to one another, but they will experience a steeper increase in runtime and peak memory compared to equivalent CTL specifications.

3. Methodology

- Three **use cases** were analysed: a cache coherence protocol, a pipeline scheduling problem, a master bus arbitration protocol; we scaled them via custom python scripts.
- We wrote **properties** equivalent in all specification types.
- We used two **model checking options:**
  Binary Decision Diagrams (BDDs) → graphical representation of boolean formulas;
  Bounded Model Checking (BMC) → translates model into SAT formulas, then checks the truth value up to bound k; available only for LTL and PSL;
- **Metrics**: runtime, peak memory, and the size of the underlying data structures.

4. Results and Conclusions

- Above the tables we present the properties checked by the model

Non-starvation: AG (P_N_1.request -> AF (P_N_3.finish & AF (P_N_3. state = 0))):

Scale (n)	SpecType	Runtime (s)	Peak memory (MB)	BDD size (nodes)
Pipelines-10	CTL	0.27	15.17	198 268
	LTL	0.56	30.63	421 064
	PSL	0.56	30.63	421 064
Pipelines-15	CTL	1.55	20.33	348 502
	LTL	2.18	35.57	811 468
	PSL	2.25	35.57	811 468
Pipelines-18	CTL	**14.00**	52.86	1 080 254
	LTL	**3.63**	44.66	1 054 704
	PSL	**3.74**	44.66	1 054 704

Mutual exclusion: AG !(pi.writable & (p(i+1).writable | p(i+2). writable | ... | p(N-1).writable))

Scale (n)	SpecType	Runtime (s)	Peak Memory (MB)	BDD size (nodes)
Cache-10	CTL	0.32	42.14	764 456
	LTL	0.48	48.36	954 548
	PSL	0.50	48.36	954 548
Cache-15	CTL	1.17	58.00	1 206 982
	LTL	2.68	57.72	1 223 334
	PSL	2.70	57.72	1 223 334
Cache-20	CTL	**4.38**	**107.73**	**2 717 498**
	LTL	**13.27**	**80.28**	**1 891 722**
	PSL	**15.63**	**80.28**	**1 891 722**

Scale (n)	SpecType	Runtime (s)	Peak Memory (MB)	SAT Clauses
Cache-10	LTL	10.07	32.94	11 020
	PSL	4.24	34.21	11 020
Cache-15	LTL	**27.07**	**40.92**	**16 530**
	PSL	**14.34**	**43.71**	**16 530**
Cache-20	LTL	**87.05**	**53.20**	**22 240**
	PSL	**25.99**	**57.39**	**22 240**

Priority Precedence: AG(master(N-1).req -> A[master(N-1).req U (grant=(N -1) | master(N-2).req | ... | master0.req)])

Scale (n)	SpecType	Runtime (s)	Peak Memory (MB)	BDD size (nodes)
Masters-6	CTL	0.14	17.35	264 698
	LTL	0.80	11.59	89 936
	PSL	0.82	11.59	89 936
Masters-8	CTL	**0.38**	**31.09**	436 394
	LTL	**14.47**	**57.71**	1 238 664
	PSL	**14.34**	**57.71**	1 238 664
Masters-10	CTL	**1.41**	**58.09**	**1 249 906**
	LTL	**184.03**	**92.42**	**2 231 026**
	PSL	**224.56**	**92.39**	**2 232 048**

Scale (n)	SpecType	Runtime (s)	Peak Memory (MB)	SAT Clauses
Masters-6	LTL	0.09	20.30	1
	PSL	0.06	19.34	1
Masters-8	LTL	**9.87**	**48.97**	15 166
	PSL	**4.95**	**50.72**	15 134
Masters-10	LTL	**11.82**	**55.86**	**19 072**
	PSL	**5.72**	**55.72**	**19 076**

Conclusions

- Mostly CTL outperforms in both time and memory - on simple properties that just had a global ("G") quantifier or on properties with "X" and "U" operators, as they seem to add overhead for LTL/PSL;
- LTL/PSL outperform on nested "F" operators;
- Generally, LTL and PSL have the same performance on BDDs.
- PSL tends to be faster within BMC, but memory and SAT size stay alike.
- BMC also is generally faster than BDD; although depth k of the unrolling affects the trust about the correctness of the model.

Limitations

- Models not of industrial scale.
- Choices made when scaling the model could give biased results.
- Properties written might not encompass all possible behaviours.

5. Future work

- Reproduce on industrial scale systems.
- Disclose more details on how NuSMV is used, in papers which integrate the tool to test their systems.
- Analyze specification types across same classes of properties (fairness, safety, etc).

Responsible Professor: Benedikt Ahrens
Supervisor: Anna Lukina

Author: Oana-Teodora Mirea
o.t.mirea@student.tudelft.nl

TU Delft