



# Advances in Space and Time Efficient Model Checking of Finite State Systems

Atanas Nikolaev Parashkevov

Thesis submitted for the degree of  
Doctor of Philosophy  
in  
The University of Adelaide  
(Faculty of Engineering, Computer and Mathematical Sciences)

November 2002

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Formal verification in context . . . . .	1
1.2	Verification of concurrent systems . . . . .	4
1.3	State space and model checking complexity . . . . .	5
1.4	Research motivation and objectives . . . . .	8
1.5	Achievements and contribution . . . . .	9
<b>2</b>	<b>Background</b>	<b>13</b>
2.1	The CSP framework . . . . .	13
2.1.1	Overview . . . . .	13
2.1.2	Syntax and operators . . . . .	15
2.1.3	Semantic models . . . . .	18
2.1.4	Refinement and equivalence relations . . . . .	21
2.1.5	Refinement checking . . . . .	22
2.2	Labelled Transition Systems (LTS) . . . . .	23
2.2.1	Reachability analysis . . . . .	25
2.3	Ordered Binary Decision Diagrams (OBDD) . . . . .	27
2.3.1	Representing sets as OBDDs . . . . .	29
2.3.2	Representing LTSs as OBDDs . . . . .	30
2.3.3	Traversing an OBDD-based transition relation . . . . .	31
2.4	Tools for automated verification . . . . .	33
2.4.1	FDR . . . . .	33
2.4.2	MRC . . . . .	35
2.4.3	SMV . . . . .	35
2.4.4	SPIN . . . . .	36
2.4.5	Others . . . . .	36
<b>3</b>	<b>Basic OBDD-based Refinement Checking</b>	<b>37</b>
3.1	Related work . . . . .	37
3.2	Compiling processes into OBDDs . . . . .	39

3.2.1	Identifiers and expressions . . . . .	41
3.2.2	Process references and recursion . . . . .	43
3.2.3	OBDD encodings of states and events . . . . .	47
3.2.4	Computing CSP semantics on LTS representations . . . . .	50
3.2.5	Syntax-driven transformation rules . . . . .	53
3.2.6	Re-encoding of sequential components . . . . .	64
3.3	Refinement checking . . . . .	66
3.3.1	Computing refusals and divergences . . . . .	66
3.3.2	Pair-by-pair refinement checking algorithm . . . . .	69
3.3.3	Experimental results . . . . .	71
3.4	Discussion . . . . .	80
3.4.1	OBDD-based versus explicit refinement checking . . . . .	80
3.4.2	Bottleneck analysis . . . . .	83
<b>4</b>	<b>Hierarchical Partitioned Transition Relations</b>	<b>85</b>
4.1	Motivation . . . . .	85
4.2	Partitioned transition relations . . . . .	87
4.3	Hierarchical partitioned transition relations (HPTR) . . . . .	89
4.3.1	Definition . . . . .	89
4.3.2	Structural optimisation . . . . .	91
4.4	Recursive image computation on HPTRs . . . . .	92
4.4.1	Leaf nodes . . . . .	93
4.4.2	Hiding nodes . . . . .	94
4.4.3	Parallel and interleave nodes . . . . .	94
4.5	Experimental results . . . . .	96
4.6	A further enhancement . . . . .	98
4.7	Related work . . . . .	99
<b>5</b>	<b>Pseudo-Root States</b>	<b>101</b>
5.1	Existing reachability analysis methods . . . . .	101
5.2	Background and motivation . . . . .	104
5.2.1	Conventional reachability analysis . . . . .	104
5.2.2	Interleaved processes example . . . . .	105
5.2.3	Counting example . . . . .	106
5.3	PRS-based reachability analysis . . . . .	107
5.3.1	Pseudo-root states . . . . .	107
5.3.2	PRS-enhanced reachability analysis algorithm . . . . .	108
5.3.3	Reaching physical memory limits . . . . .	111
5.3.4	Complexity analysis . . . . .	112
5.4	Experimental results . . . . .	113

5.5	Discussion . . . . .	115
5.5.1	Computing predecessor states . . . . .	115
5.5.2	BFS, DFS and alternatives . . . . .	116
5.5.3	Comparison to previous work . . . . .	117
<b>6</b>	<b>Exploiting Partial Orders</b>	<b>119</b>
6.1	Motivation . . . . .	119
6.2	Partial order methods . . . . .	120
6.3	Sleep sets method . . . . .	123
6.4	Combining PRS and sleep sets . . . . .	126
6.5	Computing the dependence relation . . . . .	130
6.6	Experimental results . . . . .	131
6.6.1	Performance of ARC/PP+SS . . . . .	131
6.6.2	Performance of ARC/PP+PRS+SS . . . . .	132
<b>7</b>	<b>The ARC Tool</b>	<b>135</b>
7.1	Introduction . . . . .	135
7.2	Architecture . . . . .	136
7.3	Input language . . . . .	138
7.4	Command-line options . . . . .	139
7.4.1	Verification mode options . . . . .	140
7.4.2	Compilation options . . . . .	141
7.4.3	OBDD related options . . . . .	141
7.4.4	Verification algorithms options . . . . .	142
7.4.5	General options . . . . .	143
7.5	Example run sessions . . . . .	143
<b>8</b>	<b>Extensions and Future Work</b>	<b>149</b>
8.1	Introduction . . . . .	149
8.2	Reducing refinement checking to a reachability problem . . . . .	150
8.2.1	Pair-by-pair refinement checking . . . . .	150
8.2.2	Full abstraction in CSP . . . . .	151
8.2.3	Refinement checking with observers . . . . .	152
8.2.4	Transforming specifications into observers . . . . .	154
8.2.5	OBDD-based reachability analysis . . . . .	156
8.2.6	Related work . . . . .	157
8.3	Optimised pair-by-pair refinement checking . . . . .	158
8.4	Other partial order methods . . . . .	160
8.5	OBDD-based state compression . . . . .	160
8.6	SAT-based verification . . . . .	161

8.6.1	Bounded model <b>checking</b> (BMC)	162
8.6.2	Applying <b>induction</b> and hybrid methods	163
8.7	When everything else fails: semi-formal verification	164
8.7.1	A symbolic <b>semi-formal</b> verification method	167
8.8	The need for multiple verification engines	169
8.9	Acknowledgments	170
<b>9</b>	<b>Conclusions</b>	<b>173</b>
9.1	Summary	173
9.2	Research arisen from our work	176
<b>A</b>	<b>CSP Examples</b>	<b>177</b>
A.1	Synthetic example	177
A.2	Milner's Schedulers example	178
A.2.1	Variant one	179
A.2.2	Variant two	179
A.3	Dining Philosophers example	179
A.4	Alternating Bit Protocol example	179
A.5	Monkey Puzzle example	188
A.6	Solitaire example	201

# Abstract

The complexity of today's computer and electronic systems has reached a point where traditional approaches for design verification—simulation, testing, and reviews—are no longer sufficient. The cost of design defects slipping into production has also sharply increased.

Formal verification holds the promise of a more robust, complete and hopefully automated approach to ensuring that these systems implement their desired behaviour. However, a phenomenon known as state space explosion limits the size of systems that can be handled successfully. This thesis studies automated formal verification techniques and their associated space and time implementation complexity when applied to finite state concurrent systems. In particular, the focus is on concurrent systems expressed in the Communicating Sequential Processes (CSP) framework. The large body of work on CSP includes a process description language, a set of semantic models—traces, failures, and failures-divergences, as well as formal notions of process refinement and equivalence.

An approach to the compilation of CSP system descriptions into boolean formulae in the form of Ordered Binary Decision Diagrams (OBDD) is presented. This representation of CSP semantics is further utilised by a basic algorithm that checks a refinement or equivalence relation between a pair of processes in any of the three CSP semantic models. The performance of this algorithm is studied on a set of benchmark examples and a comparative analysis of its strengths and weaknesses in relation to explicit on-the-fly model checking is performed. This analysis identifies the major factors affecting the performance of OBDD-based refinement checking.

The performance bottlenecks of the basic refinement checking algorithms are identified and addressed with the introduction of a number of novel techniques and algorithms. Hierarchical partitioned transition relations provide an alternative to monolithic transition relations which may grow too large for certain problems. A pseudo-root state approach is developed to reduce the number of states that need to be stored during reachability analysis. A partial order technique—the sleep set method—is applied to OBDD-based refinement checking, which significantly reduces the verification run-time. The pseudo-root state

and sleep set methods are synergistically combined to obtain lower space and time requirements for reachability analysis than either of these techniques alone can provide. We utilise the full abstraction property of the CSP semantics to reduce a CSP refinement checking problem into a reachability problem that can be solved efficiently using OBDDs and enables the application of SAT-based and semi-formal verification techniques.

The algorithms described in this thesis are implemented in the Adelaide Refinement Checking tool.