Concurrent Algorithms and Data Structures for Model Checking

Jaco van de Pol
Aarhus University, Denmark
University of Twente, The Netherlands
http://www.cs.au.dk/~jaco
jaco@cs.au.dk

Abstract

Model checking is a successful method for checking properties on the state space of concurrent, reactive systems. Since it is based on exhaustive search, scaling this method to industrial systems has been a challenge since its conception. Research has focused on clever data structures and algorithms, to reduce the size of the state space or its representation; smart search heuristics, to reveal potential bugs and counterexamples early; and high-performance computing, to deploy the brute force processing power of clusters of compute-servers. The main challenge is to combine these approaches – brute-force alone (when implemented carefully) can bring a linear speedup in the number of processors. This is great, since it reduces model-checking times from days to minutes. On the other hand, proper algorithms and data structures can lead to exponential gains. Therefore, the parallelization bonus is only real if we manage to speedup clever algorithms.

There are some obstacles though: many linear-time graph algorithms depend on a depth-first exploration order, which is hard to parallelize. Examples include the detection of strongly connected components (SCC) and the nested depth-first-search (NDFS) algorithm. Both are used in model checking LTL properties. Symbolic representations, like binary decision diagrams (BDDs), reduce model checking to “pointer-chasing”, leading to irregular memory-access patterns. This poses severe challenges on achieving actual speedup in (clusters of) modern multi-core computer architectures.

This talk presents some of the solutions found over the last 10 years, which led to the high-performance model checker LTSmin [2]. These include parallel NDFS (based on the PhD thesis of Alfons Laarman [3]), the parallel detection of SCCs with concurrent Union-Find (based on the PhD thesis of Vincent Bloemen [1]), and concurrent BDDs (based on the PhD thesis of Tom van Dijk [4]).

Finally, I will sketch a perspective on moving forward from high-performance model checking to high-performance synthesis algorithms. Examples include parameter synthesis for stochastic and timed systems, and strategy synthesis for (stochastic and timed) games.

2012 ACM Subject Classification Software and its engineering → Model checking; Theory of computation → Verification by model checking; Theory of computation → Parallel algorithms

Keywords and phrases model checking, parallel algorithms, concurrent datastructures

References