Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 128 (2005) 107–123
www.elsevier.com/locate/entcs

CTL* Model Checking on a Shared-Memory Architecture

Cornelia P. Inggs[1,2] and Howard Barringer[3]

Department of Computer Science
University of Manchester
UK

Abstract

In this paper we present a parallel algorithm for CTL* model checking on a virtual shared-memory high-performance parallel machine architecture. The algorithm is automata-driven and follows a games approach where one player wins if the automaton is empty while the other player wins if the automaton is nonempty. We show how this game can be played in parallel using a dynamic load balancing technique to divide the work across the processors. The practicality and effective speedup of the algorithm is illustrated by performance graphs.

Keywords: Model Checking, Shared-Memory, Parallelisation, Automata, Game Theory

1 Introduction

Model checking is an established technology for automated verification of designs, now adopted by industry to check correctness properties of many critical systems. The tremendous advances that have been made over the past decade in developing specialised state encodings and algorithms to reduce the burden of the state explosion problem, inherent with this style of verification, have been paramount to this industrial take-up. Even so, the size and complexity of

[1] This work was fully supported under a Universities UK ORS award, a University of Manchester Department of Computer Science Scholarship, and a South African Harry Crossley Bursary.
[2] inggscp@telkom.co.za
[3] howard@cs.man.ac.uk

doi:10.1016/j.entcs.2004.10.022

systems that can be verified is still heavily constrained by time and available memory, and the development of techniques to alleviate the state explosion problem remains an active area of research. One technique that has gained significant interest recently is the parallelisation of model checking.

There were a few isolated publications on the parallelisation of model checkers in the 1980s and 1990s and then Stern and Dill's seminal paper for parallel reachability analysis appeared in 1997 [17]. In the past three to four years parallel model checking has gained considerable interest.

Much of the extant research has focused on implementations over distributed networks and the development of static partioning functions. Static partitioning functions depend on the state and not on the distribution of the workload. To the best of our knowledge the only algorithms that use a dynamic partitioning function are the symbolic algorithm of Heyman et al. [12] and the two algorithms based on Heyman et al.'s article [3,11]. In these algorithms the memory balance is maintained by repartitioning the state space whenever the memory becomes unbalanced. Initially only safety checking was parallelised, but in the last few years the developement of parallel algorithms for liveness checking increased and algorithms for checking LTL [2,6,16,15], CTL [7], and the μ-calculus [5] have been developed. See [13] for a full bibliography. The development of efficient parallel algorithms for liveness checking have been less successful than for safety checking and very few parallel algorithms for checking both liveness and safety properties achieve speedups.

We explored the parallelisation of model checking for shared-memory multiprocessor computers to evaluate its feasibility and identify any inherent difficulties or pitfalls when parallelising model checking for shared memory architectures. In particular, the parallelisation of explicit-state on-the-fly model checking was investigated for both safety and liveness properties and led to the development of a parallel model checker for CTL*. This research has shown the practicality and effective speedup of model checking using a shared-memory architecture. The performance of the parallel algorithm was evaluated via theoretical analysis and also via experimental analysis using a number of prototypical models, including correctness properties of the parallel model checker itself.

In our earlier paper, [14], we proposed a parallel algorithm for reachability analaysis on a shared-memory architecture. In this paper we present a parallel model checking algorithm for CTL* that uses the dynamic load balancing technique of the parallel reachability analysis algorithm described in [14]. An overview of the parallel reachability analysis algorithm is given in the next section. This is followed by a description of the serial automata-driven game-theoretic algorithm for CTL* and its parallelisation in Sections 3 and 4.