Information and Computation 209 (2011) 799–821

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

A saturation method for the modal μ-calculus over pushdown systems

M. Hague *, C.-H.L. Ong

Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK

ARTICLE INFO

Article history:
Available online 15 December 2010

Keywords:
Modal μ-calculus
Pushdown systems
Parity games
Winning regions
Global model checking
Saturation methods

ABSTRACT

We present an algorithm for computing *directly* the denotation of a modal μ-calculus formula χ over the configuration graph of a pushdown system. Our method gives the first extension of the saturation technique to the full modal μ-calculus. Finite word automata are used to represent sets of pushdown configurations. Starting from an initial automaton, we perform a series of automaton manipulations which compute the denotation by recursion over the structure of the formula. We introduce notions of under-approximation (soundness) and over-approximation (completeness) that apply to automaton transitions rather than runs. Our algorithm is relatively simple and direct, and avoids an immediate exponential blow up. Finally, we show experimentally that the direct algorithm is more efficient than via a reduction to parity games.

1. Introduction

Pushdown systems – finite-state transition systems equipped with a stack – are an old model of computation that have recently enjoyed renewed interest from the software verification community. They accurately model the control flow of first-order recursive programs [18] (such as C and Java), and lend themselves readily to algorithmic analysis. Pushdown systems have played a key rôle in the automata-theoretic approach to software model checking [6,13,20,21]. Considerable progress has been made in the implementation of scalable model checkers of pushdown systems. These tools (e.g., Bebop [3] and Moped [21]) are an essential back-end component of such model checkers as SLAM [4].

The modal μ-calculus is a highly expressive language for describing properties of program behaviour (all standard temporal logics in verification are embeddable in it). In a seminal paper [23] at CAV 1996, Walukiewicz showed that *local* modal μ-calculus model checking of pushdown systems – or equivalently [12] the solution of *pushdown parity games* (i.e., parity games over the configuration graphs of pushdown systems) – is EXPTIME-complete. His method reduces pushdown parity games to finite parity games by a kind of powerset construction, which is immediately exponential in size.

Whilst *local* model checking asks if a designated state (of a pushdown system) satisfies a given property, *global* model checking computes a finite representation of the set of states satisfying the property. It is worth noting that global model checking used to be the norm in verification (CTL and many symbolic model checkers still perform global model checking). While local model checking can be expected to have better complexity, global model checking is important when repeated checks are required (because tests on the representing automata tend to be comparatively cheap), or where the model checking is only a component of the verification process.

1.1. Contributions

This paper presents a new algorithm for solving the global model checking problem for modal μ-calculus over pushdown systems. That is, given a pushdown system $\mathbb{P}$, a modal μ-calculus formula $\chi(\overline{Z})$ for $\overline{Z} = Z_1, \ldots, Z_n$, and a regular valuation V,

* Corresponding author. Fax: +44 1865 273839.
E-mail addresses: Matthew.Hague@comlab.ox.ac.uk (M. Hague), Luke.Ong@comlab.ox.ac.uk (C.-H.L. Ong).

0890-5401/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2010.12.004

our method can *directly* compute an automaton that recognises the set $[\![\chi(\overline{Z})]\!]^{\mathbb{P}}_V$ of $\mathbb{P}$-configurations satisfying $\chi(\overline{Z})$ with respect to V.

We represent the (regular) configuration sets as alternating multi-automata [6]. To evaluate a fixed point formula, our algorithm iteratively expands (when computing least fixed points) or contracts (when computing greatest fixed points) an approximating automaton until the denotation is precisely recognised. Our method is a generalisation of Cachat's for solving Büchi games [10,11], which is itself a generalisation of the saturation technique for reachability analysis [6]. A specialised version of this algorithm was presented in Concur 2009 [16]. This simplified algorithm computes, using a modal μ-calculus formula as a guide, the winning regions of a pushdown parity game.

Our algorithm has several advantages:

1. The algorithm is relatively simple and direct. Even though pushdown graphs are in general infinite, our construction of the automaton that recognises the denotation follows, in outline, the standard pen-and-paper calculation of the semantics of modal μ-calculus formulas in a *finite* transition system. Through the use of *projection*, our algorithm is guaranteed to terminate in a finite number of steps, even though the usual fixed point calculations may require transfinite iterations. Thanks to projection, the state-sets of the approximating automata are bounded: during expansion, the number of transitions increases, but only up to the bound determined by the finite state-set; during contraction, the number of transitions decreases until it reaches zero or stabilizes.
2. Conceptual innovations of the correctness argument are *valuation soundness* and *valuation completeness*. They are, respectively, under- and over-approximation conditions that apply *locally* to individual transitions of the automaton, rather than *globally* to the extensional behaviour of the automaton (such as runs). By combining these conditions, which reduce the overhead of the proof,[1] we show that our algorithm is both sound and complete in the usual sense.
3. The algorithm, in essence, combines the product construction – that reduces a modal μ-calculus model checking problem to a pushdown parity game – and the computation of the winning region. However, this direct computation only introduces product states that are relevant to the evaluation of the current *sub-formula* (rather than the whole formula), hence the number of states used is minimised. Since the algorithm is exponential in the number of states, even a slight reduction in the number of states can lead to significant improvements in run-times. We confirm this experimentally in Section 9.
4. Finally, our decision procedure builds on and extends the well-known saturation method, which is the implementation technique of choice of pushdown checkers. In contrast to previous solutions, our algorithm avoids an immediate exponential explosion, which we believe is important for an efficient implementation.

1.2. Related work

Cachat [11] and Serre [22] have independently generalised Walukiewicz' algorithm to provide solutions to the global model-checking problem: they use the local model-checking algorithm as an oracle to guide the construction of the automaton recognising the winning region. An alternative approach, introduced by Piterman and Vardi [19], uses two-way alternating tree automata to navigate a tree representing all possible stacks: after several reductions, including the complementation of Büchi automata, an automaton accepting the winning regions can be constructed.

An early technique for analysing modal μ-calculus properties of pushdown systems is due to Burkart and Steffen [9]. They provide an algorithm for analysing context-free systems by reduction to a finite equational fixed point computation. This can be extended to pushdown systems by adding arguments to the equations, and then performing a computation argument-wise for each of the exponential number of arguments [8].

At Concur 1997, Bouajjani et al. [6], and, independently, Finkel et al. [15] (at Infinity 1997), introduced a *saturation* technique for global model-checking reachability properties of pushdown systems. This technique was based on a string-rewriting algorithm due to Book and Otto [5]. From a finite-word automaton recognising a given configuration-set $\mathcal{C}$, they perform a backwards-reachability analysis. By iteratively adding new transitions to the automaton, the set of configurations that can reach some configuration in $\mathcal{C}$ is constructed. Since the number of new transitions is bounded, the iterative process terminates. This approach underpins the acclaimed Moped tool.

The saturation technique was generalised by Cachat to compute the winning regions of Büchi games [10]. By using *projections*, Cachat was able to show how to compute a single alternation of fixed points. We have generalised this approach to compute an arbitrary number of fixed points. Furthermore, we believe that the introduction of valuation-soundness and -completeness leads to a cleaner proof of correctness. An "automaton free" version of Cachat's approach was given by Etessami [14]. This approach computes the winning regions of a Büchi game using data flow equations. To our knowledge, it has not been applied to parity games, although such an extension may be possible.

Finally, Alur et al. introduce a version of the modal μ-calculus for recursive programs [1]. This logic is more expressive than the modal μ-calculus. A investigation of further properties, such as succinctness, is an interesting avenue of future work.

[1] Although some proofs are long, this is primarily due to the number of cases involved in an induction over the syntax of the modal μ-calculus. The proofs themselves are fairly straightforward.