Science of Computer Programming 77 (2012) 1178–1197

Contents lists available at SciVerse ScienceDirect

Science of Computer Programming

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

SAT-solving in CSP trace refinement

Hristina Palikareva *, Joël Ouaknine, A.W. Roscoe

Department of Computer Science, Oxford University, UK

ARTICLE INFO

Article history:
Available online 20 July 2011

Keywords:
CSP
FDR
Concurrency
Process algebra
SAT-solving
Bounded model checking
k-induction
Safety properties

ABSTRACT

In this paper, we address the problem of applying SAT-based bounded model checking (BMC) and temporal *k*-induction to asynchronous concurrent systems. We investigate refinement checking in the process-algebraic setting of Communicating Sequential Processes (CSP), focusing on the CSP traces model which is sufficient for verifying safety properties. We adapt the BMC framework to the context of CSP and the existing refinement checker FDR yielding bounded refinement checking which also lays the foundation for tailoring the *k*-induction technique. As refinement checking reduces to checking for reverse containment of possible behaviours, we exploit the SAT-solver to decide bounded language inclusion as opposed to bounded reachability of error states, as in most existing model checkers. Due to the harder problem to decide and the presence of invisible silent actions in process algebras, the original syntactic translation of BMC to SAT cannot be applied directly and we adopt a semantic translation algorithm based on watchdog transformations. We propose a Boolean encoding of CSP processes resting on FDR's hybrid two-level approach for calculating the operational semantics using supercombinators. We have implemented a prototype tool, SymFDR, written in C++, which uses FDR as a shared library for manipulating CSP processes and the state-of-the-art incremental SAT-solver MiniSAT 2.0. Experiments with BMC indicate that in some cases, especially in complex combinatorial problems, SymFDR significantly outperforms FDR and even copes with problems that are beyond FDR's capabilities. SymFDR in *k*-induction mode works reasonably well for small test cases, but is inefficient for larger ones as the threshold becomes too large, due to concurrency.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Model checking [10,5,3] is a powerful automatic formal verification technique for establishing correctness of systems. It requires a finite-state model of a system, capturing all its possible behaviours, and a specification property, usually modelled as a formula in some kind of temporal logic. The model checker performs analysis based on exhaustive exploration of the state space of the system to either confirm or refute that the system meets its specification. In the latter case, the model checker provides a counterexample trace for reproducing and fixing the bug. Model checking is complete and, therefore, reliable when pronouncing a system correct.

The main challenge in applying model checking in practice is the so-called *state-space explosion problem* which tends to be even more severe in the context of concurrent systems. The state-space of a concurrent system grows exponentially with the number of its concurrent components and the number and size of its data values. This puts restrictions on the size of systems that can be tractably analysed.

* Corresponding author.
E-mail address: hristina.palikareva@cs.ox.ac.uk (H. Palikareva).

0167-6423/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2011.07.008

To alleviate the state-space explosion problem, a significant number of techniques have been proposed. Methods for decreasing the size of the generated state space and enhancing the model checking algorithm include CEGAR [9], partial-order reductions [10,26], bounded model checking [1], etc. Regarding state-space representation, the major dichotomy is between explicit and symbolic [2,1] model checking. *Explicit model checking* is based on explicit enumeration and examination of individual states. *Symbolic model checking* relies on abstract representation of sets of states, generally as Boolean formulae, and properties are validated using techniques such as BDD manipulation or SAT-solving.

The recent advances of efficient SAT-solvers have significantly broadened the horizons of symbolic model checking. SAT-based bounded model checking [1] has proven to be an extremely powerful technique, mainly suited, due to its incompleteness, to falsification of properties. Approaches for making BMC complete include calculating completeness thresholds [11,12] or augmenting BMC with k-induction [38,16] or Craig interpolation [25] techniques.

Both bounded and unbounded SAT-based model checking have been mainly investigated in the context of hardware and sequential software systems. In this paper, we address applying BMC and temporal k-induction [16] to asynchronous concurrent systems.

The general problem we investigate is *refinement checking* in process-algebraic settings and, more specifically, in the context of CSP [19,31,33].

In process algebras, systems are modelled as interactions of a collection of processes, communicating with each other and with the outer world via message passing, as opposed to shared variables. Using a high-level language, processes are defined compositionally and compiled into a hierarchical structure, starting with atomic process constructs and combining those using operators such as choice, parallel and sequential composition, hiding, etc. This allows for a way of describing reactive systems that is usually very concise and much more economical in state space than shared-variable languages.

Unlike in conventional model checking, where specifications are generally defined as temporal-logic formulae, in process algebras specifications are defined as abstract designs of the systems, i.e. as processes, which allows for a step-wise development process. The refinement checking procedure decides whether the behaviours of the system are a subset of the behaviours of the specification, i.e. whether the system refines the specification. Hence, the verification problem reduces to checking for reverse containment of behaviours and, therefore, to reverse language inclusion.

Developed in the late 1970's by Hoare, CSP is one of the two original process algebras. It allows for the precise description and analysis of event-based concurrency. An advantage of the CSP framework is that it offers a well-developed syntax, algebraic and operational semantics, a hierarchy of congruent denotational semantic models, as well as a formal theory of refinement and compositional verification. In terms of syntax and semantics, among other differences with existing formalisms for modelling concurrent systems, CSP supports the usage of broadcast communication, recursion, as well as hiding and renaming of events, both of which are powerful mechanisms for abstraction.

FDR [30,17] is acknowledged as the primary tool for CSP refinement checking and has been widely used for analysing safety-critical systems. The core of FDR is refinement checking in each of the semantic models, which is carried out on the level of the operational representation of the CSP processes and is implemented using explicit state enumeration supplemented by hierarchical state-space compression techniques. When deciding whether an implementation process *Impl* refines a normalised specification process *Spec*, FDR follows algorithms exploring the Cartesian product of the state spaces of *Spec* and *Impl* in a way comparable to conventional model checking. Although until now FDR has followed the explicit model checking approach, there has been some work on the symbolic model checking of CSP resulting in the BDD-based refinement checker ARC [27] and the model checker PAT [37], both of which exploit a fully compositional encoding of CSP processes. PAT verifies systems defined in a version of CSP enhanced with shared variables and, within the BMC framework, it uses specifications defined as reachability properties on the values of the shared variables, which requires a different model checking algorithm based on reachability and not on language containment.

This paper reports our attempts to integrate SAT-based BMC and temporal k-induction into FDR. The former technique is incomplete and as such is only suitable to detecting bugs. k-induction, however, is complete and can also be used for establishing the correctness of systems. Hence, to the best of our knowledge, this is the first attempt to apply unbounded SAT-based refinement checking to CSP. We propose an alternative Boolean encoding of CSP processes based on FDR's hybrid two-level approach for calculating the operational semantics using supercombinators [18]. As we deal with a problem that reduces to language inclusion instead of to reachability and due to the presence of invisible hidden actions in process algebras, the original syntactic translation of BMC to SAT cannot be applied directly and we adopt a semantic translation algorithm based on watchdog transformations [28]. Essentially, this involves reducing a refinement check into analysing a single process which is constructed by putting the implementation process in parallel with a transformed specification process. The latter plays the role of a watchdog that monitors and marks violating behaviours. Within the scope of this paper, we only consider the translation of *trace* refinement to SAT checking.

The result is a prototype tool SymFDR[1] which, when combined with state-of-the-art SAT-solvers such as MiniSAT 2.0 [15,14], sometimes outperforms FDR by a significant margin in finding counterexamples. We compare the performance of SymFDR with the performance of FDR, FDR used in a non-standard way, PAT [36] and, in some cases, NuSMV [7], Alloy Analyzer [20] and straight SAT encodings tailored to the specific problems under consideration.

[1] SymFDR's binaries are expected to be online shortly.

Download English Version:

https://daneshyari.com/en/article/435235

Download Persian Version:

https://daneshyari.com/article/435235

Daneshyari.com