Explicit-enumeration based Verification made Memory-efficient

Ratan Nalumasu,
Ganesh Gopalakrishnan *,
Department of Computer Science,
University of Utah, Salt Lake City, UT 84112
{ratan,ganesh}@cs.utah.edu

February 24, 1995

Abstract

We investigate techniques for reducing the memory requirements of a model checking tool employing explicit enumeration. Two techniques are studied in depth: (1) exploiting symmetries in the model, and (2) exploiting sequential regions in the model. The first technique resulted in a significant reduction in memory requirements at the expense of an increase in run time. It is capable of finding progress violations at much lower stack depths. In addition, it is more general than two previously published methods to exploit symmetries, namely scalar sets and network invariants. The second technique comes with no time overheads and can effect significant memory usage reductions directly related to the amount of sequentiality in the model. Both techniques have been implemented as part of the SPIN verifier.

Keywords: Formal Methods, Verification, Model Checking

1 Introduction

With the growing complexity of hardware and software, the need to formally verify them is being increasingly felt. Among the options available today, two of the prominent ones are based on deduction and model-checking [8]. Although both methods have their proponents, model-checking [3] is preferred when a relatively high degree of automation is desired, and when one-of-a-kind reactive behaviors are involved. Model-checking can be carried out either via implicit enumeration where the state graph is implicitly traversed using (for example) BDD-methods or via explicit enumeration where the state graph is explicitly traversed and processed using graph algorithms. Both these approaches have their own strengths. Also, both methods suffer from state explosion [16], combating which forms a central research problem. This paper is about combating state explosion in explicit-enumeration-based verification.

Space/Time Tradeoffs During Explicit Enumeration

Explicit enumeration forms the basis for a number of tools that have been used with great success in validating several real-life protocols [12, 10, 11]. One problem with explicit enumeration is that the available amount of memory often decides the size of the problem that can be handled; most explicit-enumeration-based tools give 100% “coverage” till this limit is reached, and give 0% coverage once this limit is exceeded. Designers combat this abrupt loss of coverage in several ways; almost always, they use techniques such as throwing away irrelevant states, reducing the dimensions of the arrays involved, etc. [12]. Although this is essential in any verification

*Supported in part by NSF award MIP 9215878 and APRA Contract N-0003995-C0018