Distributed Modular Model Checking

Jitka Crhová

Department of Computer Science
Faculty of Informatics
Masaryk University Brno, Czech Republic

Abstract

Model checking is a formal method that verifies whether a finite state model of a system satisfies a specification given as a temporal logic formula. The most severe problem model checking suffer from is the so called state explosion problem. Distribution is one of the techniques that combat the state explosion. The aim is to distribute the state space among a number of computers so as to be able to verify larger systems. Another approach that deals with the state explosion problem is modularity, i.e., exploiting the structure of the system.

We propose to employ modular techniques to distribute model checking problem. This can be useful especially for software, as the software model checking algorithms suffer from state explosion even severely that the hardware model checking techniques even when the system consist of one sequential finite-state component. Moreover, software programs have typically richer syntactic structure that can be exploited.

Besides elaborating a theoretical background for distributed model checking based on modular approach, we also intend to develop modular approaches to partitioning the state space, in particular to define partition functions that reduce the necessary communication in the distributed environment.

Recently, in [2] we have proposed a technique that extends the approach introduced in [3]. The state space is partitioned into several “partial state spaces”. The partial state space is modeled as a Kripke structure with border states. Each computer involved in the distributed computation owns a partial state space and performs a model checking algorithm on this incomplete structure. To be able to proceed, the border states are augmented by assumptions about the truth of formulas and the computers exchange assumptions about relevant states as they compute more precise information.

In [1] we have suggested several ways how to exploit syntactic information of software programs to improve the performance of the algorithm. The main idea is to partition the state space according to syntactic structure of a sequential process, i.e., according to program counter locations. The aim of the ongoing research is to develop partition techniques based on values of program variables. We intend to exploit abstraction and slicing techniques.

