DiVINE
The Distributed Verification Environment

J. Barnat\(^1\), L. Brim\(^2\), I. Černá\(^1\), P. Šimeček\(^2\)

Faculty of Informatics, Masaryk University, Brno, Czech Republic

Abstract
This paper presents basic concepts and the current state of a general distributed verification environment (DiVINE). The environment is meant to support the development of distributed enumerative model checking algorithms, to enable unified and credible comparison of these algorithms, and to make the distributed verification available for public use in a form of a distributed verification tool.

1 Introduction

In recent years, extensive research has been conducted in parallel and distributed model checking with the aim to push forward the frontiers of still tractable systems \([13,12,11,10,4,5]\). Many parallel and distributed algorithms have been developed and experimentally evaluated, mostly on a restricted set of verification problems. A few of them have been incorporated into existing verification tools. However, these distributed tools are still far from the standards met by sequential tools and in most cases their availability to the public is limited.

Another important aspect related to parallel and distributed model checking algorithms is that their performance analysis as published in original papers cannot serve for their credible comparison. This is primarily due to the fact that the hardware, the input models, and other circumstances differ from case to case making thus reported results incomparable. Most algorithms were implemented as research prototypes only using various data structures and different optimization techniques. As a consequence they simply cannot be executed on the same set of inputs. In addition, it is impossible to assure the same conditions when redoing the original experiments.

\(^1\) Research supported by the Academy of Sciences of Czech Republic grant No. 1ET408050503

\(^2\) Research supported by the Grant Agency of Czech Republic grant No. 201/03/0509

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science

URL: www.elsevier.nl/locate/entcs
In order to produce a fair comparison, the algorithms must be re-implemented on a common base and their behavior have to be re-examined on a common set of inputs and under the same conditions. This comparative evaluation can provide useful insight into their strengths and weaknesses leading to a more informed way of how to choose an appropriate algorithm for a given verification task.

In 2002 our group at the Faculty of informatics started the DivinE project with the aim to develop a distributed LTL model checking verification tool and at the same time to provide a platform for development and comparison of distributed enumerative model checking algorithms. The main goals of DivinE – Distributed Verification Environment can be summarized as follows:

(i) To use the environment as a platform for further development and experimental evaluation of enumerative parallel and distributed model checking algorithms.

(ii) To enable credible evaluation of existing enumerative algorithms with regard to their performance and characteristics under controlled conditions.

(iii) To create a ready-to-use distributed LTL model checker.

In this short presentation we aim to announce the DiVINE project to the PDMC community, introduce basic concepts and ideas used in DiVINE, describe its architecture and give the current status of work.

2 DivinE Project

The DiVINE project splits into two main parts: DiVINE TOOLSET and DiVINE LIBRARY. These parts address potential DiVINE users at two different levels: at the level of a tool user (DiVINE TOOLSET) and at the level of a tool developer (DiVINE LIBRARY). The overall structure of DiVINE is depicted in Figure 1.

2.1 DiVINE TOOLSET

The DiVINE TOOLSET is made of a set of various model checking algorithms (referred as tools) that are accessed uniformly via a graphical user interface. An inseparable part of DiVINE TOOLSET is a collection of verification problems and case studies, i.e. a collection of models and corresponding properties to be verified. This problem set should also serve for benchmark testing and evaluation. The native DiVINE specification language (DVE) has borrowed its principles from well established and by the community generally accepted description formalisms found in tools such as SPIN or UPPAAL. Each model of the system is described as a network of finite state machines with guarded transitions. The automata communicate either via shared memory (shared variables) or through buffered communication channels.
2.2 DiViNE Library

DiViNE Library is expected to be used by the researchers who intend to design, implement and experimentally evaluate a distributed model checking algorithm. The library is therefore designed to provide the potential programmer with a plethora of functions that are typically needed for the implementation. The programmer may thus focus on the core part of the algorithm design. Furthermore, for the comparison of different algorithms it is crucial to use the same level of implementation details.

As can be seen from Figure 1, DiViNE Library is divided into several more or less independent modules.

State Generator is a module that provides functions needed for the state space generation. These include primarily the function which computes the initial state and the function which computes immediate successors of a given state. In the case of LTL model checking, the module is responsible for computing the synchronous product of system automaton and negative claim automaton, and identifying accepting states of the synchronous product. State Generator supports the programmer by additional functionality, e.g. an interface to access some structural properties of the model.

Storage module is responsible for storing states to the local memory. It provides functions for inserting states to the set of visited states, testing membership of states in the set, and removing states from this set. In addition, the module is capable of storing additional pieces of information, as the so called...
appendix, for every state stored in the set. The module is able to provide a
unique identifier for each stored state, hence, the algorithm can manipulate
states and access their appendixes using small identifiers instead of working
with large state vectors.

The purpose of *HW Monitor* and *Reporter* modules is to continuously
monitor running algorithm and to feed the algorithm with information about
hardware utilization as well as to produce logs describing the behavior of the
algorithm during its execution. The standard POSIX signal mechanism is
used to scan and log measured quantities every second.

*Network* module is the core module of the DiVinE Library part. This
module implements basic routines for communication in the distributed setting
such as send and receive primitives. In addition to the basic network primitives, the module also implements mechanisms for message buffering, functions
for sending and receiving urgent messages, functions for partitioning the state
space, etc. As for high-level primitives, the module mainly provides functions
for synchronization. In particular, the module supports

(i) *barrier synchronization* which postpones computation on a computer un-
    til all other computers enter the barrier (this synchronization has been
    adopted from MPI standard),

(ii) *termination detection* which is a probe function returning *true* if all mes-
    sages that have been sent have also been received and processed, and all
    participating computers are idle, and returning *false* otherwise. Termination
detection can be repeatedly used for synchronization of participating
computers as well as for collecting global numbers of visited states,
    sent and received messages, etc.

Other modules that are not depicted in Figure 1 include functions sup-
porting partial order reduction, time profiling, counterexample generation,
and property automaton decomposition.

3 Current State

The DiVinE project is being implemented in C++ employing MPI standard
for network communication. All the basic functions of DiVinE Library, i.e.
state generator, storage, HW monitor, reporter, and network modules, have
already been implemented. The current alpha version of the library is ready
for public use [9].

As concerns DiVinE ToolSet, we have already implemented distributed
state space generation algorithm and several distributed LTL model check-
ing algorithms [2,3,1,6]. The algorithms were tested on models specified in
native DiVinE language. DiVinE ToolSet also allows to perform guided
simulation of a model and check a model for unreachable code.

Both DiVinE Library and DiVinE ToolSet have been successfully
tested with mpich, LAM/MPI, and GridMPI/YAMPII implementations of the
MPI standard. The source codes of the DiVinE project are freely available under GNU General Public License.

4 Future Work

In the future, we would like to improve both the design and the implementation of the library, develop an appropriate user interface, and improve existing model checking algorithms as well as implement additional ones [7,8,6]. Our nearest goal however is to release a stable version of the library and DiVinE TOOLSet.

Other future goals include design and implementation of functions supporting load balancing and caching of sent states. We would also like to extend the database of DiVinE models.

We have started the DiVSpin project in cooperation with the Research Groups at RWTH Aachen and TU München. The goals of the project are twofold. First, to enhance DiVinE by adding support for ProMeLa specification language allowing DiVinE TOOLSet to verify SPIN models. Second, to build a web-accessed platform for distributed verification and provide users with an access to appropriate hardware.

References


