Integrating BDD-based and SAT-based Symbolic Model Checking

Alessandro Cimatti[1], Enrico Giunchiglia[2], Marco Pistore[1], Marco Roveri[1], Roberto Sebastiani[3], and Armando Tacchella[2]

[1] ITC-IRST, Via Sommarive 18, 38050 Trento, Italy
{cimatti,pistore,roveri}@irst.itc.it
[2] DIST – Università di Genova, Viale Causa 13, 16145 Genova, Italy
{enrico,tac}@mrg.dist.unige.it
[3] Università di Trento, Via Sommarive 14, 38050 Trento, Italy
rseba@science.unitn.it

Abstract. Symbolic model checking is a very successful formal verification technique, classically based on Binary Decision Diagrams (BDDs). Recently, propositional satisfiability (SAT) techniques have been proposed as a computational basis for symbolic model checking, and proved to be an effective alternative to BDD-based techniques. In this paper we show how BDD-based and SAT-based techniques have been effectively integrated within the NuSMV symbolic model checker.

1 Introduction

Model checking [11, 20] is a formal technique for the verification of finite state systems. The system being analyzed is represented as a Finite State Machine (FSM), while the requirements to be satisfied are expressed in temporal logics, e.g. Computation Tree Logic (CTL), or Linear Temporal Logic (LTL). Model checking algorithms are based on the exhaustive analysis of the state space of the FSM. They are able to prove that the system satisfies the requirement, or, more importantly, are able to produce a counterexample, i.e. a behaviour of the FSM that violates the requirements. Model Checking is an extremely effective debugging technique, and is being applied in several application domains, ranging from the analysis of telecommunication protocols to reactive controllers to hardware designs.

Originally, model checking was implemented by means of "explicit-state" techniques, where single states of the FSM are analyzed and stored. One of the most notable examples of explicit-state model checking is SPIN [17], that is very effective in the analysis of asynchronous systems. In general, for many application domains, the large amount of computational resources needed to analyze real-size designs (the so-called state-explosion problem) may be a significant limitation. The introduction of *Symbolic* Model Checking [18] made it possible to explore state spaces of extremely large size. In symbolic model checking, instead of manipulating individual states, the algorithms manipulate *sets* of states. These are compactly represented and efficiently constructed by means of Binary Decision Diagrams [6] (BDDs), that are canonical forms for propositional formulae. Since the seminal work of McMillan [18], several mechanisms for a partitioned representation of finite state machines and different exploration styles [7, 22, 13] have allowed to increase the applicability of BDD-based model checking. Recently, a new form of symbolic model checking, commonly known as Bounded Model Checking [4], has been introduced. Bounded Model Checking is based on the encoding of a model checking problem into a propositional satisfiability (SAT) problem, and on the application of efficient SAT solvers. This approach, in the following called SAT-based model checking, relies on the enormous progress in the field of propositional satisfiability [19]. The approach is currently enjoying a substantial success in several industrial fields (see, e.g., [12], but also [5]), and opens up new research directions.

BDD-based and SAT-based model checking are often able to solve different classes of problems, and can therefore be seen as complementary techniques. The effective integration of BDD-based and SAT-based model checking techniques is very important to widen the spectrum of applicability of symbolic model checkers. Goal of this paper is to describe how the BDD-based and SAT-based approaches to symbolic model checking have been successfully integrated within the NuSMV model checker. In Section 2 we outline the NuSMV project. In Section 3 and 4 we describe the functionalities and the architecture of NuSMV2. In Section 5 we discuss some results and outline directions for future development.

2 The NuSMV Symbolic Model Checker

NuSMV is a symbolic model checker originated from the reengineering, reimplementation and extension of SMV [18], the original BDD-based model checker developed by McMillan et al. at CMU (SMV from now on). The NuSMV project aims at the development of a state-of-the-art symbolic model checker, designed to be applicable in technology transfer projects: it is a well structured, open, flexible and documented platform for model checking, and is robust and close to industrial systems standards [8].

The first version of NuSMV, called NuSMV1 in the following, basically implements BDD-based symbolic model checking. The second version of NuSMV (NuSMV2 in the following), inherits all the functionalities and the implementation style of the previous version. However, NuSMV2 significantly extends the functionalities of NuSMV1, and its internal structure departs from the one of NuSMV1. The main novelty in NuSMV2 is the integration of model checking techniques based on propositional satisfiability. Remarkably, the integration covers the whole input language of NuSMV. NuSMV2 is currently the only publicly available system that allows for both BDD-based and SAT-based model checking. In order to integrate SAT-based and BDD-based model checking, a major architectural redesign was carried out in NuSMV2, in order to make as many functionalities as possible independent of the actual model checking engine used. An example of this are the services provided by the modules implementing the preprocessing and reduction of the model to be analyzed. This allowed for the effective integration of the new SAT-based engine, and opens up toward the implementation of other model checking procedures.

NuSMV2 is the result of a cooperative project. IRST and the University of Trento carried out the activities related to model checking, while the University of Genova provided a package implementing reduced boolean circuits [1] and the state of the art SIM SAT solver [16]. The SIM solver is particularly effective in tackling problems arising from bounded model checking [12]. NuSMV2 is publicly available, under the GNU Lesser General Public License (LGPL), at `http://nusmv.irst.itc.it/`.

3 System Functionalities

NuSMV is able to process files written in an extension of the SMV language. In this language, it is possible to describe finite state machines by means of declaration and instantiation mechanisms for modules and processes, corresponding to synchronous and asynchronous composition, and to express a set of requirements in CTL and LTL. NuSMV can work batch or interactively, with a textual interaction shell.

An SMV file is processed in several phases. The first phases require the analysis of the input file, in order to construct an internal representation of the system to be analyzed. NuSMV2 neatly separates the input language in different layers, of increasing simplicity, that are incrementally eliminated. The first step, called *flattening*, performs the instantiation of module types, thus creating modules and processes, and produces a synchronous, flat model, where each variable is given an absolute name. The second step, called *boolean encoding*, maps a flat model into a boolean model, thus eliminating scalar variables. This second step takes into account the whole SMV language, including the encoding of bounded integers, and the set-theoretic and arithmetic functions and predicates. It is possible to print out the different levels of the input file, thus using NuSMV2 as a flattener. The same reduction steps are applied to the requirements. In addition, by means of the cone of influence reduction [2], it is possible to restrict the analysis of each property to the relevant parts of the model. This reduction can be extremely effective in tackling the state explosion problem.

The preprocessing is carried out independently from the model checking engine to be used for verification. After this, the user can choose whether to apply BDD-based or SAT-based model checking. In the case of BDD-based model checking, a BDD-based representation of the the Finite State Machine is constructed. In this step, different partitioning methods and strategies [21] can be used. Then, different forms of analysis can be applied: reachability analysis, fair CTL model checking, LTL model checking via reduction to CTL model checking, computation of quantitative characteristics of the model.

In the case of SAT-based model checking, NuSMV2 constructs an internal representation of the model based on Reduced Boolean Circuit (RBC), a representation mechanism for propositional formulae. Then, it is possible to perform SAT-based bounded model checking of LTL formulae. Given a bound on the length of the counterexample, a LTL model checking problem is encoded into a SAT problem. If a propositional model is found, it corresponds to a counterexample of the original model checking problem. With respect to the tableau construction in [4], enhancements have been carried out that can significantly improve the performances of the SAT checker. The system enters a loop, interleaving problem generation and solution attempt via a call to the SAT solver, and iterates until a solution is found or the specified bound is reached. Dual techniques for invariant checking [3] can be applied to invariant properties.

The properties are handled and shown to the user by a property manager, that is independent of the model checking engine used for the verification. This means that it is possible for the user to decide what solution method to adopt for each property. Furthermore, the counterexample traces being generated by both model checking modules are presented and stored into a unique format. Similarly, the user can simulate the behaviour of the specified system, by generating traces either interactively or randomly. Simulation can be carried out both via BDD-based or SAT-based techniques.

4 System Architecture

In the development of NuSMV, particular care is directed to the architectural design, in order to obtain an open architecture that can be integrated within different design environments, and customized depending on the application domain. Therefore, the architecture of NuSMV2 has been deeply revised and extended with respect to NuSMV1, in order to allow for a clean and effective integration of SAT-based techniques and to overcome some limitations of NuSMV1. A high level view of the internal structure of NuSMV2 is reported in Figure 1. The architecture is composed of the following main modules.

Flattening: The Flattening module implements the parsing of the model, some consistency checks to guarantee the well foundedness of the definitions, and eliminates processes and modules, producing a flat, scalar model, and a set of flat properties.

Encoding: The Encoding is responsible for mapping the flat, scalar model into a boolean model. This requires the introduction of the suitable boolean variables, depending on the range of the scalar variables being manipulated. For instance, for a bounded integer variable x, ranging from 0 to 255, 8 boolean variables $x_1, \ldots, x_8$ are defined. Furthermore, an encoding that associates each of the proposition $x = v$ into a corresponding assignment to $x_1, \ldots, x_8$ is constructed. Then, for each atomic proposition in the program, the corresponding boolean expression is constructed. For instance, the atomic proposition $(x+y) \leq z$ would be associated with a boolean expression in the boolean variables associated with x, y and z. This operation is carried out by means of Algebraic Decision Diagrams, particular forms of Decision Diagrams with non-boolean leaves.

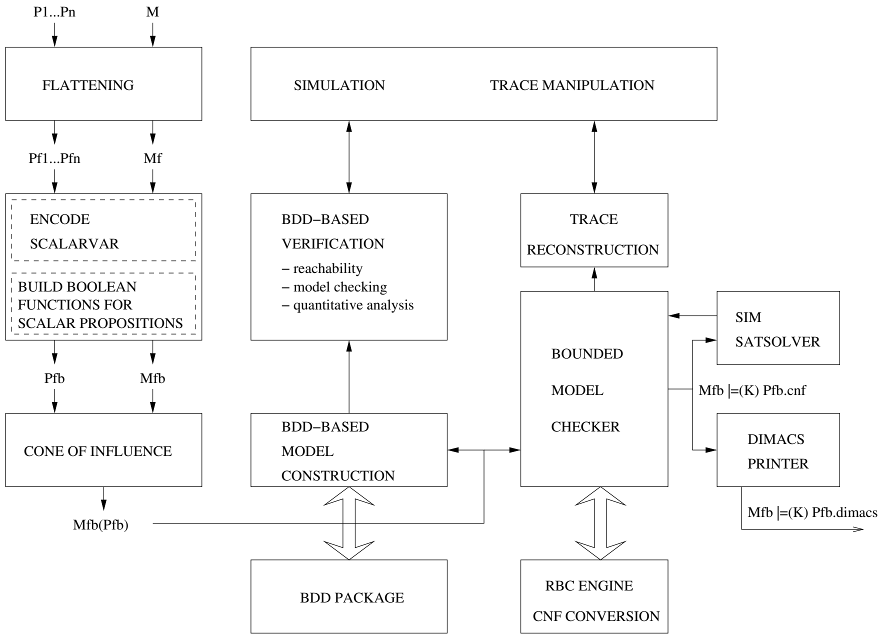

Fig. 1. The internal structure of NuSMV2.

Cone of Influence. This module implements the routines to restrict the analysis to a reduced FSM, containing only the relevant variables for each property. This reduction is amenable both for BDD-based and SAT-based model checking.

BDD-based Model Construction: The BDD-based model construction module implements the Finite State Machine corresponding to the input file in terms of BDDs. An explicit data structure for FSM's is provided, that allows to encapsulate the actual construction/partitioning method applied. It is therefore possible to have different FSM's associated to different verification problems.

BDD-based Verification: The BDD-based verification routines implement reachability analysis, LTL and CTL model checking, and quantitative analysis, in terms of the FSM data structures provided by BDD-based model construction. CTL model checking is implemented directly, while LTL model checking is reduced to a CTL model checking problems by means of a tableau construction, as described in [9]. The analysis of quantitative properties, such as the computation of the least distance between the occurrence of two given events, is carried by dedicated algorithms. All the operations only rely on image and preimage computations, and are independent of the actual partitioning mechanism.

BDD package: The functionalities for the manipulation and storage of BDDs is provided by the BDD module. This module is based on the state-of-the-art Colorado University Decision Diagram (CUDD) package developed by Fabio Somenzi [24]. An additional layer encapsulates the CUDD functionalities in order to provide a uniform interface that hides low-level issues related to garbage collection of BDDs.

Bounded Model Checker: The Bounded Model Checker module provides the SAT-based model checking functionalities. It interacts with the RBC package to generate a RBC-based model representation. At the lowest level, an association between state variables at different time instants and the corresponding RBC variables is defined. The construction is optimized by means of memoizing techniques, in order to avoid the recomputation of frequently used RBCs. The variable association schema is implemented in such a way that full blown parallel substitution can be replaced by a shifting operation. Once the internal representation of the model is complete, the Bounded Model Checker can generate the SAT problems corresponding to a given formula, with a construction that extends the one in [4]. In particular, the construction takes into account all the components of the model (e.g., fairness constraints, invariants). Several encodings of the property are possible, depending if we are checking for a violation occurring exactly at step k or at a step $\leq k$, and with different loop-back structures for the counterexample. The problem is generated as an RBC, that is then converted in CNF format and provided in input to the SIM solver. If a model is found, then it is returned to the Bounded Model Checker, whose final step it to activate the trace reconstruction of the counterexample. The produced model checking problems can also be printed out in the standard DIMACS format, thus allowing for the stand-alone use of other SAT solvers.

Reduced Boolean Circuit (RBC): The RBC package implements a simplified version of the RBC data structure and the associated primitives for storing and manipulating propositional formulas, see [1]. The RBC package comes with a depth-first traversal routine, which allows to search the RBC, applying a given function (passed as a parameter) to each node being visited either in-order, preorder, or post-order. This function is at the basis of the CNF converter. The CNF converter generates an equi-satisfiable formula obtained from the RBC by applying a structure-preserving transformation. The CNF converter also marks the variables occurring in the RBC. The ability to distinguish between the "independent" variables (i.e., the ones occurring in the formula before the CNF conversion) and the "dependent" variables (i.e., the ones introduced during the CNF conversion) can be extremely useful in driving the solver [14, 15, 23, 12].

SIM SAT Solver: SIM is an efficient SAT solver based on the Davis-Logemann-Loveland procedure. Two are the distinguishing features of SIM. First, it can limit the branching to a subset of the variables, assuming that the others can be assigned by unit propagation. Second, it allows for relevance learning, and branching heuristics based on boolean constraint propagation which analyze the whole set of (relevant) variables. As [23, 12] show, these features can produce dramatic speed-ups in the overall performances of the SAT checker, and thus of the whole system. SIM also features many other branching heuristics, and size learning (see [16]).

Simulation/Trace Manipulation: The simulation package allows for the interactive and random simulation of the behaviour of the model being processed. It is compatible both with the BDD-based and the SAT-based representation, and encapsulates a uniform trace handling mechanism through which the stored traces can be inspected and rerun.

5 Conclusions

NuSMV is a robust, well structured and flexible platform for symbolic model checking, designed to be applicable in technology transfer projects. In this paper, we have shown how BDD-based and SAT-based model checking are integrated in the new version of NuSMV, that significantly extends the previous version. In particular, we have discussed the functionalities and the architecture of NuSMV2, that integrates SAT-based state of the art verification techniques, is able to working as a problem flattener in DIMACS format, and tackles the state explosion with cone of influence reduction.

NuSMV2 is currently being used as the verification kernel of a CAD tool developed in a technology transfer project, where an imperative style programming language is used to describe embedded controllers. The integration of decomposition techniques (e.g., abstraction and compositional verification) is under development. In the future, we plan to investigate a tighter integration between BDD-based and SAT-based technologies. The new internal architecture also opens up the possibility to integrate different boolean encodings [10] and different (e.g., non boolean) verification engines.

References

1. Parosh Aziz Abdulla, Per Bjesse, and Niklas Eén. Symbolic reachability analysis based on SAT-solvers. In Susanne Graf and Michael Schwartzbach, editors, *Proc. Tools and Algorithms for the Construction and Analysis of Systems TACAS, Berlin, Germany*, volume 1785 of *LNCS*. Springer-Verlag, 2000.
2. S. Berezin, S. Campos, and E. M. Clarke. Compositional reasoning in model checking. In *Proc. COMPOS*, 1997.
3. A. Biere, A. Cimatti, E. Clarke, M. Fujita, and Y. Zhu. Symbolic Model Checking Using SAT Procedures instead of BDDs. In *Proc. 36th Conference on Design Automation*, 1999.
4. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proceedings of the Fifth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '99)*, 1999.
5. A. Borälv. A Fully Automated Approach for Proving Safety Properties in Interlocking Software Using Automatic Theorem-Proving. In S. Gnesi and D. Latella, editors, *Proceedings of the Second International ERCIM Workshop on Formal Methods for Industrial Critical Systems*, Pisa, Italy, July 1997.
6. R. E. Bryant. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3):293–318, September 1992.
7. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. *Information and Computation*, 98(2):142–170, June 1992.

8. A. Cimatti, E.M. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: a new Symbolic Model Verifier. In N. Halbwachs and D. Peled, editors, *Proceedings Eleventh Conference on Computer-Aided Verification (CAV'99)*, number 1633 in Lecture Notes in Computer Science, pages 495–499, Trento, Italy, July 1999. Springer-Verlag.
9. E. Clarke, O. Grumberg, and K. Hamaguchi. Another Look at LTL Model Checking. *Formal Methods in System Design*, 10(1):57–71, February 1997.
10. E. Clarke and X. Zhao. Word Level Symbolic Model Checking: A New Approach for Verifying Arithmetic Circuits. Technical Report CMU-CS-95-161, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3891, USA, May 1995.
11. E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time tem poral logic. In *Logic of Programs: Workshop*. Springer Verlag, May 1981. Lecture Notes in Computer Science No. 131.
12. Fady Copty, Limor Fix, Enrico Giunchiglia, Gila Kamhi, Armando Tacchella, and Moshe Vardi. Benefits of bounded model checking at an industrial setting. In *Proceedings of CAV 2001*, pages 436–453, 2001.
13. Ranan Fraer, Gila Kamhi, Barukh Ziv, Moshe Y. Vardi, and Limor Fix. Prioritized traversal: Efficient reachability analysis for verification and falsification. In *Proceedings of the 12th International Conference on Computer Aided Verification*, pages 389–402. Springer, July 2000.
14. E. Giunchiglia, A. Massarotto, and R. Sebastiani. Act, and the rest will follow: Exploiting determinism in planning as satisfiability. In *Proc. AAAI*, 1998.
15. E. Giunchiglia and R. Sebastiani. Applying the Davis-Putnam procedure to non-clausal formulas. In Evelina Lamma and Paola Mello, editors, *Proceedings of AI*IA'99: Advances in Artificial Intelligence*, pages 84–94. Springer Verlag, 1999.
16. Enrico Giunchiglia, Marco Maratea, Armando Tacchella, and Davide Zambonin. Evaluating search heuristics and optimization techniques in propositional satisfiability. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Proceedings of IJCAR 2001*, volume 2083 of *Lecture Notes in Computer Science*, pages 347–363. Springer, 2001.
17. G. J. Holzmann. The model checker Spin. *IEEE Trans. on Software Engineering*, 23(5):279–295, May 1997. Special issue on Formal Methods in Software Practice.
18. K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publ., 1993.
19. Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient sat solver. In *Proceedings of the 38th Design Automation Conference*, pages 530–535. ACM, 2001.
20. J.P. Quielle and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proceedings of the Fifth International Symposium in Programming*, 1981.
21. R. K. Ranjan, A. Aziz, B. Plessier, C. Pixley, and R. K. Brayton. Efficient BDD algorithms for FSM synthesis and verification. In *IEEE/ACM Proceedings International Workshop on Logic Synthesis*, Lake Tahoe (NV), May 1995.
22. K. Ravi and F. Somenzi. High-density reachability analysis. In *International Conference on Computer Aided Design*, pages 154–158, Los Alamitos, Ca., USA, November 1995. IEEE Computer Society Press.
23. O. Shtrichman. Tuning SAT checkers for bounded model-checking. In *Proc. 12th International Computer Aided Verification Conference (CAV)*, 2000.
24. F. Somenzi. CUDD: CU Decision Diagram package — release 2.1.2. Department of Electrical and Computer Engineering — University of Colorado at Boulder, April 1997.