Improvements to the Implementation of Interpolant-Based Model Checking

João Marques-Silva

Technical University of Lisbon,
IST/INESC-ID, Portugal
jpms@sat.inesc-id.pt

Abstract. The evolution of SAT technology over the last decade has motivated its application in model checking, initially through the utilization of SAT in bounded model checking (BMC) and, more recently, in unbounded model checking (UMC). This paper addresses the utilization of interpolants in UMC and proposes two techniques for improving the original interpolant-based UMC algorithm. These techniques include improvements to the computation of interpolants, and redefining the organization of the unbounded model checking algorithm given the information extracted from interpolant computation.

1 Introduction

The utilization of Boolean Satisfiability (SAT) in Model Checking has been the subject of intensive research in recent years. The main result of this effort has been a number of very competitive incomplete and complete algorithms for checking safety properties (see [3] for a comprehensive list of references and an extended version of this paper). Moreover, SAT-based model checking has also been rapidly adopted by industry, and a number of vendors have included SAT-based Model Checking in their tools. This paper describes preliminary work on optimizing the utilization of interpolants in SAT-based model checking [4]. Two techniques are proposed and evaluated. First, we propose the computation of interpolants directly from the proof trace and skip the generation of the resolution proof, and study the implementation of techniques for eliminating redundancy from the computed interpolants. Second, we propose to utilize information from the fixed-point checks of the UMC algorithm for redefining the organization of the UMC algorithm.

2 Interpolant-Based Unbounded Model Checking

The generic propositional formula associated with SAT-based bounded model checking is the following [2]:

$$\psi_{\text{BMC}}^{j,k} = \psi_I(Y/Y_0) \wedge \bigwedge_{i=0}^{k-1} \psi_T(Y/Y_i, Y'/Y_{i+1}) \wedge (\bigvee_{i=j}^{k} \neg\psi_S^i) \tag{1}$$

This formula represents the unfolding of the state machine for k computation steps, where $\psi_I(Y/Y_0)$ represents the initial state, $\psi_T(Y/Y_i, Y'/Y_{i+1})$ represents the transition relation between states X_i and X_{i+1}, and ψ_S^i represents the target property in computation step i. Given the BMC propositional formula $\psi_{\text{BMC}}^{j,k}$, it is straightforward to generate a CNF formula $\varphi_{\text{BMC}}^{j,k}$. The resulting formula can then be evaluated by a SAT solver. Recent work on SAT-based UMC has addressed the utilization of interpolants [4], with quite promising experimental results. McMillan's [4] interpolant-based UMC algorithm can be organized into two main phases: a BMC step, where the circuit is unfolded, and the existence of a counterexample is checked, and a UMC step, where the existence of a fixed-point is tested. Whereas the first phase corresponds essentially to the standard BMC algorithm, the second phase requires the iterative computation of interpolants until a fixed-point is reached or a (possibly) false counterexample is identified. See [3] for a detailed description of McMillan's UMC algorithm.

3 Optimizations to the Basic UMC Algorithm

This section addresses two optimizations to the basic interpolant-based UMC algorithm proposed by McMillan [4]. First, we address the construction and simplification of interpolants. Afterwards, we show how to exploit the information from the interpolant iteration phase for rescheduling either the UMC or the BMC loops. As noted by McMillan [4], interpolants obtained from unsatisfiability proofs are highly redundant Boolean expressions. In [4] the author proposes the utilization of BDDs, but no details are provided. For complex problem instances, that yield hard instances of SAT, with large unsatisfiability proofs, the interpolants before simplification can reach extremely large sizes. Our experience has been that interpolants before simplification can be more than two orders of magnitude larger than the resulting interpolants after simplification. Moreover, although modern SAT solvers can easily be instructed to generate proof traces, the generation of the actual unsatisfiability proof must be performed after the SAT solver terminates and the proof trace is concluded. A key observation is that one can avoid generating the unsatisfiability proof, and construct the interpolant directly from the proof trace.

Next we outline two algorithms for creating interpolants directly from proof traces. We should note that the organization of the two algorithms allows fairly different results in terms of the worst-case memory requirements, as illustrated in Section 4 for real-world model checking problem instances. Moreover, both algorithms utilize Reduced Boolean Circuits [1] for representing Boolean expressions, thus ensuring that constants and duplicate nodes are eliminated.

The first algorithm consists of a breadth-first traversal of the proof trace, that at each node creates a Boolean expression as indicated by the definition of interpolant (see [4]). We refer to this approach as the BFS algorithm. A key drawback of the BFS algorithm is that a large number of Boolean expressions need to be created, most of which are eventually deleted by applying the simplification techniques described above. Hence, the BFS algorithm often spends a large amount of time creating Boolean expressions that are eventually eliminated. The second algorithm consists of a depth-first traversal of the proof trace, applying the simplification techniques described above wherever possible, and eliminating depth-first visits whenever the (constant) value of a Boolean expression is known. We refer to this second approach as the DFS algorithm.

Next, we address techniques for exploiting the information provided by the UMC step of the UMC algorithm. Suppose the current unfolding size consists of K time frames. Moreover, assume the interpolant iteration procedure is executed I times, until a (possibly) false counterexample is identified. According to the definition of computed interpolants, this means that the target property cannot be satisfied within $K + I - 1$ time frames. As a result, the property cannot be satisfied for any unfolding with size no greater than $K + I - 1$ time frames. Hence, instead of a fixed policy of incrementing the size of the unfolding by INC time frames, we can safely consider the size of the next unfolding to be $K + I$ time frames. Observe that the information from interpolant computation can be used for other purposes. For example, instead of rescheduling the BMC loop to $K + I$ time steps, we can simply utilize a SAT solver more effective at *proving unsatisfiability*, and check the fixed-point earlier than $K + I$ time steps. Moreover, and since the information from the interpolant iteration procedure allows rescheduling the BMC loop, we can also reschedule the next unfolding for which to iterate interpolants and check the existence of a fixed-point, i.e. the UMC step. In general, this can be done for every unfolding at which the BMC step is evaluated.

The potential gains introduced with rescheduling can be significant. Assume a state machine and safety property such that a counterexample can be identified with an unfolding of T time frames. Moreover, assume that the BMC loop increases the unfolding by 1 time frame each time, that the initial unfolding size is 1, and that the interpolant iteration procedure runs for $T-K$ iterations for an unfolding size of K time frames (observe that if a counterexample exists, then we cannot iterate the computation of interpolants more than $T - K$ times). In this case, rescheduling guarantees that the UMC step is invoked only once, and so the number of times the SAT solver is invoked is $2 + 2 \times (T - 1) = O(T)$. In contrast, without rescheduling, the number of times the SAT solver is invoked is $T + 2 \times \sum_{i=1}^{T-1}(T - i) = O(T^2)$.

4 Results

In order to evaluate the effectiveness of the proposed techniques we implemented the algorithm described in [4], and integrated the optimizations described in the previous section. Moreover, a state of the art SAT solver was used. The experiments have been run under Linux RH 9, on a Pentium 2.8 GHz machine, with 1 GByte of RAM. Two classes of instances are considered. First, we consider a set of standard counters, for which a counterexample exists. For these instances the property requires not all state bits to be simultaneously assigned value 1. Second, we consider a set of instances (I11, I12, I21, I31, I32 and I33) obtained

Table 1. Experimental results

Instance	BFS & No-reschedule	DFS & Reschedule BMC
4bit-counter	0.31	0.09
5bit-counter	3.86	0.84
6bit-counter	21.36	10.41
7bit-counter	1780.68	175.69
I12	255.77	272.47
I11	75.28	81.89
I31	83.51	90.08
I32	19.66	14.89
I33	17.44	13.09
I21	24.93	26.48
Total Time	2282.8	685.9

from real-world examples. For these instances, I11, I12, I21 and I31 do not have a counterexample, whereas I32 and I33 have counterexamples.

Some preliminary results are shown in Table 1. Two configurations are considered: the BFS algorithm with no rescheduling, and the DFS algorithm with rescheduling. In both cases interpolants are computed directly from the proof trace. The results indicate that the proposed techniques are promising, allowing an average speedup of 3.3 over our base implementation of the UMC algorithm.

5 Conclusions

This paper proposes techniques for improving the utilization of interpolants in SAT-based unbounded model checking. As the results illustrate, improvements can be obtained from a careful implementation of the interpolant computation algorithm, and from exploiting the information provided by the procedure for iterating the computation of interpolants. For specific classes of instances, both artificially generated and obtained from industrial designs, the improvements can exceed several orders of magnitude. The utilization of interpolants in SAT-based model checking shows promise for future improvements, mostly related with exploiting the information represented by computed interpolants. Moreover, additional effective techniques for reducing the final or intermediate size of interpolants may play a crucial role in the utilization of interpolants in SAT-based model checking.

References

1. P. A. Abdulla, P. Bjesse, and N. Eén. Symbolic reachability analysis based on SAT solvers. In *Proc. TACAS*, 2000.
2. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proc. TACAS*, March 1999.
3. J. Marques-Silva. Optimizing the utilization of interpolants in SAT-based model checking. Technical Report RT-01-05, INESC-ID, January 2005.
4. K. L. McMillan. Interpolation and SAT-based model checking. In *Proc. CAV*, 2003.