VaqUoT: A Tool for Vacuity Detection

Mihaela Gheorghiu and Arie Gurfinkel

Department of Computer Science, University of Toronto,
Toronto, ON M5S 3G4, Canada.
Email: {mg,arie}@cs.toronto.edu

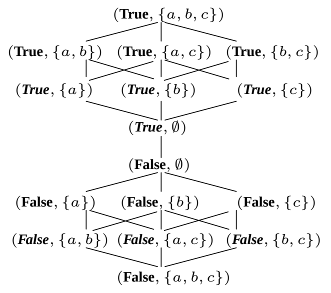
Abstract. This paper presents VaqUoT – a University of Toronto tool for vacuity detection, built on top of NuSMV. In one model-checking pass, VaqUoT establishes the truth value of a CTL formula as well as the largest set of non-overlapping subformulas in which that formula is vacuous. We describe the tool and evaluate its performance.

During model-checking, properties are sometimes satisfied by models for the wrong reasons. Suppose a CTL formula $\psi = AG\ (r \vee y \vee g)$ is checked against a model of a traffic-light controller, where atomic propositions r, y, and g stand for the colors of the light: red, yellow, and green, respectively. The formula is intended to express that in every state the light has one of these colors. This requirement may not be satisfied, even if ψ passes the check: it is possible for the model to be overconstrained so that the light always stays red. In such cases, an answer "true", given usually by model-checkers, is insufficient; a user needs to know *why* the formula is satisfied. Vacuity detection [2] can help, by determining whether some parts of the formula do not matter for the verification, *i.e.*, are *vacuous*. For instance, y and g should be reported as vacuous in ψ.

Although various approaches to vacuity detection have been proposed (*e.g.*, [1, 2, 6]), few implementations have been reported [1, 7], and to our knowledge none are publicly available. Our tool VaqUoT is publicly available as a patch for the open-source model-checker NuSMV. VaqUoT is based on techniques described in [5], where a multi-valued lattice is introduced for the detection of all vacuous subformulas. Since this lattice does not immediately lead to an efficient implementation, here we consider a simpler lattice, but a similar approach.

Given a model and a CTL formula, VaqUoT checks whether the formula is true in the model, and reports all the vacuous atomic propositions. Following [6], we consider a proposition vacuous if it can be replaced by a constant (**True** or **False**) without affecting the value of the formula in the model. We treat different occurrences of the same atomic proposition as different propositions. When the formula is true, VaqUoT reports whether all of its atomic propositions are vacuous (**Vacuously True**), none of them are vacuous (**Non-Vacuously True**), or some of the atomic propositions are vacuous (**Vacuously True** followed by a list of the vacuous propositions). Similar answers are given when the formula is false.

Implementation. The basis of VaqUoT is a multi-valued "vacuity" lattice and a translation of CTL formulas into this lattice. Instead of the formulas begin interpreted over the Boolean lattice $(\{\textbf{True}, \textbf{False}\}, \leq)$, they are interpreted over the *vacuity lattice* $\mathcal{L}_V = (\{\textbf{True}, \textbf{False}\} \times 2^A, \sqsubseteq)$, where 2^A is the powerset of the set A of atomic propositions. Lattice $\mathcal{L}_V$ is determined only by the number of atomic propositions



Model	Formulas		Basic MC		Naive VD		VaqUoT
	Total	Vacuous	Memory	Time	Witnesses	Time	
`elevator3l`	45	26	43.5	1690.16	399	5228.94	1441.22
`guidance`	23	16	10	54.18	244	306.99	274.81
`production -cell`	15	15	7.2	42.41	187	228.87	184.08
`abp10`	4	3	10.6	83.18	26	316.63	304.51
`fgs5`	6	2	106	189.57	82	239.04	191.92
`msi_wtrans`	15	3	10.3	30.21	81	53.63	83.98
`luckySeven`	4	0	12.9	469.33	20	1257.11	842.48
`eisenberg`	5	4	3	11.31	25	35.77	39.77
`ticTacToe`	42	3	8.9	15.81	363	68.72	102.51

Fig. 1. (a) Vacuity lattice for three atomic propositions; (b) Experimental results.

in the formula being checked. An element $(t, s) \in \mathcal{L}_V$ is a possible result of vacuity detection, showing that the formula has truth value t, and the largest subset of its atomic propositions that are vacuous is s. For any $u, v \in 2^A$, $(\mathbf{False}, u) \sqsubseteq (\mathbf{True}, v)$, $(\mathbf{True}, u) \sqsubseteq (\mathbf{True}, v)$ iff $u \subseteq v$, and $(\mathbf{False}, u) \sqsubseteq (\mathbf{False}, v)$ iff $v \subseteq u$ (note the reversal of set inclusion). The top element of $\mathcal{L}_V$ is $(\mathbf{True}, A)$, or **Vacuously True**. The bottom element is $(\mathbf{False}, A)$, or **Vacuously False**. The vacuity lattice for three atomic propositions a, b, c is depicted in Figure 1(a).

In VaqUoT, we replace each atomic proposition a of φ by $((a \wedge \mathbf{VT}_{A\setminus a}) \vee \mathbf{VF}_{A\setminus a})$, where $\mathbf{VT}_{A\setminus a}$ and $\mathbf{VF}_{A\setminus a}$ denote lattice values $(\mathbf{True}, A \setminus \{a\})$, and $(\mathbf{False}, A \setminus \{a\})$, respectively. All replacements are done simultaneously. The resulting multi-valued formula is then model checked. The justification for this translation follows from [5]. For example, VaqUoT reports that the traffic-light formula ψ defined earlier holds and is vacuous in both y and g, by outputting $(\mathbf{True}, \{y, g\})$.

In our implementation, we encode each value of the vacuity lattice $\mathcal{L}_V$ as a 32-bit word. The least-significant bit represents the truth: 1 for **True**, 0 for **False**. The other bits represent the vacuity: 0 for vacuous, 1 for non-vacuous. For instance, for the traffic-light formula ψ, the lattice value $(\mathbf{True}, \{y, g\})$ is represented by the word $00\ldots00011$, where the rightmost 0011 means, respectively, that g and y are vacuous, r is not, and the truth value is **True**. Thus, lattice operations can be efficiently implemented bitwise. The fixed word length, which could be increased from 32 to 64 or 128, limits the number of atomic propositions in the formulas we can check efficiently to $31, 63, 127$, respectively. Bit vectors of arbitrary length could be used, at the cost of increasing the complexity of lattice operations.

VaqUoT is built on top of NuSMV, which uses the CUDD package for the implementation of decision diagrams (DDs) [4]. We have implemented multi-valued DDs using CUDD ADDs and changed the interface between NuSMV and CUDD so that our multi-valued operations are performed instead of their BDD counterparts. These modifications do not affect the complexity of decision diagram operations or fixpoint computations, but they may affect performance, since the decision diagrams may be larger. Our changes are compatible with the various NuSMV optimizations (*e.g.*, cone of influence, dynamic reordering, partitioning). The tool is available as a patch for NuSMV v. 2.1.2, from `www.cs.toronto.edu/fm/vaquot.html`.

Experiments. A few experiments comparing VaqUoT with basic model checking and with a naive approach to vacuity detection are reported in Figure 1(b). The naive approach consists of separately replacing each atomic proposition by **True** and then by **False** and check the resulting formulas, in addition to the original formula; all these formulas are called *witnesses*. The number of witnesses reported in Figure 1(b) is the actual number of formulas checked in the naive approach, which we implemented on top of NuSMV as well. The experiments were performed on a Dell PC with a 2.4 GHz Intel Celeron CPU and 512 MB of RAM, running Linux 2.4.20. Models `guidance`, `production-cell`, `abp10`, and `msi_wtrans`, and most of their properties are from the NuSMV distribution. `elevator3l` is a model of a three-floor elevator system written by students taking the Automated Verification class at Univ. of Toronto, and `fgs5` is a proprietary model for a flight-guidance system. Models `luckySeven`, `eisenberg`, and `ticTacToe` are SMV translations of their Verilog counterparts distributed with the VIS model checker. For each model, we report the total number of formulas checked and how many were found vacuous (Formulas), the total memory (in MB) and time (in seconds) used by model-checking without vacuity detection (Basic MC), the total number of witnesses and the time used by the naive vacuity detection (Naive VD), and the running time of VaqUoT. As it can be seen, VaqUoT performs better than the naive approach in most cases, and by a considerable margin in some: our algorithm avoids much of the redundant work performed by the naive approach. In the cases where VaqUoT performs worse, we observed that the sizes of the decision diagrams are the bottleneck, and we are investigating ways to overcome this.

The method of [7] is the closest to ours, from related work. In [7], witnesses are generated and checked in parallel and compositionally, by a bottom-up exploration of the parse tree of a formula, with explicit caching of intermediate results. The representation of witnesses is explicit as well. All these are implicit in the multi-valued decision diagrams in our implementation. True and false formulas are treated differently, whereas VaqUoT handles both uniformly in one pass. Extensive experiments and comparisons between the two methods remain for future work; however, the results shown in Figure 1(b), specifically, for the last three examples (used also in [7]), indicate that both tools exhibit a similar improvement over the naive approach. In addition, for `eisenberg` and `ticTacToe`, VaqUoT found more vacuous passes.

Complementary to our vacuity checking of CTL fromulas using BDD-based techniques, the work of [8] addresses vacuity checking of LTL formulas, implemented using SAT-based methods. In a parallel development, [3] re-examines the meaning of vacuity in terms of system versus environment behavior, and argues that current vacuity checking methodology produces too many false positives, that is, cases of vacuity that do not indicate problems. As an alternative, [3] proposes checking when formulas pass/fail solely due to errors in the environment model, and shows on a realistic case study that this new methodology discovers truly problematic cases of vacuity.

References

1. R. Armoni, L. Fix, A. Flaisher, O. Grumberg, N. Piterman, A. Tiemeyer, and M. Vardi. "Enhanced Vacuity Detection in Linear Temporal Logic ". In *Proc. of CAV'03*, volume 2725 of *LNCS*, pages 368–380, 2003.
2. I. Beer, S. Ben-David, C. Eisner, and Y. Rodeh. "Efficient Detection of Vacuity in ACTL Formulas". In *Proc. of CAV'97*, volume 1254 of *LNCS*, pages 279–290, 1997.

3. M. Chechik, M. Gheorghiu, and A. Gurfinkel. "Finding Environmental Guarantees". CSRG Technical Report, submitted for publication, University of Toronto, April 2006.
4. A. Cimatti, E.M. Clarke, , E. Giunchilia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. "NUSMV Version 2: An Open Source Tool for Symbolic Model Checking". In *Proc. of CAV'02)*, volume 2404 of *LNCS*, pages 359–364, 2002.
5. A. Gurfinkel and M. Chechik. "How Vacuous Is Vacuous?". In *Proc. of TACAS'04*, volume 2988 of *LNCS*, pages 451–466, 2004.
6. O. Kupferman and M. Vardi. "Vacuity Detection in Temporal Model Checking". In *Proc. of CHARME'99*, volume 1703 of *LNCS*, pages 82–96, 1999.
7. M. Purandare and F. Somenzi. "Vacuum Cleaning CTL Formulae". In *Proc. of CAV'02*, volume 2404 of *LNCS*, pages 485–499, 2002.
8. J. Simmonds, J. Davies, and A. Gurfinkel. "VaqTree: Exlpoiting Resolution Proofs for LTL Vacuity Detection". accepted at fm'06 tool session, University of Toronto, June 2006.