BLACK: A fast, flexible and reliable LTL satisfiability checker

BLACK: A Fast, Flexible and Reliable LTL Satisfiability Checker

Luca Geatti[1,2], Nicola Gigante[3] and Angelo Montanari[1]

[1]University of Udine, Italy
[2]Fondazione Bruno Kessler, Trento, Italy
[3]Free University of Bozen-Bolzano, Italy

Abstract

BLACK, short for *Bounded Ltl sAtisfiability ChecKer*, is a recently developed software tool for satisfiability checking of *Linear Temporal Logic* (LTL) formulas. It supports formulas using both *future* and *past* operators, interpreted over both *infinite* and *finite* traces. At its core, BLACK uses an incremental SAT encoding of the one-pass tree-shaped tableau for LTL recently developed by Reynolds, which guarantees completeness thanks to its particular pruning rule. This paper gives an overview of the tool, surveys the main design choices underlying its implementation, describes its features and discusses potential future developments.

Keywords

Linear Temporal Logic, SAT, Tableaux methods

1. Introduction

Linear Temporal Logic (LTL) is the de-facto standard modeling language for temporal properties of reactive systems, both in formal verification and artificial intelligence. It is a modal logic usually interpreted over infinite state sequences, but the finite-trace semantics has recently gained attention as well [1, 2].

One of the most important tasks performed on LTL formulas, and one of the first to have been studied [3, 4], is *satisfiability checking*, *i.e.*, checking whether there exists a model of a given formula. Satisfiability checking is important in many applications, from automated planning [5] to sanity checking of specifications [6]. Many techniques have been developed over the years to solve the problem, ranging from tableau systems [4, 7, 8] to reduction to model checking [9], from temporal resolution [10, 11, 12] to automata-theoretic techniques [13]. In particular, tableau methods are useful both as a theoretical tool and as an effective solving technique. Classic tableau methods are *graph shaped* [4, 7], since they create a graph structure that is then traversed in a second pass to find suitable models for the formula. In contrast, some one-pass tableaux methods have been proposed by Schwendimann [14] and, more recently,

OVERLAY 2021: 3rd Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, September 22, 2021, Padova, Italy
lgeatti@fbk.eu (L. Geatti); nicola.gigante@unibz.it (N. Gigante); angelo.montanari@uniud.it (A. Montanari)
https://users.dimi.uniud.it/~luca.geatti/ (L. Geatti); https://www.inf.unibz.it/~gigante/ (N. Gigante); https://users.dimi.uniud.it/~angelo.montanari/index.php (A. Montanari)
0000-0002-7125-787X (L. Geatti); 0000-0002-2254-4821 (N. Gigante); 0000-0002-4322-769X (A. Montanari)

by Reynolds [8]. The latter in particular is purely tree-shaped, as each branch of the tree is independently expanded and accepted or rejected in a single pass. Thanks to this feature, it has been proven to be efficiently implementable [15] and easily parallelizable [16]. Moreover, its modular structure allowed it to be extended to different logics, in particular LTL with *past operators* [17] and real-time extensions of LTL such as *Timed Propositional Temporal Logic* (TPTL) [18].

BLACK, short for *Bounded LTL sAtisfiability ChecKer*, is a recently developed software tool for satisfiability checking of LTL formulas, based on an incremental SAT encoding of Reynolds' tableau [19]. Given an LTL formula ϕ, the tool encodes in a SAT formula the tableau for ϕ up to depth k, for increasing values of k, until an accepted branch is found or a witness of unsatisfiability is detected. In contrast to similar approaches *à la bounded model checking*, the algorithm underlying BLACK is *complete* even without computing any completeness threshold, thanks to the SAT encoding of the pruning rule of Reynolds'tableau. The modularity of Reynolds' tableau transfers to its SAT encoding. Thanks to this, BLACK is able to support both *future* and *past* temporal operators [20], and formulas interpreted both on *finite* and *infinite* traces, with minimal changes to the underlying encoding. Thanks to the direct relationship between tableau branches and models of the formula, BLACK easily supports the extraction of a model for satisfiable formulas. Experimental evaluations performed against other state-of-the-art tools show BLACK competitive performance and low memory consumption [19].

From an engineering point of view, BLACK has been designed and developed with speed, stability, flexibility and portability as primary goals. It is implemented as a shared library, with a well-defined API, that can be invoked in client applications, with the tool itself being a simple command-line wrapper over the library. As the underlying SAT solver, it supports a variety of backends, such as MathSAT [21], Z3 [22], MiniSAT [23] and CryptoMiniSAT [24]. The tool and the library are multi-platform, with support for Linux, Windows, and macOS, with easy-to-install binary packages available for all platforms.[1]

In the rest of the paper, we further describe the tool and its features, we overview its underlying algorithm and its design and implementation choices. Then, we discuss possible future developments.

2. The SAT encoding of Reynolds tableau

BLACK's algorithm for satisfiability checking is depicted in Fig. 1. At each iteration, for increasing values of k, the encoding of the tableau up to depth k is tested to either find (the encoding of) a suitable accepted branch, or a witness of unsatisfiability (*i.e.*, the encoding of a tableau with only rejected branches). An example tableau for the formula $\mathsf{G}\,\mathsf{F}(p \wedge \mathsf{X}\,\neg p)$ is shown in Fig. 2. The formulas $[\![\phi]\!]^k$, $|\phi|^k$, and $|\phi|^k_T$ encode the tableau up to depth k in different ways. For a complete definition of the encoding we refer the reader to Geatti *et al.* [19, 20]. Here, we state the main properties of such formulas, in order to understand the algorithm in Fig. 1.

Given an LTL formula ϕ, the formula $[\![\phi]\!]^k$ is the k-*unraveling* of ϕ. It represents the expansion of the tableau tree up to depth k. Intuitively, if $[\![\phi]\!]^k$ is unsatisfiable, all the branches of the tableau are rejected by propositional contradictions before depth $k+1$, and vice versa.

[1] BLACK can be downloaded from https://github.com/black-sat/black

```
1: procedure BLACK(φ)
2:     k ← 0
3:     while True do
4:         if [[φ]]^k is UNSAT then
5:             return φ is UNSAT
6:         end if
7:         if |φ|^k is SAT then
8:             return φ is SAT
9:         end if
10:        if |φ|^k_T is UNSAT then
11:            return φ is UNSAT
12:        end if
13:        k ← k + 1
14:    end while
15: end procedure
```

Figure 1: BLACK's main procedure [19]

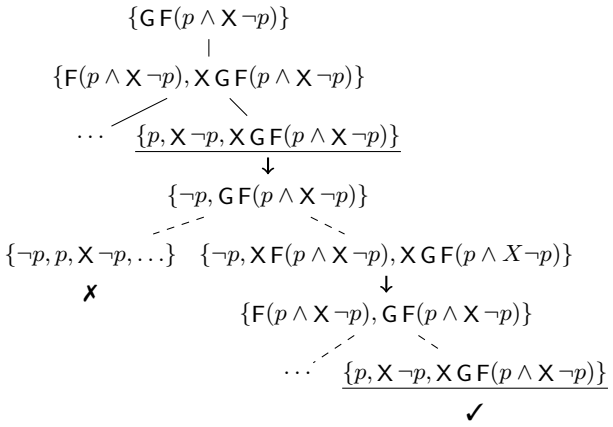

Figure 2: A tableau for the formula $\mathsf{G}\,\mathsf{F}(p \wedge \mathsf{X}\,\neg p)$

Lemma 1. *Let ϕ be an LTL formula. Then, $[\![\phi]\!]^k$ is unsatisfiable if and only if all the branches of the complete tableau for ϕ are crossed by propositional contradictions before depth $k+1$.*

The formula $|\phi|^k$ is called the *base encoding* of ϕ, and includes the k-unraveling conjuncted with additional clauses that represent the acceptance conditions of tableau branches. Hence, if $|\phi|^k$ results satisfiable, we found at least one accepted branch.

Lemma 2. *Let ϕ be an LTL formula. If the complete tableau for ϕ contains an accepted branch of depth $k+1$, then $|\phi|^k$ is satisfiable.*

Lemma 3. *Let ϕ be an LTL formula. If $|\phi|^k$ is satisfiable then the complete tableau for ϕ contains an accepted branch.*

Finally, the $|\phi|^k_T$ formula is called the *termination encoding* of ϕ. It consists in the k-unraveling conjuncted with additional clauses that represent the pruning rule of Reynolds' tableau, *i.e.*, the rule that reject branches that would otherwise be expanded forever trying to fulfill an unfulfillable future request. This is, for instance, the case of the tableau for the formula $\mathsf{G}\neg p \wedge q\mathcal{U}\neg p$. If the formula is unsatisfiable, the formula only contains rejected branches.

Lemma 4. *Let ϕ be an LTL formula. If $|\phi|^k_T$ is unsatisfiable, then the complete tableau for ϕ contains only rejected branches.*

Together, these Lemmata allow us to prove the soundness and completeness of the procedure. See Geatti *et al.* [20] for the full proofs.

Theorem 1 (Soundness and completeness)**.** *Let ϕ be an LTL formula. The BLACK algorithm always terminates and answers* satisfiable *on ϕ if and only if ϕ is satisfiable.*

As shown by Geatti *et al.* [20], the encoding can be easily extended to LTL+Past formulas, *i.e.*, LTL augmented with *past operators*. Furthermore, BLACK has been recently extended to work on formulas interpreted over *finite traces* as well. This addition involves minimal changes to the $|\phi|^k$ formula, in order to not accept looping models.

3. Design and implementation

BLACK has been written in C++17 with the goals of speed, low memory consumption and stability. The latter requirement is particularly important given the types of applications of a formal verification tool: the satisfiability checker must be trustable as much as possible. For this reason, the development has been integrated from the start with a *continuous integration* pipeline and an extensive test suite that achieve 100% code coverage. The result is a pretty solid tool with no signs of memory errors, undefined behaviors and memory leaks.

Integral part of the testing pipeline is the *trace checking* feature. Given a particular option, BLACK can output the model of satisfiable formulas in JSON format. Besides being a useful output format, this allows BLACK itself to parse its own output to check whether the given state sequence is indeed a model of the given formula. Note that this task is much easier than satisfiability checking, and it only involves a simple structural recursion on the formula independent from BLACK's solver. The trace checking is applied to the result of all the over 3300 formulas of the test suite, increasing the confidence in the correctness of the tool.

As already mentioned, the tool is actually implemented as a thin wrapper over a shared library that can be linked by any client application, increasing the flexibility of the tool. The library provides facilities to create and parse LTL formulas, and, most importantly, to manipulate them with a functional-style pattern matching mechanism. This drastically decreases the amount of code required to implement formulas transformations and to implement the SAT encoding itself. Then, the library provides an abstraction layer over the available SAT solvers. This layer hides the differences between the backends, transforms the given formulas, if needed, in *conjunctive normal form*, and invokes the SAT solver. BLACK's solver, that implements the procedure described in Section 2, is implemented on top of these facilities. Thus, besides satisfiability checking, the BLACK library can provide a useful basis for other types of applications that involve the manipulation of LTL formulas or the use of SAT solvers.

4. Conclusions and future developments

We briefly described BLACK, a satisfiability checker for LTL formulas based on the SAT encoding of the one-pass tree-shaped tableau by Reynolds [8].

Many future developments are possible, both in terms of increased performance and additional functionalities. On the performance side, a promising development is the *parallelization* of the solver. Work is in progress to find a suitable way to exploit multiple processing cores. Other optimizations such as the improvement of the CNF generation are possible.

An important additional feature would be the support of the *model checking* task in addition to satisfiability checking. This may concretely consist in the support of a subset of NuXmv [9] modeling language with LTL specifications, with the model checking problem reduced to satisfiability in the standard way. Additional algorithms such as IC3 [25] can be implemented to provide a portfolio of techniques to better suite any given scenario.

Lastly, BLACK can conceivably be augmented to support extensions of LTL. As an example, an extension of Reynolds' tableau exists for the real-time TPTL logic [18], and work is in progress to extend it to the monodic two-variable fragment of first-order LTL as well [26].

References

[1] G. De Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: F. Rossi (Ed.), Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI/AAAI, 2013, pp. 854–860.

[2] G. De Giacomo, M. Y. Vardi, Synthesis for LTL and LDL on finite traces, in: Q. Yang, M. J. Wooldridge (Eds.), Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, AAAI Press, 2015, pp. 1558–1564.

[3] A. P. Sistla, E. M. Clarke, The Complexity of Propositional Linear Temporal Logics, Journal of ACM 32 (1985) 733–749. doi:10.1145/3828.3837.

[4] P. Wolper, Temporal Logic Can Be More Expressive, Information and Control 56 (1983) 72–99. doi:10.1016/S0019-9958(83)80051-5.

[5] M. C. Mayer, C. Limongelli, A. Orlandini, V. Poggioni, Linear temporal logic as an executable semantics for planning languages, Journal of Logic, Language and Information 16 (2007) 63–89. doi:10.1007/s10849-006-9022-1.

[6] K. Y. Rozier, M. Y. Vardi, LTL satisfiability checking, Int. J. Softw. Tools Technol. Transf. 12 (2010) 123–137. URL: https://doi.org/10.1007/s10009-010-0140-3. doi:10.1007/s10009-010-0140-3.

[7] O. Lichtenstein, A. Pnueli, Propositional Temporal Logics: Decidability and Completeness, Logic Journal of the IGPL 8 (2000) 55–85. doi:10.1093/jigpal/8.1.55.

[8] M. Reynolds, A New Rule for LTL Tableaux, in: Proc. of the 7th International Symposium on Games, Automata, Logics and Formal Verification, volume 226 of *EPTCS*, 2016, pp. 287–301. doi:10.4204/EPTCS.226.20.

[9] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, S. Tonetta, The nuXmv Symbolic Model Checker, in: Computer Aided Verification, Springer, 2014, pp. 334–342. doi:10.1007/978-3-319-08867-9_22.

[10] U. Hustadt, B. Konev, TRP++2.0: A temporal resolution prover, in: Proc. of the 19th International Conference on Automated Deduction, volume 2741 of *LNCS*, Springer, 2003, pp. 274–278. doi:10.1007/978-3-540-45085-6_21.

[11] M. Fisher, A normal form for temporal logics and its applications in theorem-proving and execution, Journal of Logic and Computation 7 (1997) 429–456. doi:10.1093/logcom/7.4.429.

[12] M. Fisher, A resolution method for temporal logic, in: J. Mylopoulos, R. Reiter (Eds.), Proceedings of the 12th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1991, pp. 99–104.

[13] J. Li, Y. Yao, G. Pu, L. Zhang, J. He, Aalta: an LTL satisfiability checker over infinite/finite traces, in: S. Cheung, A. Orso, M. D. Storey (Eds.), Proc. of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM, 2014, pp. 731–734. doi:10.1145/2635868.2661669.

[14] S. Schwendimann, A new one-pass tableau calculus for PLTL, in: Proc. of the 7th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, volume 1397 of *LNCS*, Springer, 1998, pp. 277–292. doi:10.1007/3-540-69778-0_28.

[15] M. Bertello, N. Gigante, A. Montanari, M. Reynolds, Leviathan: A new LTL satisfiability checking tool based on a one-pass tree-shaped tableau, in: Proc. of the 25th International

Joint Conference on Artificial Intelligence, IJCAI/AAAI Press, 2016, pp. 950–956.
[16] J. C. McCabe-Dansted, M. Reynolds, A parallel linear temporal logic tableau, in: P. Bouyer, A. Orlandini, P. S. Pietro (Eds.), Proc. of the 8th International Symposium on Games, Automata, Logics and Formal Verification, volume 256 of *EPTCS*, 2017, pp. 166–179.
[17] N. Gigante, A. Montanari, M. Reynolds, A one-pass tree-shaped tableau for LTL+Past, in: Proc. of 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, volume 46 of *EPiC Series in Computing*, 2017, pp. 456–473.
[18] L. Geatti, N. Gigante, A. Montanari, M. Reynolds, One-pass and tree-shaped tableau systems for TPTL and TPTL_b+Past, Inf. Comput. 278 (2021) 104599. URL: https://doi.org/10.1016/j.ic.2020.104599. doi:10.1016/j.ic.2020.104599.
[19] L. Geatti, N. Gigante, A. Montanari, A SAT-Based encoding of the one-pass and tree-shaped tableau system for LTL, in: S. Cerrito, A. Popescu (Eds.), Proceedings of the 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, volume 11714 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 3–20. doi:10.1007/978-3-030-29026-9_1.
[20] L. Geatti, N. Gigante, A. Montanari, G. Venturato, Past matters: Supporting LTL+Past in the BLACK satisfiability checker, in: Proceedings of the 28th International Symposium on Temporal Representation and Reasoning, 2021.
[21] A. Cimatti, A. Griggio, B. J. Schaafsma, R. Sebastiani, The MathSAT5 SMT solver, in: N. Piterman, S. A. Smolka (Eds.), Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, volume 7795 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 93–107. doi:10.1007/978-3-642-36742-7_7.
[22] L. M. de Moura, N. Bjørner, Z3: an efficient SMT solver, in: C. R. Ramakrishnan, J. Rehof (Eds.), Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, volume 4963 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 337–340. doi:10.1007/978-3-540-78800-3_24.
[23] N. Eén, N. Sörensson, An extensible sat-solver, in: Selected Revised Papers of the 6th International Conference on Theory and Applications of Satisfiability Testing, 2003, pp. 502–518. doi:10.1007/978-3-540-24605-3_37.
[24] M. Soos, K. Nohl, C. Castelluccia, Extending SAT solvers to cryptographic problems, in: O. Kullmann (Ed.), Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, volume 5584 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 244–257. URL: https://doi.org/10.1007/978-3-642-02777-2_24. doi:10.1007/978-3-642-02777-2_24.
[25] A. R. Bradley, Sat-based model checking without unrolling, in: R. Jhala, D. A. Schmidt (Eds.), Verification, Model Checking, and Abstract Interpretation - 12th International Conference, VMCAI 2011, Austin, TX, USA, January 23-25, 2011. Proceedings, volume 6538 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 70–87. URL: https://doi.org/10.1007/978-3-642-18275-4_7. doi:10.1007/978-3-642-18275-4_7.
[26] R. Kontchakov, C. Lutz, F. Wolter, M. Zakharyaschev, Temporalising tableaux, Stud Logica 76 (2004) 91–134. doi:10.1023/B:STUD.0000027468.28935.6d.