A New Tableau-based Satisfiability Checker for Linear Temporal Logic
(Extended Abstract)

Matteo Bertello[1], Nicola Gigante[1], Angelo Montanari[1], and Mark Reynolds[2]

[1] University of Udine, Italy
{bertello.matteo,gigante.nicola}@spes.uniud.it
angelo.montanari.uniud.it
[2] University of Western Australia, Australia
mark.reynolds@uwa.edu.au

Abstract Tableaux-based methods were among the first techniques proposed for Linear Temporal Logic satisfiability checking. The earliest tableau for LTL by Wolper worked by constructing a graph whose path represented possible models for the formula, and then searching for an actual model among those paths. Subsequent developments led to the tree-like tableau by Schwendimann, which works by building a structure similar to an actual search tree, which however still has back-edges and needs multiple passes to assess the existence of a model. This paper summarizes the work done on a new tool for LTL satisfiability checking based on a novel tableau method. The new tableau construction, which is very simple and easy to explain, builds an actually tree-shaped structure and it only requires a single pass to decide whether to accept a given branch or not. The implementation has been compared in terms of speed and memory consumption with tools implementing both existing tableau methods and different satisfiability techniques, showing good results despite the simplicity of the underlying algorithm.

1 Introduction

Linear Temporal Logic (LTL) is a modal logic useful to reason about propositions whose truth value depends on a linear and discrete flow of time. Initially introduced in the field of formal methods for the verification of properties of programs and circuit designs [13], it has found applications also in AI, *e.g.*, as a specification language for temporally extended goals in planning problems [3].

The most studied problem regarding LTL is probably *model checking*, *i.e.*, the problem of establishing whether a given temporal structure satisfies an LTL formula. However, *satisfiability checking*, that is, the problem of deciding whether a formula has a satisfying model in the first place, has also received a lot of attention. After being proved to be PSPACE-complete [18], the satisfiability problem for LTL was solved by a number of different methods developed over the years. First of all, LTL satisfiability can be easily reduced to the model checking problem, for which a number of successful techniques exist [5]. Substantial work has

also been devoted to methods based on *temporal resolution*, first pioneered by Cavalli and Fariñas del Cerro, and later refined by Fisher et al. in [7, 8]. Temporal resolution is also at the core of the more recent *labeled superposition* method [19], which proved to be very fast in practice. See [15, 16, 20] for comprehensive experimental comparisons among the tools implementing these techniques.

This paper focuses on tableau-based decision procedures for LTL, which were among the first satisfiability checking methods proposed for it. The first tableau-based method for LTL has been proposed by Wolper [21]. His tableau works by first building a graph-shaped structure, and then performing a number of operations on this graph. Thus, it can be classified as a *graph-shaped* and *multiple-pass* tableau method. An incremental version, which does not require to build the whole graph, was later proposed in [10]. In a subsequent development by Schwendimann [17], a tree-like tableau was proposed which, according to experimental comparisons [9], outperformed the graph-shaped one. The major breakthrough of this new tableau was that of being *single-pass*. While the shape of Schwendimann's tableau is arguably similar to a tree, it is actually still a graph since a number of back-edges have to be maintained. Moreover, the extraction of an actual model from the built tableau is possible, but it requires some work.

Here, we describe an original tool to check LTL satisfiability based on the tree-shaped tableau proposed in [14]. A detailed account of the tool and of relevant experiments can be found in [4]. In contrast to the tableau by Schwendimann, an actual tree is built, and a successful branch directly provides the corresponding satisfying model. Moreover, the tableau rules are very easy to explain and to reason about, but, despite this simplicity, an efficient implementation has shown to offer good average performance on a number of standard benchmarks.

The next sections are organized as follows. Section 2 introduces LTL syntax and semantics, and it quickly describes how previous tableau-based methods behave. Section 3 gives a short account of the new one-pass tree-shaped tableau, and it summarizes the result of experimental comparisons that were reported in [4]. Section 4 outlines possible future developments of the work.

2 Tableau-based methods for LTL

Before illustrating tableau-based decision procedures for LTL, we briefly recap syntax and semantics of the logic. An LTL formula is obtained from a set Σ of proposition letters by possibly applying the usual Boolean connectives and the two temporal operators X (*tomorrow*) and $\mathcal{U}$ (*until*). Formally, LTL formulae ϕ are generated by the following syntax:

$$\phi := p \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \mathsf{X}\,\phi_1 \mid \phi_1\,\mathcal{U}\,\phi_2,$$

where ϕ_1 and ϕ_2 are LTL formulae and $p \in \Sigma$. Standard derived Boolean connectives can also be used, together with logical constants $\bot \equiv p \wedge \neg p$, for $p \in \Sigma$, and $\top \equiv \neg\bot$. Moreover, two derived temporal operators $\mathsf{F}\,\phi \equiv \top\,\mathcal{U}\,\phi$ (*eventually*) and $\mathsf{G}\,\phi \equiv \neg\,\mathsf{F}\,\neg\phi$ (*always*) are defined.

LTL formulae are interpreted over temporal structures. A *temporal structure* is a triple $M = (S, R, g)$, where S is a finite set of states, $R \subseteq S \times S$ is a binary relation, and, for each $s \in S$, $g(s) \subseteq \Sigma$. R is the transition relation, which is assumed to be total, and g is a labeling function, that tells us which proposition letters are true at each state. Given a structure M, we say that an ω-sequence of states $\langle s_0, s_1, s_2, \ldots \rangle$ from S is a *full-path* if and only if, for all $i \geq 0$, $(s_i, s_{i+1}) \in R$. If $\sigma = \langle s_0, s_1, s_2, \ldots \rangle$ is a full-path, then we write σ_i for s_i and $\sigma_{\geq i}$ for the infinite suffix $\langle s_i, s_{i+1}, \ldots \rangle$ (also a full-path). We write $M, \sigma \models \varphi$ if and only if the LTL formula φ is true on the full-path σ in the structure M, which is defined by induction on the structural complexity of the formula:

- $M, \sigma \models p$ iff $p \in g(\sigma_0)$, for $p \in \Sigma$,
- $M, \sigma \models \neg\varphi$ iff $M, \sigma \not\models \varphi$
- $M, \sigma \models \varphi_1 \vee \varphi_2$ iff $M, \sigma \models \varphi_1$ or $M, \sigma \models \varphi_2$
- $M, \sigma \models \mathsf{X}\varphi$ iff $M, \sigma_{\geq 1} \models \varphi$
- $M, \sigma \models \varphi_1 \,\mathcal{U}\, \varphi_2$ iff there is some $i \geq 0$ such that $M, \sigma_{\geq i} \models \varphi_2$ and for all j, with $0 \leq j < i$, it holds that $M, \sigma_{\geq j} \models \varphi_1$

Most existing tableau methods for LTL, including the one described in this paper, make use of the observation that any LTL formula can be rewritten by splitting it into two parts, one prescribing something about the current state, and one talking about the next state. In particular, this is true for formulae whose outermost operator is a temporal one:

$$\alpha \,\mathcal{U}\, \beta \equiv \beta \vee (\alpha \wedge \mathsf{X}(\alpha \,\mathcal{U}\, \beta)); \quad \mathsf{F}\,\beta \equiv \beta \vee \mathsf{X}\mathsf{F}\,\beta; \quad \mathsf{G}\,\alpha \equiv \alpha \wedge \mathsf{X}\mathsf{G}\,\alpha$$

Here, the formulae $\mathsf{X}(\alpha \,\mathcal{U}\, \beta)$ and $\mathsf{X}\mathsf{F}\,\beta$ are called X-*eventualities*, *i.e.*, pending requests that has to be eventually fulfilled somewhere in the future, but that can for the moment be postponed. Note that $\mathsf{G}\,\alpha$ does not lead to an eventuality, since it has to be fulfilled immediately in any case.

The first tableau by Wolper starts by building a graph where each node is labeled by a set of *locally consistent* formulae belonging to the *closure* of ϕ. Each node collects the relevant formulae that are true at a specific state. Nodes u and v are connected by an edge if v can be a successor state of u according to the semantics of the logic. The equivalences shown above are useful in determining these edges. Paths in the graph represent potential models, which are consistent when we look at the single transitions. Finding an actual model then consists of searching for a path that fulfills all the pending *eventualities*. This approach requires the construction of the entire graph before the actual search, which means that an exponential amount of memory was required, which is not optimal with regards to the complexity of the problem, which is PSPACE-complete. An incremental version of this tableau, which does not require to build the whole graph beforehand, thus achieving the polynomial space lower bound, was introduced by Kesten et al. [10]. Later, a tableau *à la* Wolper was provided by Lichtenstein and Pnueli [12] to also handle *past* temporal operators.

Marking a significant development, a new *one-pass* tableau for LTL was introduced by Schwendimann [17]. His tableau method works by building a tree-like

structure, more similar to a search tree. Since the fulfillment of the eventualities in a branch are checked during the construction, subsequent passes are not needed. While the tableau structure resembles a tree, it is actually still a sort of cyclic graph (called *loop tree* in the original presentation), and the searches performed on separate branches are not completely independent. Although the complexity of the decision procedure based on Schwendimann's tableau is worse than the one by Wolper, since it requires *doubly* exponential time in the worst case, experimental comparisons [9] have shown that in practice this method outperforms previous tableaux, in some cases by large margins.

3 A new one-pass and tree-shaped tableau for LTL

A new *one-pass* and *tree-shaped* tableau for LTL was proposed in [14]. A satisfiability checker based on it[3], written in C++, and a detailed comparison with previous tableaux as well as with tools implementing different satisfiability checking techniques are given in [4].

In contrast to Schwendimann's one, the new tableau works by building an actual tree. In the tableau tree for a formula ϕ, each node is labeled by a set Γ of *formulae*. For each $\psi \in \Gamma$, ψ is a subformula of ϕ or is a formula of the form $\mathsf{X}\,\psi'$, where ψ' is a subformula of ϕ. The tree construction starts from the root being labeled by $\{\phi\}$. The tree is then built by applying a sequence of *rules* which, for what concerns Boolean connectives, resembles the classical tableau for propositional logic, with disjunctions causing a node to fork into different *branches*. Temporal formulae are instead expanded using the already mentioned equivalences from Section 2. A node where no further expansion is possible is said to have a *poised* label, and it represents what is true at the current state in the resulting model. In a poised label, only literals or *tomorrow* temporal operators are present at top level. A Step rule is then used to advance the branch to the next temporal state, by creating a new node whose label includes a formula α for each formula of the form $\mathsf{X}\,\alpha$ found in the previous node. If contradictory literals are ever introduced into a label, the branch is rejected (✗), while if a Step rule results into an empty label, the branch is accepted (✓), and a model can be extracted from all the nodes preceding the application of the Step rule from there to the root of the tree.

These rules alone, however, are insufficient to handle formulae satisfiable by infinite models only as well as formulae that are unsatisfiable not because of propositional contradictions but because of unsatisfiable eventualities. To handle these cases, the following two rules are applied before the Step one. The first, the Loop rule, accepts a branch each time we find ourselves on a label that have already been expanded before, and all the eventualities have been fulfilled in between, meaning that the node needs not to be further expanded because the repeating part of an ultimately periodic model has been found. The second, the Prune rule, handles unsatisfiable formulae like, for instance, $\mathsf{G}\,\neg p \wedge q\,\mathcal{U}\,p$, by

[3] `http://www.github.com/corralx/leviathan`

ensuring that the tableau expansion does not hang into the infinite expansion of a branch that would not be able to fulfill the remaining pending eventualities. The latter is definitely the most sophisticated rule of the tableau system. One of its distinctive features is that it needs to go through three different nodes with the same label before crossing the branch.

A complete description of the rules can be found in [14], but it can already be noted how simple the whole construction is. The space and running time worst cases are the same as those of the tableau system by Schwendimann, but the rules and the bookkeeping required to apply them is simpler and can be implemented in an efficient way. The result is an implementation that, despite its simplicity, has good performance on average both in terms of speed and memory consumption on a number of standard benchmarks [4].

4 Conclusions and Future Work

In this extended abstract, we described a new one-pass and tree-shaped tableau for LTL which is very simple to state and to reason about and can be implemented in an efficient way, showing good performance when compared with previous tableau-based systems. Simplicity may be regarded as its major advantage, that we plan to exploit in future developments. For example, we expect that its simple search procedure can be augmented with advanced search heuristics like clause-learning techniques used in propositional SAT solvers. SAT and SMT technologies can also be exploited in order to improve performance when dealing with temporal formulae that sport large propositional parts.

Such a simple tableau can also be viewed as a useful tool to reason about theoretical properties of LTL and its extensions. For instance, extending its rules to support a parametric X^n operator, with n represented succinctly, appears to be straightforward, and immediately results into an optimal decision procedure for this simple EXPSPACE extension of LTL. In a similar way, we plan to investigate the possibility of implementing other LTL extensions on top of this framework, such as logics that feature metric variants of the *until* operator [2], past operators with forgettable past [11], freeze quantifiers [1], finite models [6], and others.

References

[1] R. Alur and T. A. Henzinger. "A Really Temporal Logic." In: *Journal of the ACM* 41.1 (1994), pp. 181–204.

[2] R. Alur and T. A. Henzinger. "Real-Time Logics: Complexity and Expressiveness." In: *Information and Computation* 104 (1993), pp. 35–77.

[3] F. Bacchus and F. Kabanza. "Planning for Temporally Extended Goals." In: *Annals of Mathematics and Artificial Intelligence* 22 (1998), pp. 5–27.

[4] M. Bertello, N. Gigante, A. Montanari, and M. Reynolds. "Leviathan: A New LTL Satisfiability Checking Tool Based on a One-Pass Tree-Shaped Tableau." In: *Proc. of the 25th Int. Joint Conference on Artificial Intelligence*. 2016.

[5] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.

[6] G. De Giacomo and M. Y. Vardi. "Linear Temporal Logic and Linear Dynamic Logic on Finite Traces." In: *Proc. of the 23rd Int. Joint Conference on Artificial Intelligence*. 2013.

[7] M. Fisher. "A Normal Form for Temporal Logics and its Applications in Theorem-Proving and Execution." In: *Journal of Logic and Computation* 7.4 (1997), pp. 429–456.

[8] M. Fisher, C. Dixon, and M. Peim. "Clausal Temporal Resolution." In: *ACM Transactions on Computational Logic* 2.1 (2001), pp. 12–56.

[9] V. Goranko, A. Kyrilov, and D. Shkatov. "Tableau Tool for Testing Satisfiability in LTL: Implementation and Experimental Analysis." In: *Electronic Notes in Theoretical Computer Science* 262 (2010), pp. 113–125.

[10] Y. Kesten, Z. Manna, H. McGuire, and A. Pnueli. "A Decision Algorithm for Full Propositional Temporal Logic." In: *Proc. of the 5th Int. Conference on Computer Aided Verification*. 1993, pp. 97–109.

[11] F. Laroussinie, N. Markey, and P. Schnoebelen. "Temporal Logic with Forgettable Past." In: *Proc. of the 17th IEEE Symposium on Logic in Computer Science*. 2002, pp. 383–392.

[12] O. Lichtenstein and A. Pnueli. "Propositional Temporal Logics: Decidability and Completeness." In: *Logic Journal of the IGPL* 8.1 (2000), pp. 55–85.

[13] A. Pnueli. "The Temporal Logic of Programs." In: *Proc. of the 18th Annual Symposium on Foundations of Computer Science*. 1977, pp. 46–57.

[14] M. Reynolds. "A new rule for LTL tableaux." In: *Proc. of the 7th Int. Symposium on Games, Automata, Logics and Formal Verification*. 2016.

[15] K. Y. Rozier and M. Y. Vardi. "LTL Satisfiability Checking." In: *International Journal on Software Tools for Technology Transfer* 12.2 (2010), pp. 123–137.

[16] V. Schuppan and L. Darmawan. "Evaluating LTL Satisfiability Solvers." In: *Proc. of the 9th Int. Symposium Automated Technology for Verification and Analysis*. 2011, pp. 397–413.

[17] S. Schwendimann. "A New One-Pass Tableau Calculus for PLTL." In: *Proc. of the 4th Int. Conference on Automated Reasoning with Analytic Tableaux and Related Methods*. 1998, pp. 277–292.

[18] A. P. Sistla and E. M. Clarke. "The Complexity of Propositional Linear Temporal Logics." In: *Journal of the ACM* 32.3 (1985), pp. 733–749.

[19] M. Suda and C. Weidenbach. "A PLTL-Prover Based on Labelled Superposition with Partial Model Guidance." In: *Proc. of the 6th Int. Joint Conference on Automated Reasoning*. 2012, pp. 537–543.

[20] M. Y. Vardi, Li J, L. Zhang, G. Pu, and J. He. "LTL Satisfiability Checking Revisited." In: *Proc. of the 20th Int. Symposium on Temporal Representation and Reasoning*. 2013, pp. 91–98.

[21] P. Wolper. "The Tableau Method for Temporal Logic: An Overview." In: *Logique et Analyse* 28 (1985).