The Tableau-based Theorem Prover $_3T^AP$
Version 4.0

Bernhard Beckert, Reiner Hähnle, Peter Oel, Martin Sulzmann

University of Karlsruhe
Institute for Logic, Complexity and Deduction Systems
Am Fasanengarten 5, 76128 Karlsruhe, Germany
{beckert,haehnle,oel,sulz}@ira.uka.de
http://i12www.ira.uka.de/~threetap/

Overview

$_3T^AP$ is a tableau-based theorem prover for many-valued first-order logics with sorts (in the two-valued version with equality); it is implemented in Prolog.

This paper gives an overview of the system with a special focus on the new features of $_3T^AP$ Version 4.0, including: efficient completion-based equality reasoning, methods for handling redundant axiom sets, utilization of pragmatic information contained in axioms to rearrange the search space, and a graphical user interface for controlling $_3T^AP$ and visualizing its output.

$_3T^AP$ has been developed at the University of Karlsruhe. In 1989 the project started in cooperation with the Institute for Knowledge Based Systems of IBM Germany. Since 1992 the system is maintained and improved as part of a new project at the University of Karlsruhe funded by the Deutsche Forschungsgemeinschaft (DFG).

Supported Logic(s): Specification, Syntax, and Semantics

$_3T^AP$ is able to handle full first-order logics with any finite number of truth values. Hierarchical (tree-shaped) sorts attached to terms are supported. In the two-valued case, special handling of the equality predicate is provided. Currently, versions for classical first-order logic, for a certain three-valued first-order logic [18] and for a seven-valued propositional logic [11] are specified. It is possible either to prove a theorem or to try to check the consistency of the axioms.

The user specifies the input as a set of axioms and theorems contained in a knowledge base file; the formulae do not have to be in any normal form. "if-then" and "if-then-else" connectives can occur in the input to rearrange the search space such as to reflect their particular pragmatics, that differs from material implication. Furthermore, an ordering on constant and function symbols can be specified in the knowledge base.

Problems from the TPTP problem library [20] can be directly loaded. They are automatically converted into $_3T^AP$ format.

The Calculus

$_3T^AP$'s calculus is based on free variable semantic tableaux [8, 15] (with Skolem functions). Using free variable quantifier rules is crucial for efficient implementation—even more if equality has to be handled. They reduce the number of possibilities to proceed at each step in the construction of a tableau proof and thus the size of the search space. When universal quantifier rules are applied, a new free variable is substituted for the quantified variable, instead of replacing it by a ground term, that has to be "guessed". Free variables can later be instantiated "on demand", when a tableau branch is closed or an equality is applied to expand a branch.

For replacing an existentially quantified variable by a Skolem term, $_3T^AP$ uses the following rule: The term needs not contain all the free variables on the current branch; rather it suffices to include just the free variables occurring in the quantified formula (this has been proven to be sound in [12]). In addition, the same function symbol is used for skolemizing formulae that are identical up to variable renaming [3]. This more subtle rule for existential quantifiers leads to an easier implementation and can yield shorter proofs.

To avoid unnecessary instantiations, two kinds of free variables are distinguished, namely *universal* and *rigid* variables. Informally speaking, a variable is considered to be universal in a formula ϕ, if a copy of ϕ could be deduced and added to the current branch without causing any branching of the tableau. Other free variables are considered to be rigid. Free variables marked as universal do not become instantiated when used in a branch closure or equality application. This feature allows $_3T^AP$ to find shorter proofs.

A naïve approach to theorem proving in many-valued logic would build a separate tableau for each non-designated truth value in order to refute a formula. $_3T^AP$ uses the concept of truth value sets-as-signs introduced in [9] (see [6, 10] for details). As a consequence, $_3T^AP$ needs to build merely one tableau even in the many-valued case.

Another feature of $_3T^AP$ is the generation of local lemmata [7, 13]: If a tableau rule generates branch extensions with common models, then lemmata can be added to extensions in order to make them logically disjoint. In the two-valued case only disjunctive rules ("β-rules" in Smullyan's [19] terminology) are of this type. Its extensions β_1 and β_2 resulting from an application to, say, $\beta_1 \vee \beta_2$ have the models satisfying both β_1 and β_2 in common. Thus, if a local lemma $\neg\beta_1$ is added, the result are logically disjoint extensions β_1 and $\beta_2 \wedge \neg\beta_1$ (another possibility is to add $\neg\beta_2$ to the first extension).

$_3T^AP$ uses a depth-first strategy to search for proofs; it proceeds by closing individual branches of a tableau one after the other. When a substitution is found that closes a branch, it is applied (to the whole tableau), and then the prover tries to close the next branch. If, later on, a branch cannot be closed (observing a limit on the number of copies of universally quantified formulae that may be used on each branch or a limit on the length of branches), backtracking is initiated and other closing substitutions are searched for. Different closing substitutions for a single branch are the only choice points in $_3T^AP$'s proof procedure; all other indeterminisms in semantic tableaux, like choosing the next formula to be expanded, are resolved using fair (deterministic) heuristics and strategies.[1]

Equality Handling

A special background reasoner is used for handling equality in $_3T^AP$: Mixed E-unification problems are extracted from tableau branches and passed on to the background reasoner, that employs the completion-based method from [1] to solve mixed E-unification problems and, thus, to close tableau branches.

The equality reasoner can be invoked multiply during the construction of a single tableau branch. After a futile try to find a unifier, the data computed by the background reasoner is reused for later calls, in particular it is reused for different extensions of the branch. We call this feature of $_3T^AP$ incremental equality reasoning [4].

Search Space Restrictions

One possibility to restrict the search space is to avoid putting redundant axioms on the tableau in the first place. This is particularly useful with huge axiomatizations where only a small subset of the axioms is actually needed to prove a given theorem. Only formulae that potentially take part in closing a tableau branch are fetched from the knowledge base.

To decide which axioms are redundant and which might be needed to close a branch, $_3T^AP$ employs a method that is an extension of the well known technique of computing links (connections) between atomic subformulae: whether a formula is linked to an atom on the branch depends on the equalities both on the branch and in the knowledge base; in addition, sorts have to be taken into concern.

Another method for restricting the search space is pruning tableau branches: If a tableau rule application generates several subbranches and in the following one of the subbranches can be closed without using any formula created by that rule application, then the other subbranches could be closed the same way and can, thus, be pruned (removed). This technique is essentially what has been called *condensing* in [14], although we control it differently.

User Interaction

Apart from commands given in the Prolog shell, $_3T^AP$ can be controlled via a graphical user interface (GUI) based on X-Windows. The GUI allows the user to

[1] The minimalistic tableau-based theorem prover leanT^AP [5], that consists of only 15 lines of Prolog code, is an implementation of the same basic search strategy. leanT^AP is, due to the elimination of all overhaed, quite efficient, and easy to understand. On the other hand, it is missing most of the extensions of the calculus and additional features built into $_3T^AP$, like equality handling, powerful heuristics for rearranging the search space, a graphical user interface, etc.

load and compile (pre-process) knowledge bases, to change all parameters and switches, to specify the theorem to be proven, and to start the prover. During the proof search the tableau tree is displayed and continually updated. The user can define spy points to stop the prover, for example, whenever a branch is closed or after a certain number of tableau rule applications. When the prover stops, the user can navigate through the tableau to inspect the (partial) proof that has been constructed.

Integration of Automated and Tactical Theorem Proving

The aim of integrating tactical and automated theorem provers is pursued in a joint project with a research group in tactical theorem proving for formal software verification. A system that integrates the respective latest versions of $_3T^AP$ and the *Karlsruhe Interactive Verifier* (KIV) [16, 17] has been built. Here, $_3T^AP$ is used to prove problems from two-valued first-order logic with equality, that have been generated by KIV. The main issues that have to be dealt with in these problems are their redundancy and their size (up to one thousand axioms). Some of $_3T^AP$'s features such as sorts, orderings, incremental completion-based equality reasoning, if-then connectives, and restricted formula fetching have been developed to cope with such problems.

Availability

$_3T^AP$ is implemented in SICStus Prolog with a small part of portable C. Parts of $_3T^AP$'s compiler module are written using the Unix tools Lex and Yacc (resp. Flex and Bison). The design is as modular as possible; therefore, it is not too difficult to port $_3T^AP$ to other architectures and to add new features.

$_3T^AP$ is available via the *World Wide Web* and via anonymous ftp. To obtain further information or to download the source code and the user's manual [2], use the $_3T^AP$ home page `http://i12www.ira.uka.de/~threetap/`. Alternatively, connect via anonymous ftp to `sonja.ira.uka.de` (`129.13.31.3`) and change to the directory `pub/threetap`.

References

1. Bernhard Beckert. A completion-based method for mixed universal and rigid E-unification. In A. Bundy, editor, *Proceedings, 12th International Conference on Automated Deduction (CADE), Nancy, France*, LNCS 814, pages 678–692. Springer, 1994.
2. Bernhard Beckert, Reiner Hähnle, Karla Geiß, Peter Oel, Christian Pape, and Martin Sulzmann. The many-valued tableau-based theorem prover $_3T^AP$, version 4.0. Interner Bericht 3/96, Universität Karlsruhe, Fakultät für Informatik, 1996.
3. Bernhard Beckert, Reiner Hähnle, and Peter H. Schmitt. The even more liberalized δ-rule in free variable semantic tableaux. In Georg Gottlob, Alexander Leitsch, and Daniele Mundici, editors, *Proceedings, 3rd Kurt Gödel Colloquium (KGC), Brno, Czech Republic*, LNCS 713, pages 108–119. Springer, 1993.

4. Bernhard Beckert and Christian Pape. Incremental theory reasoning methods for semantic tableaux. In *Proceedings, 5th Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Palermo, Italy*, LNCS. Springer, 1996.
5. Bernhard Beckert and Joachim Posegga. leanTAP: Lean tableau-based deduction. *Journal of Automated Reasoning*, 15(3):339–358, 1995.
6. Marcello D'Agostino, Dov Gabbay, Reiner Hähnle, and Joachim Posegga, editors. *Handbook of Tableau Methods*. Kluwer, Dordrecht, 1996. To appear.
7. Marcello D'Agostino and Marco Mondadori. The taming of the cut. *Journal of Logic and Computation*, 4(3), 1994.
8. Melvin C. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, second edition, 1996.
9. Reiner Hähnle. Towards an efficient tableau proof procedure for multiple-valued logics. In *Proceedings, Workshop on Computer Science Logic, Heidelberg, Germany*, LNCS 533, pages 248–260. Springer, 1990.
10. Reiner Hähnle. *Automated Deduction in Multiple-valued Logics*, volume 10 of *International Series of Monographs on Computer Science*. Oxford University Press, 1994.
11. Reiner Hähnle and Werner Kernig. Verification of switch level designs with many-valued logic. In A. Voronkov, editor, *Proceedings, 4th International Conference on Logic Programming and Automated Reasoning (LPAR), St. Petersburg, Russia*, LNCS 698, pages 158–169. Springer, 1993.
12. Reiner Hähnle and Peter H. Schmitt. The liberalized δ-rule in free variable semantic tableaux. *Journal of Automated Reasoning*, 13(2):211–222, 1994.
13. Reinhold Letz, Klaus Mayr, and C. Goller. Controlled integration of the cut rule into connection tableau calculi. *Journal of Automated Reasoning*, 13(3):297–338, December 1994.
14. F. Oppacher and E. Suen. HARP: A tableau-based theorem prover. *Journal of Automated Reasoning*, 4:69–100, 1988.
15. Steve Reeves. Semantic tableaux as a framework for automated theorem-proving. In C. S. Mellish and J. Hallam, editors, *Advances in Artificial Intelligence (Proceedings of AISB-87)*, pages 125–139. Wiley, 1987.
16. Wolfgang Reif. The KIV-system: Systematic construction of verified software. In *Proceedings, 11th International Conference on Automated Deduction (CADE), Saratoga Springs, NY, USA*, LNCS 607, pages 253–267. Springer, 1992.
17. Wolfgang Reif, Gerhard Schellhorn, and Kurt Stenzel. Interactive correctness proofs for software modules using KIV. In *Proceedings, 10th Annual Conference on Computer Assurance (COMPASS), Washington, USA*, pages 151–162, 1994.
18. Peter H. Schmitt. Computational aspects of three-valued logic. In J. H. Siekmann, editor, *Proceedings, 8th International Conference on Automated Deduction (CADE)*, LNCS, pages 190–198. Springer, 1986.
19. Raymond M. Smullyan. *First-Order Logic*. Dover Publications, New York, second corrected edition, 1995. First published 1968 by Springer-Verlag.
20. Geoff Sutcliffe, Christian Suttner, and Theodor Yemenis. The TPTP problem library. In A. Bundy, editor, *Proceedings, 12th International Conference on Automated Deduction (CADE), Nancy, France*, LNCS 814, pages 708–722. Springer, 1994. Current version available on the *World Wide Web* at the URL `http://www.cs.jcu.edu.au/ftp/users/GSutcliffe/TPTP.HTML`.

This article was processed using the LaTeX macro package with LLNCS style