Tableaux-based Theorem Provers – Systems and Implementations –

J. Schumann, Editor

Institut für Informatik
Technische Universität München
80290 München
email: schumann@informatik.tu-muenchen.de

Abstract

The following list of tableaux-based theorem provers was assembled in the Spring and Summer 1993 as the result of a wide-spread enquiry via e-mail. It is intended to provide a short overview of the field and existing implementations. For each system, a short description is given. Additionally, useful information about the system is presented in tabular form. This includes the type of logic which can be handled by the system (input), the implementation language, hardware and operating systems requirements (implementation). Most of the systems are available as binaries or as sources with documentation and can be obtained via anonymous ftp or upon request. The descriptions and further information have been submitted by the individuals whose names are given as contact address (except for 12, 14, 21 which have been formulated by the editor on the basis of submitted texts). The provers are ordered alphabetically by their name (or the author's name).

1 ${}_3T^AP$

"${}_3T^AP$ [23, 22, 8] has been developed at the University of Karlsruhe. Despite of its name, ${}_3T^AP$ is able to deal with classical – i.e. two-valued – first-order predicate logic as well as with any finite-valued first-order logic, provided the semantics is specified by truth-tables. Currently implemented versions are working for two-valued and for a certain three-valued first-order predicate logic, which is a variant of the strong Kleene logic. The multiple-valued version implements the concept of generalized signs. ${}_3T^AP$ is able to deal with equality, which is treated as a two-valued predicate in the multiple-valued case."

Input:	arbitrary many-valued first order logics; classical first order logic with equality
Implementation: Availability:	sequential, Prolog source code and documentation available
Contact Address:	R. Hähnle email: reiner@ira.uka.de B. Beckert email: beckert@ira.uka.de

2 Bluegum

"*Bluegum* is a tableaux-based theorem prover for first-order predicate logic in clausal form. Bluegum uses a calculus based on model elimination [28], and employs a variety of inference restriction techniques and heuristics to prune the search space [48]. The proof procedure uses depth-first iterative-deepening to guarantee completeness, and a variety of techniques inspired by *links* to improve the overall efficiency [50].

Bluegum is implemented as a set of C++ objects, and makes extensive use of advanced object-oriented techniques such as inheritance, polymorphism, dynamic binding, and parameterized-typing to engineer an efficient platform for experimentation. Facilities are provided to generate both text and graphical representations of the proof, and produce a variety of proof statistics. The current system can handle propositional logic, and work on a first-order system is well-advanced."

Input:	first order predicate logic in clausal form
Implementation: Availability:	sequential, C++ (AT&T v3) on SunOS
Contact Address:	K. Wallace email: kevin@cs.newcastle.edu.au

3 Cassandra

"Cassandra [19, 20, 21] is a simple prover-generation system, allowing the user to design and execute tableau prover prototypes for a variety of applications, including Classical, Modal, Temporal Logics and word problems for simple grammars.

While the system lacks any real proof search control mechanism, its data structures are very efficient, allowing it to work on proofs of moderate difficulty.

The system does not yet offer reasoning with equality or arbitrary functions. A future version is planned (to be written in LISP), offering more extensive proof control and more reasoning facilities for functions."

Input:	full first order predicate logic, modal and temporal logics, some grammars
Implementation: Availability:	sequential, C sources and documentation available
Contact Address:	M. Grundy email: markg@arp.anu.edu.au

4 Deep Thought (DT)

"Deep Thought (DT) [16] is an automated free variable analytic tableau prover for full first order logic without equality. There is also a version available for a certain three-valued logic. The next version will include the possibility to load a user-defined multiple valued logic and the corresponding rules. DT uses the liberalized δ–rule described in [24]. Static indexing has been implemented for axiom selection. Tableau expansion is controlled by a sophisticated strategy. The Amiga version includes a graphical user interface. DT's protocol may be saved as LaTeX source, a TreeTeX representation of the proof may be generated (due to the limitations of TreeTeX this feature is only useful for small proofs). DT has been mostly inspired by ${}_3TAP$ (see Section 1)."

Input:	first order predicate logic (three-valued logic)
Implementation: Availability:	sequential, C on Commodore Amiga (Sun OS in preparation) source code, binaries and documentation available
Contact Address:	S. Gerberding email: gerberding@inferenzsysteme.informatik.th-darmstadt.de

5 FAUST

"There are three versions of the FAUST-Prover [25, 40, 41]: the first one is based on a Sequent Calculus which is extended by a special form of unification in order to compute instances of Gamma-rules. The second version of FAUST is based on structures called tableau graphs which resemble matrices of connection calculi. The third version of FAUST will be based on Shannon Graphs and also on Binary Decision Diagrams. The FAUST-Prover is used in the domain of hardware-verification which requires in general higher-order logic. Therefore the prover has been integrated in

a higher-order proof checker called HOL which is available by anonymous ftp."

Input:	first order predicate logic
Implementation:	sequential, SML of New Jersey (Sun OS)
Availability:	system available
Contact Address:	K. Schneider, Th. Kropf email: schneide@ira.uka.de *or* kropf@ira.uka.de

6 Forest Theorem Prover

"The UK Alvey Forest project theorem prover [4, 10] was developed for proving properties of Real Time Requirements Specifications. The basic specification logic is a first order modal logic of action, with deontic operators for normative behaviour, constant domain and non-rigid designators. The theorem provers have been tableau based, with an iterative linear search algorithm. There is no publically available version now, but the work is described in the cited references, and related mixed-logic developments are under way."

Input:	first order modal logic of action (see above)
Implementation:	sequential, PROLOG
Availability:	
Contact Address:	J. Cunningham email: rjc@doc.ic.ac.uk

7 HimMLKreuz

"HimMLKreuz [18, 17] is a theorem prover for first order logic (without equality) based on Binary Decision Diagrams (BDD) with a strategy of proof search through control of information in the sense of Shannon. HimMLKreuz incorporates analogues of linear resolution, the pure literal rule and Billon's instance subtraction technique which generalizes cg-resolution."

Input:	first order predicate logic
Implementation:	sequential, HimML (Standard ML + Sets)
Availability:	proprietary system (Bull S.A.)
Contact Address:	J. Goubault email: Jean.Goubault@frcl.bull.fr

8 iTAB

"iTAB is a prover based on the ILFA-library which has been developed by IBM Germany. iTAB has been integrated into the ILF deduction experimental shell

[12]."

Input:	First order predicate logic
Implementation:	sequential, C
Availability:	source code available, ILFA-library is proprietary by IBM Germany
Contact Address:	A. Wolf email: wolf@informatik.hu-berlin.de

9 linseq

"Linseq [47] is a tableaux-based prover for full first-order propositional linear logic, an undecidable logic. Linseq uses several modifications to standard rules, suited for tableaux-style theorem proving, and special strategies. The main purpose of implementing linseq was comparing tableaux-style and resolution-style theorem proving for linear logic (LL). The resolution-style prover "linres" for LL is also implemented. In our experiments the efficiency of linseq and linres was roughly comparable, except for the nonexponential sublanguage of LL, where linres was much better than linseq."

Input:	full first-order propositional linear logic
Implementation:	sequential, C and Scheme
Availability:	source code and documentation available ftp: ftp.cs.chalmers.se:/pub/provers/misc/linseq.tar.Z
Contact Address:	T. Tammet email: tammet@cs.chalmers.se

10 Meteor

"Meteor [3, 2, 1] compiles clauses into a data structure that is then "interpreted" by a sequential (any UNIX machine) inference engine or parallel (Butterfly TC2000, Network of Workstations) engines. The underlying inference mechanism is Model Elimination, but Meteor also employs caching and lemmaizing (with demodulation) as redundancy reducing mechanisms. These methods, combined with several different depth measures used in conjunction with an iterative deepening search, permit Meteor to discover proofs of hard theorems. "

Input:	first order predicate logic in clausal form
Implementation:	sequential and OR-parallel, C (UNIX on sun, DEC)
Availability:	binaries available
Contact Address:	O.L. Astrachan email: ola@cs.duke.edu

11 [W. Neitz]

"We present a theorem prover which performs a selective backtracking strategy. This prover [30, 31] is based on the idea of the Prolog Technology Theorem Prover (PTTP) introduced by Stickel (see Section 19).

First order formulas are transformed into a set of Prolog clauses which performs when executed the complete Model Elimination procedure with an iterative-deepening-search strategy and selective backtracking.

The information needed for selective backtracking is provided by a unification algorithm also coded in Prolog."

Input:	First order predicate logic
Implementation: Availability:	sequential, Prolog (Quintus Prolog)
Contact Address:	W. Neitz email: wneitz@informatik.uni-leipzig.de

12 [W. Ophelders]

This tableau-based theorem prover [32, 33, 14, 13] is able to handle formulae of first order predicate logic with function symbols but without equality. The sound and complete prover uses unification without first to skolemize, a technique which additionally takes care of restrictions for the construction of new terms. Such restrictions contain the occurs-check and further constraints in a given list. Extensions to intuitionistic logic and other non-classical logics are discussed in [32]. Using Pelletier's list of 75 problems for testing automated theorem provers a comparison is made with both resolution based provers (PCPROVE, Otter, Satchmo) and tableau-based provers (Fitting, Reeves).

Input:	
Implementation: Availability:	sequential, LPA-Prolog source code and documentation available
Contact Address:	W. Ophelders email: Ophelders@Facburo.FEW.EUR.NL H. de Swart email: swart@kub.nl

13 PartabX

"None of the following list of theorem provers (with the generic name "PartabX") require clausal form. These provers are implemented in Strand and run on distributed or parallel architectures. The development rationale was to parallelise the tableau method and its derivatives in order to utilise a distributed network of machines. These theorem provers are lumped together under the name PartabX (Parallel tableaux), however, they are extremely different and include dummy/free variable, exhaustive tableau, and connection method based systems. For this reason they are categorised below. The overriding necessity was to develop all systems in a uniform environment for comparative purposes. All tableau systems use OR-parallelism. Connection method systems apply speculative parallelism except for the final one being developed currently. Strand itself operates using a condition-synchronised AND-parallelism regime. All systems will obviously all run sequentially if necessary. All tableau systems incorporate a number of heuristics (uniformly), many of which are outlined in [34]. Additionally, there is an (almost always useful) option

of discarding used sentences."

Name	Input	Description
prop	propositional logic	Basic tableau system
exfop	function-free FOL	First order exhaustive method
dfop	—"—	FOL dummy/free-variable method
dbdfop	—"—	FOL dummy/free-variable method tableau system adaptation. Multiple conclusions (interrogations) may be tested against a single set of premises (the database) for validity to simulate database interrogation.
*prop	prop. modal logics K, K4, D, D4, T, S4	Propositional modal tableau systems with parallel evaluation of possible worlds.
cp01,cp02 cp03	propositional logic	Implementation of Bibel's Connection Method
con	propositional logic	Basic path elaboration through a connection method clausal matrix

Input:	see above (clausal form not required)
Implementation:	parallel, Strand
Availability:	
Contact Address:	R. Johnson email: robj@sun.com.mmu.ac.uk

14 Parthenon

"Parthenon (PARallel THEorem prover for NON-Horn clauses) [9] is an OR-parallel theorem prover for first order predicate logic. The underlying proof calculus is a variant of Model Elimination. Parthenon exploits OR-parallelism by dynamically executing independent parts of the search tree on different processors, using a computational model similar to the SRI-Model for OR-parallel Prolog. PARTHENON, being the first implementation of an OR-parallel theorem prover, runs on various multi-processors, e.g., an Encore Multimax (16 processors, 32 MBytes shared memory), and an IBM RP3 (64 ROMP processors)."

Input:	first order predicate logic in clausal form
Implementation:	OR-parallel, C (with C-threads) under mach
Availability:	upon request
Contact Address:	E.M. Clarke email: Edmund_Clarke@G.GP.CS.CMU.EDU

15 PegaSys

"PegaSys is a system design tool[1] that uses a simple tableaux-based prover for checking constraints on system specifications (e.g., "type constraints" on the predicates), for checking the correctness of design refinement steps, and for answering queries. PegaSys is currently being used by the client who sponsored its development on several major commercial development efforts; it's out in the world proving thousands of (easy) theorems every day."

Input:	first order predicate logic
Implementation: Availability:	sequential, Lisp
Contact Address:	R. Riemenschneider email: rar@csl.sri.com

16 PI

"The PI system [38, 39, 29] computes all prime implicants and prime implicates of a propositional formula in negation normal form (NNF). We have shown that avoiding CNF and DNF is exponentially advantageous for certain classes of formulas. The system combines a new algorithm, PI, with the path dissolution inference rule. (Dissolution is an efficient generalization of the tableau method.) The core of the system is an implementation of path dissolution which can be run as a theorem prover for propositonal logic. The current system is only a prototype but will be available via ftp in the future.

It is planned to extend the system to output essential versus non-essential implicants/implicates, and to output minimal sets of implicants/implicates that are equivalent to the input formula. Further extensions for handling incremental problems and for multiple-valued logics are planned."

Input:	ground NNF (negation normal form)
Implementation: Availability:	sequential, C and Lisp on SUN 3/60 (SunOS 4.*) source code, binaries, and documentation available
Contact Address:	A.G. Ramesh email: rameshag@cs.albany.edu *or* nvm@cs.albany.edu

17 PROTEIN

"The PROTEIN (PROver with a Theory Extension INterface, [6]) system is a theorem prover for first-order predicate calculus. It follows basically Loveland's Model Elimination procedure with several calculus refinements (e.g. regularity, factorisation). Additionally, PROTEIN provides the calculus variant "Restart Model Elimination" ([7]) which does not need contrapositives. Equality is built-in using Brand's

[1]The author(s) of PegaSys have several papers either submitted for publication or in preparation, but none describes the theorem prover in any detail. They consider it to be an application of proven technology, not part of their research.

"modification method". An important feature is PROTEIN's interface for general theory reasoning, which can e.g. be instantiated with automatically "completed" theories ([5]).
PROTEIN is implemented as a compiler according to Stickel's PTTP-Technique and thus exhibits high inference rates."

Input:	first order predicate logic in clausal form, theories (optionally)
Implementation:	sequential, ECL^{i}PS^e-Prolog
Availability:	source code available
Contact Address:	P. Baumgartner email: peter@informatik.uni-koblenz.de

18 pTAB

"pTAB [49] is a tableaux-based theorem prover implemented in Prolog. It has been developed within a diploma thesis and is integrated into the ILF deduction experimental shell [12]. pTAB can also be used as a standalone system."

Input:	first order predicate logic
Implementation:	sequential, Prolog (sun OS and DOS)
Availability:	source code available
Contact Address:	A. Wolf email: wolf@informatik.hu-berlin.de

19 Prolog Technology Theorem Prover (PTTP)

"The Prolog Technology Theorem Prover (PTTP) [45, 46] is an implementation of the Model Elimination theorem-proving procedure that extends Prolog to the full first-order predicate calculus. PTTP differs from Prolog in its use of unification with the occurs check for soundness, depth-first iterative-deepening search instead of unbounded depth-first search to make the search strategy complete, and the Model Elimination reduction rule that is added to Prolog inferences to make the inference system complete. Two versions are available: one is written in Common Lisp and compiles clauses into Common Lisp; the other is written in Prolog and compiles clauses into Prolog."

Input:	first order predicate logic in clausal form
Implementation:	sequential, Prolog and Lisp (Common Lisp)
Availability:	source code available
Contact Address:	M. Stickel email: stickel@ai.sri.com

20 [C. Schwind]

"This tableaux-based theorem prover [44, 43, 26] contains a kernel for classical logic and extensions to several systems of propositional, modal and temporal logic. One

extension of this prover is for default logic, one for action logic."

Input:	see above
Implementation:	C (for propositional logic), Prolog II
Availability:	binaries and documentation available
Contact Address:	C. B. Schwind email: schwind@gia.univ-mrs.fr

21 SETHEO

The SEquential THEOrem prover SETHEO [27] is based on the Model Elimination Calculus [28]. It is implemented as an extended Warren Abstract Machine; input clauses are preprocessed and compiled into abstract machine code. Completeness is ensured by iterative deepening using several depth measures.

A number of mechanisms for pruning the search space, based upon the efficient handling of syntactic unequality constraints have been developed and allow to successfully tackle difficult problems. Proofs, found with SETHEO, can be displayed in a graphical way (X-Windows). Several parallel provers have been developed on top of SETHEO: dynamic partitioning of the search space, using a dynamic scheme for work distribution (PARTHEO, [42]), static partitioning with slackness (SPTHEO), and random competition (RCTHEO, [15]).

Input:	first order predicate logic in clausal form
Implementation:	sequential and parallel, C
Availability:	binaries and sources available ftp: flop.informatik.tu-muenchen.de:/pub/fki/setheo.tar.Z
Contact Address:	J. Schumann email: setheo@informatik.tu-muenchen.de

22 SHARE

"Shannon graphs are a representation that lies between Binary Decision Diagrams (BDDs) and semantic tableaux: Shannon graphs can be understood as either non-reduced BDDs, or as a linear representation of fully expanded tableau (linear w.r.t. the length of the negation normal form of a formula). The SHARE (SHAnnon graph REfutation system) system [35, 36, 37] is an experimental, compilation-based theorem prover: it translates arbitrary first-order formulae into Shannon graphs and then compiles the graphs into a Prolog program; the proof search is carried out by running the generated program."

Input:	first order predicate logic
Implementation:	sequential, Prolog (Quintus Prolog) on SunOS
Availability:	system available upon request
Contact Address:	J. Posegga email: posegga@ira.uka.de

23 Tableau

"Tableau [11] is an implementation of the propositional case of Smullyan's tableau based inference algorithm. For efficiency, unit-resolution is special cased and several heuristics are used to pick the clauses to branch on. Tableau has been used for a series of large experiments on locating hard areas in satisfiability problems (see [11][2])."

Input:	propositional logic in clausal form
Implementation:	sequential, C, Lisp, shell-scripts for coarse-grain parallelism
Availability:	
Contact Address:	J. Crawford email: jc@research.att.com L.D. Auton email: lda@research.att.com

24 [D. Vallstroem]

"This theorem prover is designed for "Solovay's" modal logics of provability, G and G*. The "system" is just a small theorem prover based on the method of semantic trees, depth first, together with a brief description of the logics and the program, and a couple of examples. There should be informal proofs of the corectness of the important parts of the program ready soon. The program is meant to be used together with an interactive compiler/interpreter."

Input:	"Solovay's" modal logics of provability, G and G*
Implementation:	sequential, SML
Availability:	system available
Contact Address:	D. Vallstroem email: hugo@student.cs.chalmers.se

References

[1] O.L. Astrachan. *Investigations in Model Elimination Based Theorem Proving.* PhD thesis, Duke University, 1992.

[2] O.L. Astrachan and D.W. Loveland. METEORs: High Performance Theorem Provers using Model Elimination. In R.S. Boyer, editor, *Automated Reasoning: Essays in Honor of Woody Bledsoe*. Kluwer Academic Publishers, 1991.

[3] O.L. Astrachan and M.E. Stickel. Caching and Lemmaizing in Model Elimination Theorem Provers. In D. Kapur, editor, *Proc. CADE 11, 11th International Conference on Automated Deduction, Saratoga Springs, NY, USA*, volume 607 of *LNAI*, pages 224 – 238. Springer, June 1992.

[4] W. Atkinson and J. Cunningham. Proving Properties of a Safety-Critical System. *IEEE Software Engineering Journal*, 6(2):41–50, 1991. Charles Babbage Award Paper.

[5] P. Baumgartner. Linear Completion: Combining the Linear and the Unit-Resulting Restrictions. Research Report 9/93, University of Koblenz, 1993.

[6] P. Baumgartner and U. Furbach. Model Elimination without Contrapositives. In *Proc. 12th International Conference on Automated Deduction*. Springer, 1994. (to appear).

[2] A postscript version of the paper can be obtained upon request.

[7] P. Baumgartner and U. Furbach. PROTEIN: A *PRO*ver with a *T*heory *E*xtension *I*nterface. In *12th International Conference on Automated Deduction*. Springer, 1994. (to appear).

[8] B. Beckert and R. Hähnle. An Improved Method for Adding Equality to Free Variable Semantic Tableau. In D. Kapur, editor, *Proc. CADE 11, 11th International Conference on Automated Deduction, Saratoga Springs, NY, USA*, volume 607 of *LNAI*, pages 507 – 521. Springer, June 1992.

[9] S. Bose, E.M. Clarke, D.E. Long, and S. Michaylov. Parthenon: A Parallel Theorem Prover for Non-Horn Clauses. *Journal of Automated Reasoning*, 8:153–181, 1992.

[10] M. Costa and J. Cunningham. Tableaux for an Action Logic. Technical Report PPR1, Esprit 3125 Medlar project, 1993. to appear and accepted subject to revision for the Journal of Logic and Computation.

[11] J.M. Crawford and L.D. Auton. Experimental Results on the Crossover Point in Satisfiability Problems. In *Proc. AAAI-93*, 1993. To appear.

[12] B.I. Dahn. Integration of Logic Functions. In J. Denzinger J. Avenhaus, editor, *Proc. of the Annual Meeting of "GI-Fachgruppe Deduktion"*, SEKI-Report SR-93-11, page 7ff. Universität Kaiserslautern, 1993.

[13] E.A. de Kogel and W.M.J. Ophelders. A Tableaux-based Automated Theorem Prover. In H.C.M. de Swart, editor, *LOGIC: Mathematics, Language, Computer Science and Philosophy*, volume II. Verlag Peter Lang, 1994.

[14] H.C.M. de Swart and W.M.J. Ophelders. Tableaux, Resolution, and Complexity of Formulas. In *Methods of Logic in Computer Science*, 1993. to appear.

[15] W. Ertel. OR-Parallel Theorem Proving with Random Competition. In *Proceedings of LPAR'92*, pages 226–237, St. Petersburg, Russia, 1992. Springer LNAI 624.

[16] S. Gerberding. DT—Ein Tableaubeweiser für dreiwertige Prädikatenlogik erster Stufe. User's Manual and System Description, January 1992.

[17] J. Goubault. *Démonstration automatique en logique classique: complexité et méthodes*. PhD thesis, Laboratoire d' informatique de l' Ecole Polytechnique, Ecole Polytechnique, Palaiseau, France, June 1993.

[18] J. Goubault. Syntax Independent Connections. In *Proc. 2nd Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Marseilles, France*. Max Planck-Institut für Informatik, Saarbrücken, Germany, April 1993.

[19] M. Grundy. Cameo: Refutation Proving by Unordered Model Classifications. Technical Report TR–ARP–6/90, The Australian National University, Automated Reasoning Project, 1990.

[20] Mark Grundy. Tableau Efficiency and Cameo. Technical Report TR–ARP–13/91, The Australian National University, Automated Reasoning Project, 1991.

[21] Mark Grundy. *Theorem Prover Generation using Refutation Procedures*. PhD dissertation, The University of Sydney, Basser Department of Computer Science, 1992.

[22] R. Hähnle. *Automated Theorem Proving in Multiple–Valued Logics*. Oxford University Press, *forthcoming*, 1993.

[23] R. Hähnle, B. Beckert, S. Gerberding, and W. Kernig. The Many–Valued Tableau–Based Theorem Prover $_3P^AP$. Technical report, IBM Germany Scientific Center Institute of Knowledge Based Systems, July 1992.

[24] R. Hähnle and P. H. Schmitt. The Liberalized δ–Rule in Free Variable Semantic Tableaux. Technical report, Universität Karlsruhe, Institut für Logik, Komplexität und Deduktionssysteme, April 1991.

[25] R. Kumar, K. Schneider, and Th. Kropf. Structuring and Automating Hardware Proofs in a Higher-Order Theorem-Proving Environment. *Journal of Formal Methods in System Design*, 2(2):165–230, 1993.

[26] E. Lafon and C. B. Schwind. A Theorem Prover for Action Performance. In E. Kodratoff, editor, *European Conference on Artificial Intelligence, ECAI-88*, pages 541 – 546. Pitman, 1988.

[27] R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performance Theorem Prover. *Journal of Automated Reasoning*, 8(2):183–212, 1992.

[28] D.W. Loveland. *Automated Theorem Proving: a Logical Basis*. North-Holland, 1978.

[29] N. Murray and E. Rosenthal. On the Relative Merits of Path Dissolution and the Method of Analytic Tableaux. (Condensed). In *Proc. 2nd Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Marseilles, France*, pages 183–194. Max Planck-Institut für Informatik, Saarbrücken, Germany, April 1993.

[30] W. Neitz. A Connection Method based Theorem Prover with Selective Backtracking. In *Proc. Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Lautenbach, Germany*, number TR 8/92. Institut für Logik, Komplexität und Deduktionssysteme, Universität Karlsruhe, August 1992.

[31] W. Neitz. Selective Backtracking in the Connection Method. In *Proc. IMYCS'92*. Gorden & Breach, 1992.

[32] W.M.J. Ophelders. *Automated Theorem Proving based upon a Tableau-Method with Unification under Restrictions: Theory, Implementation and Empirical Results*. PhD thesis, Tilburg University, The Netherlands, 1992.

[33] W.M.J. Ophelders and H.C.M. De Swart. Tableaux versus Resolution: a Comparison. *Fundamenta Informaticae*, 18(2,3,4):109–127, April 1993. Special Issue: Algebraic Logic and its Applications.

[34] S. Oppacher and E. Suen. HARP: A Tableau-Based Theorem Prover. *Journal of Automated Reasoning*, (4):69–100, 1988.

[35] J. Posegga. Deduktion mit Shannongraphen für Prädikatenlogik erster Stufe. Dissertation, Universität Karlsruhe, Germany, 1993.

[36] J. Posegga and B. Ludäscher. Towards First-order Deduction based on Shannon Graphs. In *Proc. German Workshop on Artificial Intelligence (GWAI), Bonn, Germany*, LNAI. Springer, 1992.

[37] J. Posegga and K. Schneider. Deduction with First-order BDDs. In *Proc. 2nd Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Marseilles, France*. Max Planck-Institut für Informatik, Saarbrücken, Germany, April 1993.

[38] A. Ramesh and N. Murray. Experiments in Computing Prime Implicants and Prime Implicates Using Techniques not Requiring Clause Form. In *Proc. 2nd Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Marseilles, France*, pages 225 – 228. Max Planck-Institut für Informatik, Saarbrücken, Germany, April 1993.

[39] A. Ramesh and N. Murray. Non-Clausal Deductive Techniques for Computing Prime Implicants and Prime Implicates. In *Fourth International Conference on Logic Programming and Automated Reasoning (LPAR), St. Petersburg*, LNAI. Springer, July 1993.

[40] K. Schneider, R. Kumar, and Th. Kropf. Accelerating Tableaux Proofs using Compact Representations. *Journal of Formal Methods in System Design*, 1993.

[41] K. Schneider, R. Kumar, and Th. Kropf. Hardware Verification with First-Order BDDs. In *Conference on Computer Hardware Description Languages*, 1993.

[42] J. Schumann and R. Letz. PARTHEO: A High-Performance Parallel Theorem Prover. In *Proceedings of the 10th International Conference on Automated Deduction (CADE)*, pages 40–56. Springer LNAI 449, 1990.

[43] C. B. Schwind. Un Dimonstrateur de Thiorhmes pour des Logiques Modales et Temporelles, en Prolog. In *5hme Congrhs AFCET Reconnaissances des formes et Intelligence Artificielle, Grenoble*, pages 897–913, 1985.

[44] C. B. Schwind. A Tableaux-based Theorem Prover for a Decidable Subset of Default Logic. In M.E. Stickel, editor, *Proc. CADE 10, 10th International Conference on Automated Deduction, Kaiserslautern, Germany*, volume 449 of *LNAI*, pages 528–542. Springer Verlag, July 1990.

[45] M. E. Stickel. A Prolog technology theorem prover: Implementation by an extended Prolog compiler. *Journal of Automated Reasoning*, 4(4):353 – 380, December 1988.

[46] M. E. Stickel. A Prolog technology theorem prover: a new exposition and implementation in Prolog. *Theoretical Computer Science*, 104:109 – 128, 1992.

[47] T. Tammet. Proof Search Strategies in Linear Logic. Programming Methodology Group Report 70, Chalmers University of Technology, University of Göteborg, 1993.

[48] K. Wallace. *Tableau-Based Automated Theorem Provers: the Use of Heuristics and Software Engineering Techniques to Increase Deductive Power.* PhD thesis, University of Newcastle, Australia, 1994. to appear.

[49] A. Wolf. Deduktionssysteme und Taktikübersetzung. Diplomarbeit, Humboldt-Universität zu Berlin, Fachbereich Mathematik, June 1992.

[50] G. Wrightson. Semantic tableaux, unification and links. Technical Report CSD-ANZARP-84-001, Victoria University, Wellington, New Zealand, 1984.