# Um Provador de Teoremas Multi-Estratégia
# *A Multi-Strategy Theorem Prover*

Adolfo Gustavo Serra Seca Neto
DAINF - UTFPR
http://www.dainf.ct.utfpr.edu.br/~adolfo

Data desta versão: 14 de novembro de 2008
Date of this version: November 14th, 2008

# Sobre

Este é um capítulo da minha tese de Doutorado intitulada "Um Provador de Teoremas Multi-Estratégia". Esta tese, na área de Ciência da Computação, foi defendida em 30 de janeiro de 2007 no Instituto de Matemática e Estatística (IME) da Universidade de São Paulo (USP). Meu orientador foi o Prof. Dr. Marcelo Finger. O texto completo desta tese está disponível em
`http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04052007-175943/`

# About

*This is a chapter of my Ph.D. thesis entitled "A Multi-Strategy Theorem Prover". This Computer Science thesis was defended on January 30th, 2007 at the Institute of Mathematics and Statistics (IME) of the University of São Paulo (USP). My advisor was Prof. Dr. Marcelo Finger. Thesis full text is available at* `http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04052007-175943/`. *Only the first chapter was written in Portuguese. All the following appendices were written in English.*

# Apêndice F

# Brief User Manual

Here we present a simple user manual. Firstly we show its installation procedure. After that we describe some scenarios for using **KEMS**, where we will see its functionalities.

## F.1 Installation

As **KEMS** was implemented in Java, it can be run on any platform for which there is a Java Runtime Environment (JRE), version 5.0. Therefore the only requirement is to have a JRE installed on your computer. The JRE is available for several operating systems (see [80] for more details on how to install a JRE for a specific system).

The following instructions are specific for the Linux platform, but it is easy to adapt them for Windows and other operating systems. First we assume you have placed the `kems.zip` file (available at [92]) in the directory `/home`. Unzip the file with the

```
unzip kems.zip
```

command. The `kems.export` directory should be created. Henceforth we refer to

```
/home/kems.export
```

as `KEMS.HOME`.

If the JRE is in the `PATH`[1] and you are in `KEMS.HOME`, then you can issue the

[1]The JRE installation should put the JRE directory (`$JAVA_HOME/bin`) in the `PATH`, but if that is not the case you can see how to set the `JAVA_HOME` and PATH environment variables in [80].

```
java -jar kems.jar
```

command. This is going to start **KEMS** graphical interface. If you want to determine the amount of heap memory to be used by **KEMS**[2], you can issue the

```
java -Xms size -Xmx size -jar kems.jar
```

command. '`-Xmx`' and '`-Xms`' are `java` command options. The `-Xmx size` option sets the maximum heap size to size. Kilobytes are indicated by the letters k or K and megabytes by m or M. Similarly, `-Xms size` determines the initial heap size. For instance,

```
java -Xms200m -Xmx800m -jar kems.jar
```

sets the minimum heap size to 200 megabytes and the maximum heap size to 800 megabytes.

## F.2 Scenarios

Here we will present some scenarios for using **KEMS**. Let us enumerate the main ones:

1. configure the prover;
2. run a problem (that is in a file) with one prover configuration;
3. edit a problem and run it with one prover configuration;
4. run a sequence of problems with a list of prover configurations.

### F.2.1 Configuring the Prover

Several scenarios require the prover to be previously configured. To configure the prover we need to:

1. open the Prover Configurator (see Figure F.2) by choosing on main window (see Figure F.1) menu bar the `Configure` option and then clicking on the `Prover` option;

[2]This can be necessary to run some difficult problems.

2. choose a logical system;

3. choose a strategy;

4. set the number of times to run each problem;

5. set the maximum number of minutes to run each problem (with a prover configuration);

6. choose the analyzer name;

7. choose a sorter;

8. mark/unmark the 'save formula origin', 'discard closed branches', and 'save discarded branches' options.


Figure F.1: **KEMS** main window.

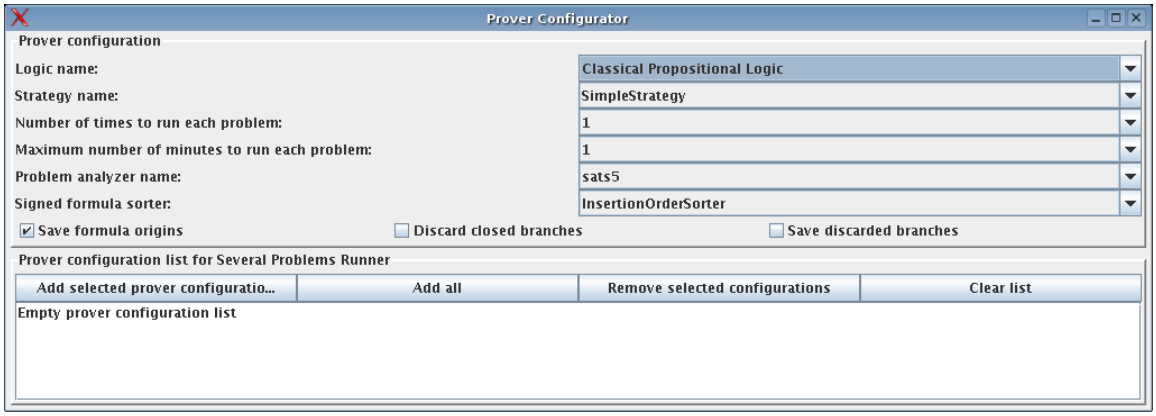


Figure F.2: Prover Configurator window.

### F.2.2 Choosing and Running a Problem

To run the first scenario (run a problem that is in a file with one prover configuration) you must first configure the prover. After that you must choose on main window menu bar the `Problem` option and then click on the `Instance - Choose, Run and View Proof of a Problem Instance` option. Next you perform the following actions:

1. choose a problem file (you can browse directories on the `Open` window until you find the desired file);
2. run the problem (by clicking on the `Run` button);
3. browse the proof.

Later we will describe how the user can browse the proof.

### F.2.3 Editing and Running a Problem

The second scenario, edit a problem and run it with one prover configuration, comprises the following:

1. open the Problem Editor (see Figure F.3) by choosing main window menu bar `Problem` option and after that clicking on `Editor`;
2. type the problem on (or load it from a file to) the Problem Editor window;
3. configure the prover;
4. run the problem (choose Problem Editor menu bar `Run` option and after that click on `Run this problem`);
5. browse the proof.

The user can enter a problem either using SATLIB CNF format [101] or in an extension of the format used in [38]. To enter a problem (either in the Problem Editor window or when editing a problem file in any text editor) using the extension of the format of [38], one must provide a list of signed formulas. Each line can contain at most one s-formula.

Comment lines start with a `#`. A signed formula is a sign followed by a formula. The allowed signs are `T` (true) and `F` (false).

Formulas can be atomic or composite. An atomic formula is a string of letters and numbers initiated by a letter. Composite formulas have a connective and zero, one or two subformulas (which are themselves formulas). The notation for composite formulas is the common infix notation. For zeroary connectives, a formula is its `Connective`. For unary connectives, `Connective Formula`. And for binary connectives, `Formula Connective Formula`. Parentheses are used when necessary to establish precedence.

The allowed connectives are:

**zeroary -** `TOP` and `BOTTOM`;

**unary -** `!` (not), `@` (consistency) and `*` (inconsistency);

**binary -** `&` (and), `|` (or), `->` (implication), `<=>` (bi-implication) and `+` (exclusive or).

The following is an example of a **CPL** problem:

```
T  A <=> (B&!C)
T  TOP-> !(A|D)
F  BOTTOM | (!D)
```

And here we can see an example of a **LFI** problem:

```
T  A <=> @(B&!*C)
T  TOP-> !(A|@D)
F  BOTTOM | @(!D)
```

In **mbC** and **mCi**, '`*A`' is translated into '`!@A`'. The previous problem can also be submitted to a **CPL** strategy: '`@A`' formulas will be translated into '`TOP`' and '`*A`' into '`BOTTOM`'.

### F.2.4 Running a Problem Sequence

For the third scenario, running a sequence of problems with a list of prover configurations, we have the following steps:

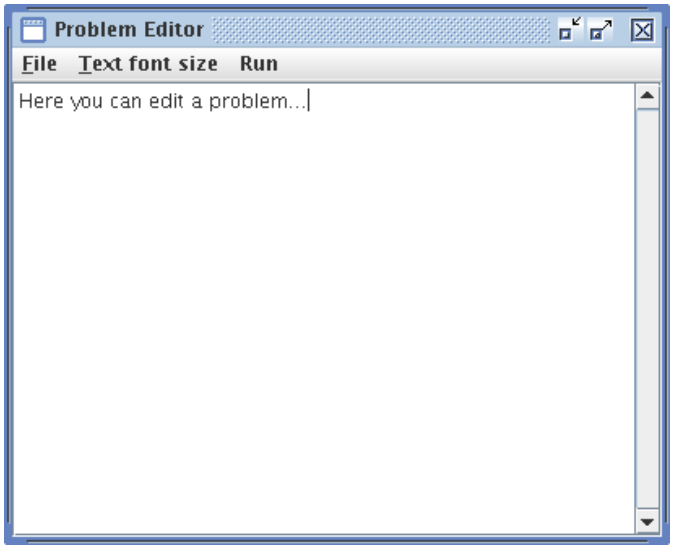


Figure F.3: Problem Editor.

1. configure the prover (here it is sometimes useful to create a list of prover configurations in the bottom part of the Prover Configurator window[3]);

2. open the Several Problems Runner window (see Figure F.4) by choosing main window menu bar `Problem` option and after that clicking on `Several - Choose and Run Several Problems`;

3. choose problem files (see below) one or more times;

4. run the problem sequence (choose Several Problems Runner menu bar `Run` option and after that click on `Run problems`);

5. wait until **KEMS** finishes all problems with all selected prover configurations[4].

The results window show some extra information about the proof such as proof size, time spent, proof tree height and problem size.

To choose one or more problem files you must:

1. click on the `Choose one or more problem instances` button;

[3]For other scenarios only the currently selected prover configuration (PC) is used — the list of PCs is not taken into consideration.

[4]You can also choose Several Problems Runner menu bar `Results window` option and after that click on `Show current results` option to show a partial results window (more recent results appear on the top) that will be updated as new results are obtained.

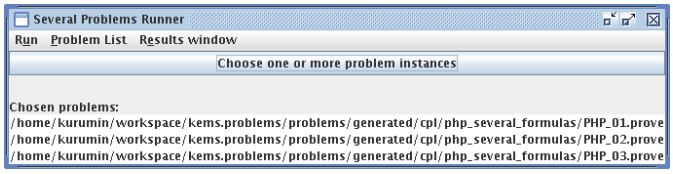


Figure F.4: Several Problems Runner window.

2. browse directories on the `Open` window until you find the desired files. If you hold the `ctrl` key you can choose more than one file;

3. click on the `Choose` button to add the selected files to the problem list.

### F.2.5 Browsing a Proof

After a proof is obtained, a window such as the one in Figure F.5 is opened. On the left side there is an interactive proof viewer. On the right side we have a graphical proof viewer.

#### Interactive Proof Viewer

The interactive proof viewer (IPV) allows the user to see one branch at a time. The IPV shows a list of signed formulas. The currently selected branch is highlighted in the graphical proof viewer and identified in a button on the top of this list. By clicking on this button it one can see more information about the branch.

The window that is opened to show more information about the currently selected branch displays the prover configuration used to run the problem, a valuation (when the proof tree is open), and statistics about the proof.

If the user clicks with the left mouse button on one signed formula button, that s-formula is highlighted as well as all other s-formulas that gave origin to that s-formula. If the user clicks with the right mouse button on one signed formula button that s-formula's immediate origin (rule, major premise and minor premise) is shown.

Whenever there is a (PB) rule application, instead of one s-formula button, two s-formula buttons appear on one line: one for each (PB) s-formula. One of the two s-formula buttons is highlighted – the one which is in current branch – and the other is not

highlighted. If the user clicks on the unselected s-formula button, the IPV then shows the leftmost branch that includes that s-formula.

Whenever IPV shows a closed branch, a button that shows an '×' symbol is included as the last button of the list of s-formula buttons of that branch.

### Graphical Proof Viewer

The graphical proof viewer (GPV) starts by showing a graphical representation of the proof. Initially, circles are presented in the place of s-formulas. This graphic gives us an idea of the proof form, how many applications of (PB) were necessary, how many s-formulas were included in the proof, and so on. The '×', '■' and '□' symbols that appear in the end of every branch denotate, respectively, that a branch is either 'closed', 'open and completed', or 'open and not completed'.

If the user clicks on GPV's area with the right mouse button a menu will appear allowing the user to set or unset some options and to perform one action:

- if the 'show circles' option is set, GPV shows circles as elements of tableau tree nodes. If it is unset, GPV shows s-formulas;
- if the 'show circles' option is unset and the 'show numbers' option is set, a unique sequential number is assigned to each s-formula (and show at the right of the s-formula);
- if the 'show circles' option is unset, the 'show sign marking used formulas' option is set, and the tableau is closed, a '*' sign is assigned to each s-formula effectively used to close the tableau (and displayed at the right of the s-formula);
- if the 'change parameters' action is selected, a window with some GPV and IPV parameters is exhibited. If the user changes some parameter(s) and clicks the 'Update' button, both viewers are refreshed with the new values for the changed parameters.

After setting parameters the user can scroll the proof using the horizontal and vertical scroll bars (that appear in GPV depending on GPV's width and height parameter values). It would be interesting if these parameters could automatically adjust themselves

according to the proof object and other GPV parameters, but that does not seem to be a trivial task and was not implemented in **KEMS** current version.

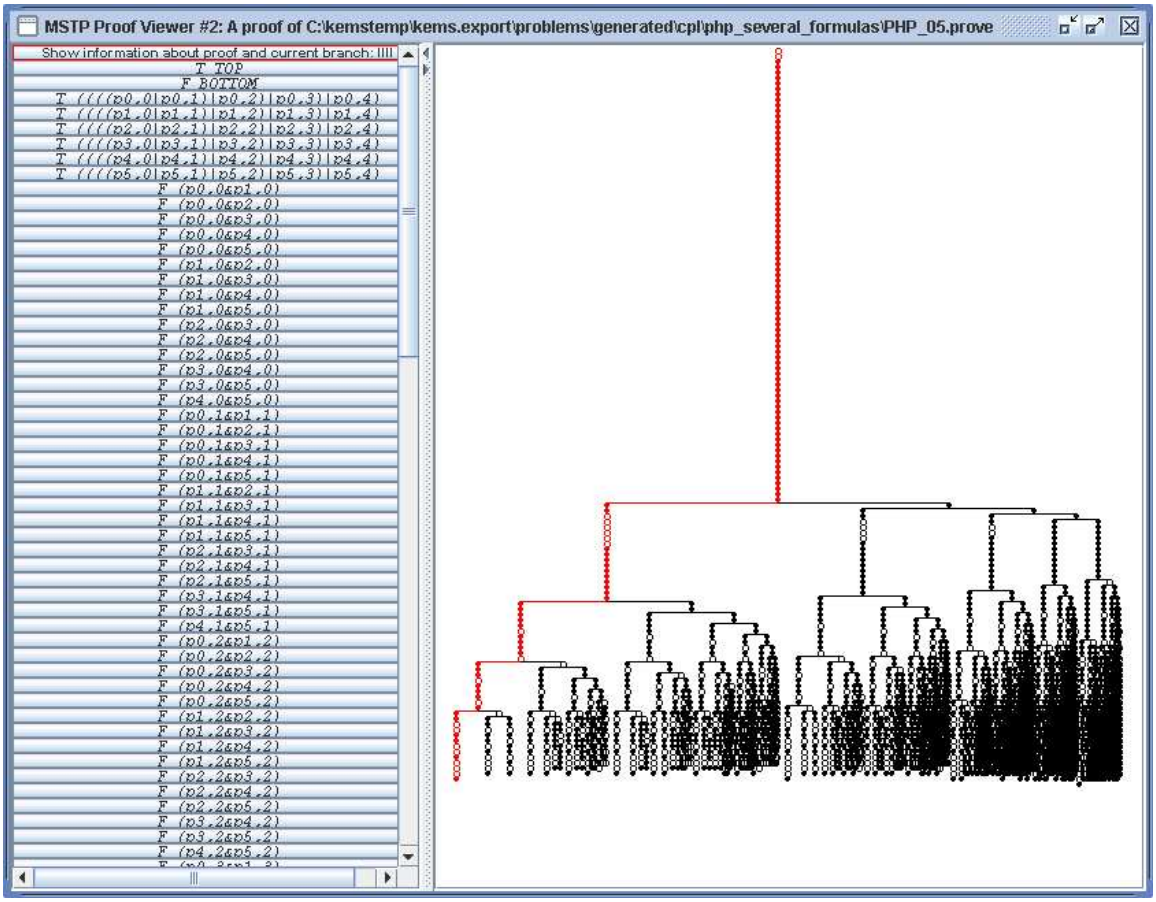


Figure F.5: A Proof Viewer window.

### F.2.6 Command-line Sequence Runner

We will not describe in detail here but it is possible to run a sequence of problems from command line without opening **KEMS** graphical interface. A sequence file such as the following must be given as input:

```
parser=sats5

saveOrigin=false

discardClosedBranches=true

saveDiscardedBranches=false

times=1
```

```
timeLimit=3


problems=
problems/generated/cpl/php_several_formulas/PHP_08.prove
problems/generated/cpl/php_several_formulas/PHP_09.prove
problems/generated/cpl/php_several_formulas/PHP_10.prove



strategies=
MemorySaverStrategy
SimpleStrategy
#BackjumpingSimpleStrategy

comparators=
ReverseInsertionOrderComparator
OrComparator
TrueComparator

run
```

The sequence file above tells **KEMS** to run 3 PHP instances with 6 strategy-sorter pairs (all possible combinations of the 2 strategies and 3 comparators listed). One strategy name has a '#' character before it. This character is used to mark a comment so this strategy name will not be used by **KEMS**.

# Referências Bibliográficas


[1] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. In *STOC '02: Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 448–456, New York, NY, USA, 2002. ACM Press.

[2] Noriko Arai, Toniann Pitassi, and Alasdair Urquhart. The complexity of analytic tableaux. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 356–363. ACM Press, 2001.

[3] Bernhard Beckert, Richard Bubel, Elmar Habermalz, and Andreas Roth. jTAP - a Tableau Prover in Java, February 1999. Universitat Karlsruhe. Available at `http://i12www.ira.uka.de/~aroth/jTAP/`. Last accessed, November 2006.

[4] Bernhard Beckert and Joachim Posegga. leanTAP: Lean tableau-based deduction. *Journal of Automated Reasoning*, 15(3):339–358, 1995.

[5] Evert W. Beth. *The Foundations of Mathematics*. North-Holland Publishing Company, Amsterdam, 1959.

[6] Maria Paola Bonacina and Thierry Boy de la Tour. Fifth Workshop on Strategies in Automated Deduction - Workshop Programme, 2004. `http://tinyurl.com/y8dkbj`. Last accessed, November 2006.

[7] Maria Luisa Bonet and Nicola Galesi. A study of proof search algorithms for resolution and polynomial calculus. In *FOCS '99: Proceedings of the 40th Annual*

*Symposium on Foundations of Computer Science*, page 422, Washington, DC, USA, 1999. IEEE Computer Society.

[8] Krysia Broda, Marcello D'Agostino, and Marco Mondadori. A Solution to a Problem of Popper. In *The Epistemology of Karl Popper*. Kluwer, 1995. `http://citeseer.nj.nec.com/broda95solution.html`. Last accessed, November 2006.

[9] S. R. Buss. Polynomial size proofs of the propositional pigeonhole principle. *Journal of Symbolic Logic*, 52:916–927, 1987.

[10] Liming Cai. *Nondeterminism and optimization*. PhD thesis, Texas A&M University, USA, 1994.

[11] Carlos Caleiro, Walter Carnielli, Marcelo E. Coniglio, and Joao Marcos. Two's company: "The humbug of many logical values". In *Logica Universalis*, pages 169–189. Birkhäuser Verlag, Basel, Switzerland, 2005. Pre-print available at `http://tinyurl.com/yb5qbz`. Last accessed, November 2006.

[12] Alessandra Carbone and Stephen Semmes. *Graphic Apology for Symmetry and Implicitness*. Oxford University Press, 2000.

[13] W. A. Carnielli. Systematization of the finite many-valued logics through the method of tableaux. *The Journal of Symbolic Logic*, 52:473–493, 1987.

[14] W.A. Carnielli and J. Marcos. Ex Contradictione Non Sequitur Quodlibet. *Bulletin of Advanced Reasoning and Knowledge*, 1:89–109, 2001.

[15] W.A. Carnielli and J. Marcos. Tableau systems for logics of formal inconsistency. *Proceedings of the International Conference on Artificial Intelligence (IC-AI'2001)*, pages 848–852, 2001.

[16] Walter Carnielli. How to build your own paraconsistent logic: an introduction to the Logics of Formal (In)Consistency. *Proceedings of the Workshop on Paraconsistent Logic (WoPaLo)*, 2002.

[17] Walter Carnielli, Marcelo Coniglio, and Ricardo Bianconi. *Logic and Applications: Mathematics, Computer Science and Philosophy (in Portuguese).* Unpublished, 2005. Preliminary version. Chapters 1 to 5.

[18] Walter Carnielli, Marcelo E. Coniglio, and Joao Marcos. Logics of Formal Inconsistency. In *Handbook of Philosophical Logic*, volume 12. Kluwer Academic Publishers, 2005. To appear. Pre-print available at `http://tinyurl.com/ybn4yw`. Last accessed, November 2006.

[19] Walter A. Carnielli and Richard L. Epstein. *Computabilidade – Funções Computáveis, Lógica e os Fundamentos da Matemática*. Editora da Unesp, 2006.

[20] Stephen Cook. The P versus NP problem, 2000. `http://tinyurl.com/n5thm`. Last accessed, May 2005.

[21] Stephen Cook. The importance of the P versus NP question. *J. ACM*, 50(1):27–29, 2003.

[22] Stephen Cook and Robert Reckhow. On the lengths of proofs in the propositional calculus (preliminary version). In *STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 135–148, New York, NY, USA, 1974. ACM Press.

[23] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM Press.

[24] Stephen A. Cook. A short proof of the pigeon hole principle using extended resolution. *SIGACT News*, 8(4):28–32, 1976.

[25] W. Cook, C. R. Coullard, and G. Turán. On the complexity of cutting-plane proofs. *Discrete Appl. Math.*, 18(1):25–38, 1987.

[26] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms - Second Edition*. MIT Press, 2001.

[27] Newton C. A. da Costa, Décio Krause, and Otávio Bueno. Paraconsistent logics and paraconsistency: Technical and philosophical developments. *CLE e-prints (Section Logic)*, 4(3), 2004. Pre-print available at `http://tinyurl.com/yxhon7`. Last accessed, November 2006.

[28] Marcello D'Agostino. Are Tableaux an Improvement on Truth-Tables? Cut-Free proofs and Bivalence, 1992. Available at `http://citeseer.nj.nec.com/140346.html`. Last accessed, May 2005.

[29] Marcello D'Agostino. Tableau methods for classical propositional logic. In Marcello D'Agostino et al., editor, *Handbook of Tableau Methods*, chapter 1, pages 45–123. Kluwer Academic Press, 1999.

[30] Marcello D'Agostino, Dov Gabbay, and Krysia Broda. Tableau methods for substructural logics. In Marcello D'Agostino et al., editor, *Handbook of Tableau Methods*, chapter 6, pages 397–467. Kluwer Academic Press, 1999.

[31] Marcello D'Agostino and Marco Mondadori. The taming of the cut: Classical refutations with analytic cut. *Journal of Logic and Computation*, pages 285–319, 1994.

[32] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[33] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

[34] Sandra de Amo, Walter Carnielli, and João Marcos. A Logical Framework for Integrating Inconsistent Information in Multiple Databases. In Thomas Eiter and Klaus-Dieter Schewe, editors, *Lecture Notes in Computer Science*, volume 2284, pages 67–84. Springer-Verlag, Berlim., 2002.

[35] Eleonora de Moraes. *Lista de discussão gestar bem interior-sp*, 2004. `http://br.groups.yahoo.com/group/gestarbeminterior-sp`. Last accessed, December 2006.

[36] Rina Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[37] Luis Fariñas del Cerro, David Fauthoux, Olivier Gasquet, Andreas Herzig, Dominique Longin, and Fabio Massacci. Lotrec: The generic tableau prover for modal and description logics. In *IJCAR '01: Proceedings of the First International Joint Conference on Automated Reasoning*, pages 453–458. Springer-Verlag, 2001.

[38] Wagner Dias. Tableaux implementation for approximate reasoning (in portuguese). Master's thesis, Computer Science Department, Institute of Mathematics and Statistics, University of São Paulo, 2002.

[39] Simone Grilo Diniz and Ana Cristina Duarte. *Parto normal ou cesárea? O que toda mulher deve saber (e todo homem também)*. Editora da Unesp, São Paulo, 2004.

[40] Heidi Dixon. *Automating Pseudo-Boolean Inference Within a DPLL Framework*. PhD thesis, University of Oregon, USA, Dec 2004. Available at `http://www.cirl.uoregon.edu/dixon/dixonDissertation.pdf`. Last accessed, August 2005.

[41] Amigas do Parto. *Lista de discussão Parto Nosso*, 2003. `br.groups.yahoo.com/group/partonosso`. Last accessed, December 2006.

[42] Itala M. Loffredo D'Ottaviano and Milton Augustinis de Castro. Analytical Tableaux for da Costa's Hierarchy of Paraconsistent Logics $C_n, 1 \leq n \leq \omega$. *Journal of Applied Non-Classical Logics*, 15(1):69–103, 2005.

[43] Tzilla Elrad, Robert E. Filman, and Atef Bader. Aspect-Oriented Programming. *Communications of the ACM*, 44, 2001.

[44] M. Enkin, M. Skiers, J. Nelson, C. Crowder, L. Duly, and E. Hodnett. *A guide to effective care during pregnancy and childbirth*. Oxford, UK: Oxford University Press, 2000.

[45] Fadynha. *Lista de discussão partonatural*, 1999. `br.groups.yahoo.com/group/partonatural`. Last accessed, December 2006.

[46] Luis Fariñas del Cerro, David Fauthoux, Olivier Gasquet, Andreas Herzig, Dominique Longin, and Fabio Massacci. Lotrec: the generic tableau prover for modal and description logics. In *International Joint Conference on Automated Reasoning*, LNCS, page 6. Springer Verlag, 18-23 juin 2001.

[47] M. Finger and R. Wassermann. Approximate Reasoning and Paraconsistency-Preliminary Report. *Proceedings of the Eigth Workshop on Logic, Language, Information and Comunication (WoLLIC), Braslia, Brazil*, 2001.

[48] M. Finger and R. Wassermann. The universe of approximations. *Electronic Notes in Theoretical Computer Science*, 84:1–14, 2003.

[49] M. Finger and R. Wassermann. Anytime Approximations of Classical Logic from Above. *Journal of Logic and Computation*, 2006.

[50] M. Finger and R. Wassermann. The universe of propositional approximations. *Theoretical Computer Science*, 355(2):153–166, 2006.

[51] Melvin Fitting. *First-order logic and automated theorem proving (2nd ed.)*. Springer-Verlag New York, Inc., 1996.

[52] Melvin Fitting. Introduction. In Marcello D'Agostino et al., editor, *Handbook of Tableau Methods*, chapter 1, pages 1–43. Kluwer Academic Press, 1999.

[53] Zhaohui Fu, Yogesh Mahajan, and Sharad Malik. New Features of the SAT'04 versions of zChaff, 2004. `http://www.princeton.edu/~chaff/zchaff/sat04.pdf`. Last accessed, September 2005.

[54] Dov M. Gabbay. *Labelled Deductive Systems, Volume 1.* Oxford University Press, Oxford, 1996.

[55] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Adisson-Wesley, 1994.

[56] Gerhard Gentzen. Investigations into logical deductions, 1935. In M. E. Szabo, editor, *The Collected Works of Gerhard Gentzen*, pages 68–131. North-Holland, Amsterdam, 1969.

[57] J. Gosling, B. Joy, and G. Steele. *The Java Programming Language*. Addison-Wesley, Reading, MA, 1996.

[58] Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985.

[59] Klaus Havelund and Natarajan Shankar. Experiments in theorem proving and model checking for protocol verification. In *FME '96: Proceedings of the Third International Symposium of Formal Methods Europe on Industrial Benefit and Advances in Formal Methods*, pages 662–681. Springer-Verlag, 1996.

[60] J. Hintikka. Form and content in quantification theory. *Acta Philosophica Fennica*, 8:7–55, 1955.

[61] Jan Holub and Borivoj Melichar. Implementation of nondeterministic finite automata for approximate pattern matching. In *WIA '98: Revised Papers from the Third International Workshop on Automata Implementation*, pages 92–99. Springer-Verlag, 1999.

[62] Scott E. Hudson, Frank Flannery, C. Scott Anaian, Dan Wang, and Andrew Appel. *CUP Parser Generator for Java*, 1999. `http://www2.cs.tum.edu/projects/cup`. Last accessed, November 2006.

[63] Ullrich Hustadt and Renate A. Schmidt. Simplification and backjumping in modal tableau. In *Lecture Notes in Computer Science*, volume 1397 of *Lecture Notes in Computer Science*, pages 187–201, 1998.

[64] Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William G. Griswold. Getting Started with AspectJ. *Communications of the ACM*, 44:59–65, 2001.

[65] Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William G. Griswold. An overview of AspectJ. *Lecture Notes in Computer Science*, 2072:327–355, 2001.

[66] Gerwin Klein. *JFlex: the fast lexical analyzer generator for Java*, 1998. `http://jflex.de`. Last accessed, November 2006.

[67] S. Kundu and J. Chen. Fuzzy logic or Lukasiewicz logic: A clarification. *Fuzzy Sets and Systems*, 95(3):369–379, 1998.

[68] L. Di Lascio. Analytic fuzzy tableaux. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 5(6):434–439, Dec 2001.

[69] D. W. Loveland. Automated deduction: Some achievements and future directions. Technical report, National Science Foundation, 1997. Available at `http://tinyurl.com/y7mb6p`. Last accessed, May 2005.

[70] D. W. Loveland. Automated deduction: achievements and future directions. *Commun. ACM*, 43(11es):10, 2000.

[71] Wendy MacCaull. Tableau method for residuated logic. *Fuzzy Sets Syst.*, 80(3):327–337, 1996.

[72] Alistair Manning, Andrew Ireland, and Alan Bundy. Increasing the Versatility of Heuristic Based Theorem Provers. In *LPAR'93*, 1993.

[73] Heiko Mantel and Jens Otten. lintap: A tableau prover for linear logic. In *TABLEAUX '99: Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 217–231, London, UK, 1999. Springer-Verlag.

[74] João Marcos. Personal communication by email, October 2006.

[75] Fabio Massacci. Simplification: A general constraint propagation technique for propositional and modal tableaux. In *TABLEAUX '98: Proceedings of the Inter-*

*national Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 217–231. Springer-Verlag, 1998.

[76] Fabio Massacci. The proof complexity of analytic and clausal tableaux. *Theor. Comput. Sci.*, 243(1-2):477–487, 2000.

[77] William McCune and Larry Wos. Otter - The CADE-13 Competition Incarnations. *J. Autom. Reason.*, 18(2):211–220, 1997.

[78] Elliott Mendelson. *Introduction to Mathematical Logic*. Chapman & Hall, London, UK, fourth edition, 1997.

[79] Paulo Blauth Menezes. *Linguagens Formais e Autômatos*. Instituto de Informática da UFRGS : Editora Sagra Luzzatto, Porto Alegre, 232p., 2005.

[80] Sun Microsystems. Java Runtime Environment (JRE) 5.0 Installation Notes, 2006. `http://java.sun.com/j2se/1.5.0/jre/install.html`. Last accessed, November 2006.

[81] S. H. Mirian and M. Mousavi. Nondeterminism in set-theoretic specifications (in persian). In *Proceedings of Iranian Computer Society Annual Conference (CSICC'02)*, feb 2002.

[82] David G. Mitchell, Bart Selman, and Hector J. Levesque. Hard and easy distributions for SAT problems. In Paul Rosenbloom and Peter Szolovits, editors, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 459–465, Menlo Park, California, 1992. AAAI Press.

[83] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)*, June 2001.

[84] John K. Myers. An introduction to planning and meta-decision-making with uncertain nondeterministic action using 2nd-order probabilities. In *Proceedings of the first*

*international conference on Artificial intelligence planning systems*, pages 297–298. Morgan Kaufmann Publishers Inc., 1992.

[85] Adolfo Neto. An Object-Oriented Implementation of a KE Tableau Prover, nov 2003. Avaliable at `http://tinyurl.com/y3qmkx`. Last accessed, November 2006.

[86] Adolfo Neto. Modifications on the implementation of a framework for tableau methods, jul 2003. Avaliable at `http://tinyurl.com/yjgjnq`. Last accessed, November 2006.

[87] Adolfo Neto and Marcelo Finger. A Multi-Strategy Tableau Prover. In *I Simpósio de Iniciação Científica e Pós-Graduação do IME-USP*. University of São Paulo, 2005. Available at `http://tinyurl.com/tbdd6`. Last accessed, November 2006.

[88] Adolfo Neto and Marcelo Finger. A Multi-Strategy Tableau Prover. In *SeMe-2005. Workshop "Semantics and Meaning"*, IFIP International Federation for Information Processing. Unicamp. Campinas-SP., 2005. Available at `http://tinyurl.com/yzx8ve`. Last accessed, November 2006.

[89] Adolfo Neto and Marcelo Finger. Implementing a multi-strategy theorem prover. In Ana Cristina Bicharra Garcia and Fernando Santos Osório, editors, *Proceedings of the V ENIA (Encontro Nacional de Inteligência Artificial), held in São Leopoldo-RS, Brazil, July 22-29 2005*, 2005. Available at `http://tinyurl.com/yd6n6n`. Last accessed, November 2006.

[90] Adolfo Neto and Marcelo Finger. Using Aspect-Oriented Programming in the Development of a Multi-Strategy Theorem Prover. In *Anais da II Jornada do Conhecimento e da Tecnologia do Univem*, Marília-SP, 2005. Available at `http://www.ime.usp.br/~adolfo/trabalhos/jornada2005.pdf`. Last accessed, November 2006.

[91] Adolfo Neto and Marcelo Finger. Effective Prover for Minimal Inconsistency Logic. In *Artificial Intelligence in Theory and Practice*, IFIP International Federation for Information Processing, pages 465–474. Springer Verlag, 2006. Available at

`http://www.springerlink.com/content/b80728w7m6885765`. Last accessed, November 2006.

[92] Adolfo Neto and Marcelo Finger. *KEMS - A* **KE** *Multi-Strategy Tableau Prover*, 2006. `http://kems.iv.fapesp.br`. Last accessed, November 2006.

[93] Michel Odent. *The Caesarean*. Free Association Books, 2004.

[94] Lawrence C. Paulson. *Handbook of logic in computer science (vol. 2): background: computational structures*, chapter Designing a theorem prover, pages 415–475. Oxford University Press, Inc., 1992.

[95] Francis Jeffry Pelletier. Seventy-five problems for testing automatic theorem provers. *J. Autom. Reason.*, 2(2):191–216, 1986.

[96] J. V. Pitt and R. J. Cunningham. Theorem proving and model building with the calculus ke. *Journal of the IGPL*, 4(1):129–150, 1996.

[97] Awais Rashid and Lynne Blair. Editorial: Aspect-oriented Programming and Separation of Crosscutting Concerns. *The Computer Journal*, 46(5):527–528, 2003.

[98] Alexandre Riazanov and Andrei Voronkov. Vampire 1.1 (system description). In *IJCAR '01: Proceedings of the First International Joint Conference on Automated Reasoning*, pages 376–380. Springer-Verlag, 2001.

[99] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1):107–136, 2006.

[100] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.

[101] Satisfiability suggested format, 1993. `http://www.satlib.org`. Last accessed, March 22, 2005.

[102] Satisfiability library, 2003. `http://www.satlib.org`. Last accessed, March 22, 2005.

[103] N. Scharli, S. Ducasse, O. Nierstrasz, and A.P. Black. Traits: Composable units of behaviour. *Proc. of ECOOP*, 2743:248–274, 2003.

[104] J. Schumann. Tableau-based theorem provers: Systems and implementations. *Journal of Automated Reasoning*, 13(3):409–421, 1994. `http://www.springerlink.com/content/k182u80451306371`. Last accessed, November 4th, 2006.

[105] Bart Selman, Hector J. Levesque, and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In Paul Rosenbloom and Peter Szolovits, editors, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446, Menlo Park, California, 1992. AAAI Press.

[106] Raymond M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.

[107] Sérgio Soares and Paulo Borba. AspectJ - Programação orientada a aspectos em Java. *Tutorial no SBLP 2002, 6o. Simpósio Brasileiro de Linguagens de Programação. 5 a 7 de Junho, PUC-Rio, Rio de Janeiro, Brasil*, pages 39–55, 2002.

[108] R. Statman. Bounds for proof-search and speed-up in the predicate calculus. *Annals of Mathematical Logic*, pages 225–287, 1978.

[109] Friedrich Steimann. The paradoxical success of aspect-oriented programming. *SIGPLAN Not.*, 41(10):481–497, 2006.

[110] Geoff Sutcliffe. An overview of automated theorem proving, 2001. `http://www.cs.miami.edu/~tptp/OverviewOfATP.html`. Last accessed, March 2005.

[111] Geoff Sutcliffe. Thousands of problems for theorem provers, 2001. `http://www.cs.miami.edu/~tptp`. Last accessed, March 2005.

[112] Geoff Sutcliffe and Christian Suttner. The CADE ATP System Competition, 2003. `http://www.cs.miami.edu/~tptp/CASC`. Last accessed, March 2005.

[113] Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987.

[114] Marian Vittek. A compiler for nondeterministic term rewriting systems. In *RTA '96: Proceedings of the 7th International Conference on Rewriting Techniques and Applications*, pages 154–167. Springer-Verlag, 1996.

[115] Wikipedia. *Nondetermnistic algorithm*, 2007. `http://en.wikipedia.org/wiki/Nondeterministic`. Last accessed, February 2007.