

KEMS - A Multi-Strategy Tableau Prover

Adolfo Neto¹ and Marcelo Finger² (Advisor)

¹ State University of Santa Catarina (UDESC), Computer Science Department,
Joinville, Brazil

adolfo.usp@gmail.com

<http://www2.joinville.udesc.br/~dcc2agssn>

² University of São Paulo, Computer Science Department,
São Paulo, Brazil

Abstract. In the literature, there are not many theorem provers that treat strategies as first-class citizens, allowing one to prove the same problem with several different strategies and to compare the proofs obtained. In theorem proving, the use of different proof strategies when solving a problem yields proofs of different sizes. And the size of a proof has an impact on the time spent to obtain a proof. We succeeded in developing a multi-strategy theorem prover where we can vary the strategy without modifying the core of the implementation. **KEMS** allows us to describe several proof strategies for the same logical system, and to implement different logical systems.

PhD. Thesis defended on January 30th, 2007. Full text available at <http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04052007-175943>.

1 Introduction

In this paper we describe KEMS, a multi-strategy tableau prover based on the **KE** tableau method [1]. The **KE** system is a tableau method originally developed for classical propositional logic (**CPL**) by Marco Mondadori and Marcello D'Agostino [1], but that has been extended for other logical systems. The **KE** system was presented as an improvement, in the computational sense, over traditional Analytic Tableaux. It is a refutation system, that though close to the analytic tableau method, is not affected by the anomalies of cut-free systems [1].

The **KE** method is one of many logical methods that can be used to solve the satisfiability problem (SAT) for classical propositional logic (**CPL**). SAT is a well known NP-complete problem [2], and the validity of **CPL** sequents is a co-NP-complete problem.

1.1 A Motivating Example

Let us begin with an example that motivates the use of multiple strategies in theorem provers. In Figures 1 and 2, we present two different **KE** proofs of the same sequent: the third instance of the Γ family [3] of problems. The Γ_3 problem instance is represented by the following valid sequent:

$$(p_1 \vee q_1), (p_1 \rightarrow (p_2 \vee q_2)), (q_1 \rightarrow (p_2 \vee q_2)), (p_2 \rightarrow (p_3 \vee q_3)), (q_2 \rightarrow (p_3 \vee q_3)),$$

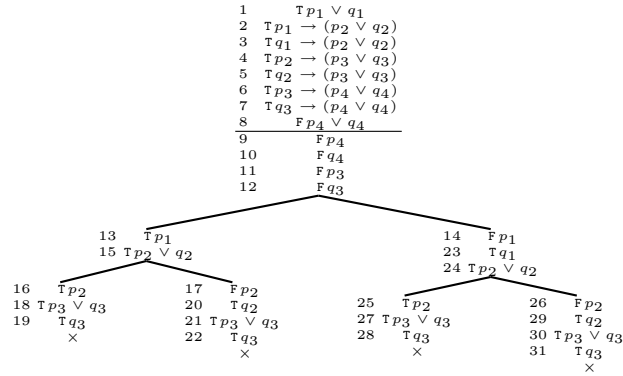


Fig. 1. A CPL KE proof of Γ_3 .

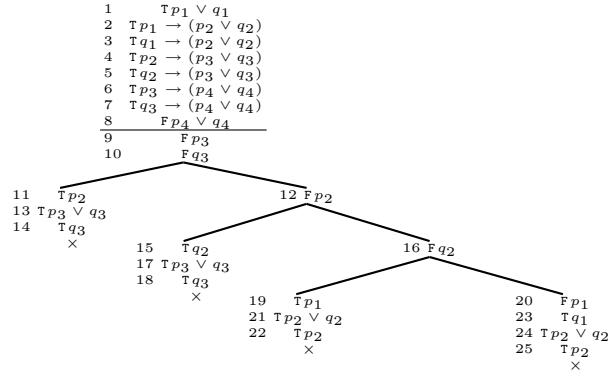


Fig. 2. A smaller CPL KE proof of Γ_3 .

$$(p_3 \rightarrow (p_4 \vee q_4)), (q_3 \rightarrow (p_4 \vee q_4)) \vdash (p_4 \vee q_4)$$

In both proofs, the first step was to include the signed formulas³ numbered 1 to 8 (representing the falsification of the sequent) in the origin. In Figure 1, the next step was to apply all **KE** linear rules that could be applied. This generated formulas 9 to 12. Then, we had to choose the first formula to apply the **KE PB** rule (the only branching rule in **KE**). In this case, we would do better by choosing a formula that could be used as an auxiliary premise with one of the five formulas (1-5) that were not yet used as main premises. By choosing the left subformula of 2, the best result is a proof with size⁴ 71 and 31 nodes.

In Figure 2, we used a different strategy. We did not apply all linear rules that could be applied (formula 8 was not expanded), generating only 9 and 10. After that, we chose the left subformula of 4 to apply the (PB) rule, and the result was a proof

³ From now on we will use the term *s-formula* to refer to signed formulas.

⁴ The size of a proof in the number of logical symbols (formulas and connectives) in that proof.

with size 61 and 25 nodes. The obvious conclusion is that different strategies yield proofs of different sizes. And it is also clear that, for theorem provers, the size of a proof has an impact on the time spent to obtain a proof. As this happened in this very simple problem instance, in bigger instances/more difficult problems the performance gap between strategies is even higher.

1.2 Contributions

In what follows, we present some features of the design and implementation of **KEMS**, a multi-strategy theorem prover based on the **KE** tableau inference system. A multi-strategy theorem prover is a theorem prover where we can vary the strategy without modifying the core of the implementation. Besides being multi-strategy, **KEMS** is capable of proving theorems in three logical systems: classical propositional logic, **mbC** and **mCi**.

We list below some of the contributions of our thesis [4]:

- an analytic, correct and complete **KE** system for **mbC** (see Section 3);
- a correct and complete **KE** system for **mCi** (see Section 3);
- an implementation of a multi-strategy prover which (see Section 4):
 - accepts problems in three logical systems: classical propositional logic, **mbC** and **mCi**;
 - has 6 implemented strategies for classical propositional logic, 2 for **mbC** and 2 for **mCi**;
 - has 13 sorters to be used alongside with the strategies;
 - implements simplification rules of classical propositional logic;
 - provides a proof viewer with a graphical user interface;
 - is open source and is available on the internet at <http://kems.iv.fapesp.br>;
- benchmark results obtained by **KEMS** comparing its classical propositional logic strategies with several problem families (see Section 5);
- seven problem families designed to evaluate provers for logics of formal inconsistency;
- the first benchmark results for the problem families designed to evaluate provers for logics of formal inconsistency.

Some of the results obtained were published in the following papers:

- *A KE Tableau for a Logic of Formal Inconsistency* [5];
- *Effective Prover for Minimal Inconsistency Logic* [6];
- *Implementing a Multi-Strategy Theorem Prover* [7];
- *Using Aspect-Oriented Programming in the Development of a Multi-Strategy Theorem Prover* [8]; and
- *A Multi-Strategy Tableau Prover* [9,10].

And the following papers are under preparation to be submitted to conferences and/or journals:

- *Implementing Backjumping in a Multi-Strategy Tableau Prover*;
- *Implementing Learning in a Multi-Strategy Tableau Prover*; and
- *A KE-based Multi-Strategy Tableau Prover* – a longer paper summarizing the results of our thesis.

2 Logical Systems

We have implemented strategies to obtain proofs for problems in **CPL** and two paraconsistent logics: **mbC** and **mCi**. Paraconsistent logics can be used as the underlying logic to inconsistent but non-trivial theories [11]. Logics of Formal Inconsistency (**LFI**s) [12] are a family of paraconsistent logics that internalize the notions of consistency and inconsistency at the object-language level. This family of logics has some nice proof-theoretic features and have been used in some computer science applications such as the integration of inconsistent information in multiple databases [13]. They can also be used as a tool for knowledge engineering and as the base logic in rule-based decision-support systems.

The logic **mbC** is the weakest **LFI** based on classical logic [12]. One of **mbC** features is the introduction of a connective to represent ‘consistency’ (\circ). That is, the intended reading of $\circ A$ is ‘ A is consistent’.

The **mCi** logic is another **LFI** based on classical logic presented in [12]. It is an extension to **mbC**. The motivation for its development was to enrich **mbC** so as to be able to define an inconsistency connective by the direct use of the paraconsistent negation. The inconsistency connective ‘ \bullet ’ can be defined in **mCi** as $\bullet A \stackrel{\text{def}}{=} \neg \circ A$.

A full account of the syntax and semantics of **mbC** and **mCi** can be found in [12].

3 The KE System

The **KE** System, a tableau method developed by Marco Mondadori and Marcello D’Agostino [14], was presented as an improvement, in the computational efficiency sense, over the Analytic Tableau method. It is a refutation system that, though close to the Analytic Tableau method, is not affected by the anomalies of cut-free systems [1].

In **KEMS** we have implemented strategies for the **KE** systems for **CPL**, **mbC** and **mCi**. The **KE** system for **CPL** we have implemented is an extension of the one described in [14]. To obtain a more efficient system, we have added a set of simplification rules as described in [15]. These simplification inference rules do not cause branching and in some cases may even prevent it. They play for tableau methods the same role of unit propagation for DLL and subsumption for resolution. More details of this extension can be found in [4].

Sound and complete tableau system for **mbC** and **mCi** obtained by using a general method for constructing tableau systems were presented in [12]. However, those systems are not **KE** systems. To be a **KE** system they should have only one branching rule – each system has 5 branching rules. As explained in [1], branching rules lead to inefficiency. To obtain more efficient proof system, we used those tableau systems as a basis to devise original **mbC** and **mCi** **KE** systems.

The rules for the **mbC** **KE** system are presented in Figure 3. As the **mCi** **KE** system is an extension of the **mbC** **KE** system, only its additional rules are shown in Figure 4. We have proved (see [4]) that the **mbC** **KE** system is analytic, correct and complete. And that the **KE** system for **mCi** is correct and complete.

4 Design and Implementation

A **KEMS** strategy is responsible, among other things, for: (i) choosing the next rule to be applied, (ii) choosing the formula on which to apply the (PB) rule, and (iii)

$\frac{\text{T } A \rightarrow B}{\text{T } A} \quad (\text{T} \rightarrow_1)$	$\frac{\text{T } A \rightarrow B}{\text{F } B} \quad (\text{T} \rightarrow_2)$	$\frac{\text{F } A \rightarrow B}{\text{T } A} \quad (\text{F} \rightarrow)$
$\frac{\text{F } A \wedge B}{\text{T } A} \quad (\text{F} \wedge_1)$	$\frac{\text{F } A \wedge B}{\text{T } B} \quad (\text{F} \wedge_2)$	$\frac{\text{T } A \wedge B}{\text{T } A} \quad (\text{T} \wedge)$
$\frac{\text{T } A \vee B}{\text{F } A} \quad (\text{T} \vee_1)$	$\frac{\text{T } A \vee B}{\text{F } B} \quad (\text{T} \vee_2)$	$\frac{\text{F } A \vee B}{\text{F } A} \quad (\text{F} \vee)$
$\frac{\text{T } \neg A}{\text{T } \circ A} \quad (\text{T} \neg')$ $\frac{\text{F } A}{\text{F } A}$	$\frac{\text{F } \neg A}{\text{T } A} \quad (\text{F} \neg)$	
$\frac{}{\text{T } A \mid \text{F } A} \quad (\text{PB})$		

Fig. 3. mbC KE rules.

$\frac{\text{T } \neg(\circ A)}{\text{T } A} \quad (\text{T} \neg \circ)$	$\frac{\text{F } \circ(\neg^n(\circ A))}{\times} \quad \text{for } (n \geq 0) \quad (\text{F} \circ \neg^n \circ)$
---	--

Fig. 4. Additional KE rules for mCi.

verifying branch closure. In other provers, these features can be scattered in several prover modules if strategies are not designed as first-class citizens.

In **KEMS**, strategies are first-class citizens. This is **KEMS** most important feature. In **KEMS**, the core of the implementation is shared by all strategies and each strategy is defined in one main class and possibly some auxiliary classes and aspects. In this way we can prove the same problem with several different strategies and compare the proofs obtained.

KEMS current version implements strategies for three logics: **CPL**, **mbC** and **mCi**. An overview of the implemented strategies is presented in Table 1.

To solve a problem with a strategy we must also choose a sorter. The order in which s-formulas are analyzed is an aspect that can have a strong influence on a strategy performance. At every moment in the proof search procedure, the prover has a list of not-analyzed s-formulas in the current node from which it is going to choose the next formula to be analyzed. As the s-formulas in the beginning of the list are analyzed first, if we sort the s-formulas we can change the strategy behavior. Some of the implemented sorters are described in Table 2.

Strategy	Main feature
CPL Simple Strategy	Keeps formula reference data structures in memory
CPL Memory Saver Strategy	Does not keep formula reference data structures in memory
CPL Backjumping Simple Strategy	Implements the backjumping technique
CPL Learning Strategy	Implements a learning technique
CPL Comb Learning Strategy	Implements the comb learning technique
CPL Configurable Strategy	Allows a higher control over which formulas are analyzed first
mbC Simple Strategy	An extension of CPL Simple Strategy for mbC
mbC Extended Strategy	An extension of mbC Simple Strategy that applies derived rules before applying (PB)
mCi Simple Strategy	An extension of CPL Simple Strategy for mCi
mCi Extended Strategy	An extension of mCi Simple Strategy that applies derived rules before applying (PB)

Table 1. Overview of **KEMS** Strategies.

Sorter	Description
Insertion Order	least recently inserted s-formulas are analyzed first
Reverse Order	most recently inserted s-formulas are analyzed first
And	s-formulas with ‘ \wedge ’ as main connective are analyzed first
True	s-formulas with T as sign are analyzed first
Increasing	smaller s-formulas are analyzed first
Decreasing	bigger s-formulas are analyzed first

Table 2. Some of **KEMS** sorters.

5 Evaluation

Theorem provers are usually compared by using benchmarks [16]. We have chosen to evaluate **KEMS** using as benchmarks some families of difficult problems [3,17], some of which are well known, and some new families we developed to test **KEMS**.

We present below two of the families we have developed to evaluate **mbC** and **mCi** strategies (other 5 families can be found in [4]). We had two objectives in mind when we devised problem families to evaluate **mbC** and **mCi** theorem provers. First, to obtain families of valid problems whose **KE** proofs were as complex as possible. And second, to devise problems which required the use of many, if not all, **KE** rules of the **KE** systems for **LFI**s.

The first family (Φ^1) of valid sequents for **mbC**. In this family all **mbC** connectives from the Σ° signature are used.

The sequent to be proved for the n -th instance (Φ_n^1) of this problem is:

$$\bigwedge_{i=1}^n (\neg A_i), \bigwedge_{i=1}^n ((\circ A_i) \rightarrow A_i), [\bigvee_{i=1}^n (\circ A_i)] \vee (\neg A_n \rightarrow C) \vdash C \quad (1)$$

The second family of problems (Φ^2) is a variation over the first family whose proofs can be exponential. The sequent to be proved for the n -th instance of this family (Φ_n^2) is:

$$\bigwedge_{i=1}^n (\neg A_i), [\bigwedge_{i=1}^n [(\circ A_i) \rightarrow ((\bigvee_{j=i+1}^n \circ A_j) \vee ((\neg A_n) \rightarrow C))]], [\bigvee_{i=1}^n (\circ A_i)] \vee (\neg A_n \rightarrow C) \vdash C$$

5.1 Results Obtained

We exhibit below some of the results obtained by **KEMS** on a Pentium IV machine with a 3.20G Hz processor and 3775MB memory running Linux version 2.6.15-26-386.

The most important prover configuration parameters for our evaluation were the strategy and the sorter. The reason is that we have noticed through our experiments that these two are the parameters that most affect a prover configuration performance. For this reason, in the following we will refer to a prover configuration as a strategy-sorter pair, or simply a pair.

For instance, the biggest second family instance solved by **mbC** pairs within the time limit was Φ_{14}^2 , whose size is 472. And the biggest instance solved by all pairs was Φ_{10}^2 (whose size is 278). In Table 3 we can see the results obtained by the only two

pairs that solved the biggest solved instance. The interesting fact is that both use the same sorter: Reverse Insertion Order. In Table 4 we show some results obtained with the biggest instance solved by all pairs (Φ_{10}^2). The difference in time between the best and worst pairs is more than 12 times. And the difference in size between the best and worst pairs is more than 15 times.

Pair	Time spent	Proof Size	Comments
<MBCSS,rev>	25213	57827	best in time
<MBCES,rev>	26297	57827	second best in time

Table 3. mbC Φ_{14}^2 results table.

Pair	Time spent	Proof Size	Comments
<MBCES,rev>	2397	9769	best in time
<MBCSS,rev>	2401	9769	second best in time
<MBCSS,inc>	10984	26565	third best in time
<MBCES,and>	30573	116037	worst in time
<MBCSS,imp>	23767	149504	worst in size
<MBCES,imp>	24401	149504	worst in size

Table 4. mbC Φ_{10}^2 results table.

In Chapter E of [4] we have presented all evaluation results obtained by **KEMS** with the families of problems using **KEMS** strategies and sorters. In this way we have provided the first benchmark results for **LFI** theorem provers.

6 Related work

The idea of having several strategies implemented in the same prover and being able of varying the strategy used is not new: in [18], in the context of first-order theorem proving, this idea is clearly present.

Some object-oriented tableau-based provers for propositional logics have implemented multiple strategies. The prover developed by Wagner Dias [19], was written in C++ and implements Analytic and KE propositional tableaux methods. jTAP [20] is a propositional tableau prover, written in Java, based on the method of signed Analytic Tableaux. Both systems have some strategies implemented and can be extended with new ones, but strategies are not well modularized since one has to create subclasses of one or more classes of the system, as well as modify existing ones, to implement a new strategy.

LOTREC [21] is a generic tableau prover for modal and description logics (MDLs). It aims at covering all logics having possible worlds semantics, in particular MDLs. In

LOTREC, strategies are described using a very simple language, not in a programming language. They are limited to establishing the order and the number of times the rules will be applied.

In **KEMS** strategies are represented as Java classes (as in jTAP). Some strategies require auxiliary classes to be implemented. The ideal situation would be that each strategy were composed of other classes representing features of strategies (for instance, order of application of rules). That is not available in **KEMS** current version. Notwithstanding, **KEMS** can be used to provide effective provers for many different logical systems, as well as to enable the comparison between strategies for these systems. That is, the strategies already implemented can be used with other logical systems (if one implements them in **KEMS**) and other strategies can be implemented for the logical systems already represented in **KEMS**.

7 Conclusion and Further Work

We succeeded in developing a multi-strategy theorem prover where we can vary the strategy without modifying the core of the implementation. **KEMS** allows us to describe several proof strategies for the same logical system, and to implement different logical systems.

On the logical side, it would be useful to have a general procedure for automatically generating correct and complete **KE** systems for **LFI**s and other logical systems, similar to the procedure for generating tableau systems presented in [22]. This could help us to extend **KEMS** to other logical systems. Marcelo Coniglio (one of the authors of [22]) informed us that they have already thought of adapting their method that obtains what we called here **C³M** systems (which for **LFI**s are a kind of mixture of **AT** and **KE**) to be able to produce **KE** systems.

On the implementation side, from the results we have obtained⁵, we can see that it would be very useful to have an adaptive strategy that changes its behavior according to features of the problem presented to it. This strategy can behave as other implemented strategies and it will be able to vary its actions not only for different problems but also for different subproblems of the same problem given as input.

References

1. D'Agostino, M.: Tableau methods for classical propositional logic. In: Handbook of Tableau Methods. Kluwer Academic Press (1999) 45–123
2. Cook, S.: The importance of the P versus NP question. J. ACM **50**(1) (2003) 27–29
3. Carbone, A., Semmes, S.: Graphic Apology for Symmetry and Implicitness. Oxford University Press (2000)
4. Neto, A.: A Multi-Strategy Tableau Prover. PhD thesis, University of São Paulo (2007) <http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04052007-175943/>. Last accessed, June 6th, 2008.
5. Neto, A., Finger, M.: A KE tableau for a logic of formal inconsistency. In: Proceedings of TABLEAUX'07 position papers and Workshop on Agents, Logic and

⁵ That is, from the conclusion that there is no strategy-sorter pair which is the best for every problem family.

- Theorem Proving. Technical Report (LSIS.RR.2007.002) of the LSIS/Université Paul Cézanne, Marseille, France (2007)
6. Neto, A., Finger, M.: Effective Prover for Minimal Inconsistency Logic. In: Artificial Intelligence in Theory and Practice. IFIP, Springer Verlag (2006) 465–474 Available at <http://www.springerlink.com/content/b80728w7m6885765>. Last accessed, November 2006.
 7. Neto, A., Finger, M.: Implementing a multi-strategy theorem prover. In Garcia, A.C.B., Osório, F.S., eds.: 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.
 8. Neto, A., Finger, M.: 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.
 9. Neto, A., Finger, M.: 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.
 10. Neto, A., Finger, M.: 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.
 11. D’Ottaviano, I.M.L., de Castro, M.A.: 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) (2005) 69–103
 12. Carnielli, W., Coniglio, M.E., Marcos, J.: Logics of Formal Inconsistency. In: Handbook of Philosophical Logic. Volume 14. Springer (2007) 15–107
 13. de Amo, S., Carnielli, W., Marcos, J.: A Logical Framework for Integrating Inconsistent Information in Multiple Databases. In: LNCS. Volume 2284., Springer-Verlag. (2002) 67–84
 14. D’Agostino, M., Mondadori, M.: The taming of the cut: Classical refutations with analytic cut. Journal of Logic and Computation (1994) 285–319
 15. Massacci, F.: Simplification: A general constraint propagation technique for propositional and modal tableaux. In: TABLEAUX ’98, Springer-Verlag (1998) 217–231
 16. Sutcliffe, G., Suttner, C.: The CADE ATP System Competition (2003) <http://www.cs.miami.edu/~tptp/CASC>. Last accessed, March 2005.
 17. Pelletier, F.J.: Seventy-five problems for testing automatic theorem provers. J. Autom. Reason. **2**(2) (1986) 191–216
 18. Manning, A., Ireland, A., Bundy, A.: Increasing the Versatility of Heuristic Based Theorem Provers. In: LPAR’93. (1993)
 19. Dias, W.: Tableaux implementation for approximate reasoning (in portuguese). Master’s thesis, Computer Science Department, Institute of Mathematics and Statistics, University of São Paulo (2002)
 20. Beckert, B., Bubel, R., Habermalz, E., Roth, A.: jTAP - a Tableau Prover in Java (February 1999) Universität Karlsruhe. Available at <http://i12www.ira.uka.de/~aroth/jTAP/>. Last accessed, November 2006.
 21. Fariñas del Cerro, L., Fauthoux, D., Gasquet, O., Herzig, A., Longin, D., Massacci, F.: Lotrec: the generic tableau prover for modal and description logics. In: IJCAR, Springer Verlag (2001)
 22. Caleiro, C., Carnielli, W., Coniglio, M.E., Marcos, J.: Two’s company: “The humbug of many logical values”. In: Logica Universalis. Birkhäuser Verlag, Basel, Switzerland (2005) 169–189