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 D

KEMS Evaluation

Theorem provers are usually compared by using benchmarks [112]. SATLIB [102] (for **CPL**) and TPTP [111] (for first-order classical logic) are two web sites that contain benchmark problems to evaluate theorem provers. We have chosen to evaluate **KEMS** using as benchmarks some families of difficult problems [12, 95], some of which well known and some new families we developed to test **KEMS**. We present below the families we used to evaluate **CPL**, **mbC** and **mCi** strategies.

D.1 Problem Families

A *problem family* is a set of problems that we know, by construction, whether they are valid, satisfiable or unsatisfiable. For any family f we have a procedure such that, for any $n \in \mathbb{N}$, $n \geq 1$, we construct an instance f_n of this family.

In the problem families described below, formulas such as $\bigwedge_{i=m}^{n} A_i$ (an iterated conjunction) and $\bigvee_{i=m}^{n} A_i$ (an iterated disjunction) may appear, where $m, n \in \mathbb{N}$. If $m = n$, then both reduce to A_m. If $m < n$, then the first reduces to $(A_m \wedge (A_{m+1} \wedge (\ldots \wedge (A_{n-1} \wedge A_n))))$ and the second to $(A_m \vee (A_{m+1} \vee (\ldots \vee (A_{n-1} \vee A_n))))$. And if $m > n$, then the first formula is an empty conjunction (which corresponds to the $\top$ formula) and the second is an empty disjunction (which corresponds to the $\bot$ formula). Besides that, whenever we have $(A \wedge \top)$, $(\top \wedge A)$, $(A \vee \bot)$ or $(\bot \vee A)$, for any formula A, we replace any of these formulas by A.

D.1.1 CPL Problem Families

The problem families used to evaluate **KEMS CPL** strategies contain explicit propositional valid sequents whose proofs in Sequent Calculus [56], Analytic Tableaux [106] or Resolution [100] can be exponential. For instance, zChaff [53], one of the fastest SAT solvers available, takes a couple of days to prove the instance number 14 of the PHP family on a personal computer.

Γ Problems

The n-th instance of the Γ family [12] has $2n$ propositional variables and $2n + 2$ formulas. It is possible to find both exponential and non-exponential (in n) proofs of instances of this family using **AT**. For the n-th instance (Γ_n), the sequent to be proved is:

$$p_1 \vee q_1, C_n \vdash p_{n+1} \vee q_{n+1}$$

where

$$C_n = \{p_i \rightarrow (p_{i+1} \vee q_{i+1}), q_i \rightarrow (p_{i+1} \vee q_{i+1}) | 1 \leq i \leq n\}$$

H Problems

For this family, the sequent to be proved for the n-th instance is

$$\vdash H_n$$

where H_n is constructed in the following way:

$$\begin{aligned}
H_1 &= p_1 \vee \neg p_1 \\
H_2 &= (p_1 \wedge p_2) \vee (p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2) \vee (\neg p_1 \wedge \neg p_2) \\
H_3 &= (p_1 \wedge p_2 \wedge p_3) \vee \ldots \vee (\neg p_1 \wedge \neg p_2 \wedge \neg p_3) \\
&\vdots
\end{aligned}$$

These formulas (also called “truly fat” formulas) were defined in [28] and used to prove that the Analytic Tableaux method cannot polynomially simulate the truth-table

method[1]. Each instance of this family has n propositional variables, but the size of the formula H_n grows exponentially in n (because any instance has 2^n clauses $q_1 \wedge \ldots \wedge q_n$, where each q_i is either equal to p_i or $\neg p_i$).

Statman Problems

Statman formulas [108] can be constructed as follows. Consider

$$A_k = \bigwedge_{j=1}^{k}(p_j \vee q_j)$$
$$B_1 = p_1$$
$$C_1 = q_1$$

and, inductively:

$$B_{i+1} = A_i \rightarrow p_{i+1} \qquad C_{i+1} = A_i \rightarrow q_{i+1}$$

The sequent to be proved for the n-th instance of this family is:

$$B_1 \vee C_1, \ldots, B_n \vee C_n \vdash p_n \vee q_n$$

Statman has proved that this family of formulas has polynomial proofs in sequent calculus if we use the cut rule, but only exponential size cut-free proofs [12].

Pigeon Hole Principle Problems

The Pigeon Hole Principle (PHP) [95, 12] states that given $n-1$ pigeon holes and n objects to be put in these holes, there is always one hole that will receive at least two objects.

The sequent to be proved is

$$\vdash \mathrm{PHP}_n$$

where

$$\mathrm{PHP}_n = A_n \rightarrow B_n$$

[1]And this problem family has a structure which is very similar to the structure of the problems used in [22] (and commented in [76]) to analyze the complexity of Analytic Tableaux.

and

$$A_n = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n-1} p_{i,j}$$

$$B_n = \bigvee_{i=1}^{n-1} \bigvee_{k=i}^{n} \bigvee_{j=1}^{n-1} (p_{i,j} \wedge p_{k,j})$$

where $p_{i,j}$ expresses that object number i was inserted in hole number j. The A_n formula states that every object goes to some hole and the B_n formula that at least one hole receives 2 objects.

This problem leads to a lot of branching in Analytic Tableaux and Sequent Calculus. In Sequent Calculus, cut-free proofs are exponential but there is a very complicated polynomial proof with cut [9].

U Problems

The U family was described in [95]. It is stated there that resolution system proofs of this problem instances increase exponentially with n. The sequent to be proved for the n-th instance of this family is $\vdash U_n$, where U_n is defined as follows:

$$\begin{aligned}
U_1 &: \quad (P_1 \leftrightarrow P_1) \\
U_2 &: \quad (P_1 \leftrightarrow (P_2 \leftrightarrow (P_1 \leftrightarrow P_2))) \\
U_3 &: \quad (P_1 \leftrightarrow (P_2 \leftrightarrow (P_3 \leftrightarrow (P_1 \leftrightarrow (P_2 \leftrightarrow P_3))))) \\
&\vdots \\
U_n &: \quad (P_1 \leftrightarrow (P_2 \leftrightarrow (P_3 \leftrightarrow \ldots \leftrightarrow (P_n \leftrightarrow (P_1 \leftrightarrow P_2 \ldots \leftrightarrow P_n) \ldots)
\end{aligned}$$

Square Tseitin Problems

We present below a description (found in [95]) of the arbitrary graph problems due to Tseitin:

> Consider a graph with the edges labelled. (...) Assign 0 or 1 arbitrarily to nodes of the graph. For each node of the graph, we associate a set of clauses as follows:

1. every label of an edge emanating from that node will occur in each clause (of the set of clauses generated from that node);
2. if the node is assigned 0, then the number of negated literals in each of the generated clauses is to be odd. Generate all such clauses for that node;
3. if the node is assigned 1, then the number of negated literals in each of the generated clauses is to be even. Generate all such clauses for that node.

Tseitin's result is this: the sum (mod 2) of the 0's and 1's assigned to the nodes of the graph equals 1 if and only if the set of all generated clauses is inconsistent.

A rather obvious subset of Tseitin's problems is what we call here Square Tseitin (ST) problems. The ST_n instance is constructed (using Tseitin procedure) from a graph that resembles a matrix, with n lines and n columns of nodes. Every node is connected with its four closest neighbors: left, right, top and bottom[2]. The top left node is assigned 0 and the other nodes are assigned 1. The result is an inconsistent set of clauses. By negating this set of clauses we obtain a valid sequent. We will not describe the exact procedure here; we only present ST_1, which is the following sequent: $\vdash (V_{00} \leftrightarrow H_{00}) \vee ((V_{01} \oplus H_{00}) \vee ((V_{00} \oplus H_{10}) \vee (V_{01} \oplus H_{10})))$. An alternative version of this instance is: $(V_{00} \oplus H_{00}) \vdash (V_{01} \oplus H_{00}), (V_{00} \oplus H_{10}), (V_{01} \oplus H_{10})$.

Backjumping PHP Problems

To test the performance of strategies in presence of irrelevant premises, we devised a schema to generate what we called backjumping versions of any family. For instance, for each PHP family instance we can generate one or more backjumping versions (B_PHP) of these problems. The idea is to include in the beginning of each instance one or more

[2] Obviously, some nodes do not have one or more of these neighbors.

signed formulas that are not relevant, such as:

$$A_{i,1} \oslash A_{i,2}$$

or

$$\neg(A_{i,1} \oslash A_{i,2})$$

where $i \in \mathbb{N}^+$ and $\oslash \in \{\wedge, \vee, \rightarrow\}$.

For instance, if we add three irrelevant formulas to PHP_4, we obtain the following B_PHP_4^3 instance:

$$\neg(A_{1,1} \wedge A_{1,2}), (A_{2,1} \vee A_{2,2}), (A_{3,1} \rightarrow A_{3,2}) \vdash \text{PHP}_4$$

where we know that no $A_{i,j}$ for any i or j appears in PHP_4.

We used backjumping versions of PHP to evaluate **KEMS**.

Random SAT Problems

Random K-SAT formulas were presented in [82] as a class of random formulas to be used as a benchmark for **CPL** satisfiability-testing procedures. To generate a problem instance of this class one must provide values for three parameters: number of propositional symbols, number of clauses and length of each clause. The generated set of clauses can be satisfiable or not. We have generated six instances of random K-SAT problems (see Table D.1) to test **KEMS**. To verify unsatisfiability of a set of clauses $\{C_1, C_2 \ldots C_n\}$ we try to prove whether $(C_1 \wedge C_2 \wedge \ldots \wedge C_n) \vdash \perp$ is valid.

name	propositional variables	clauses	clause length	size	**CPL** satisfiability
cnf_1	10	30	3	223	satisfiable
cnf_2	10	70	3	526	unsatisfiable
cnf_3	20	70	3	517	satisfiable
cnf_4	10	140	3	1042	unsatisfiable
cnf_5	20	140	3	1043	unsatisfiable
cnf_6	30	140	3	1066	unsatisfiable

Table D.1: Random K-SAT problems.

D.1.2 LFI Problem Families

We present below the problem families we devised to evaluate **mbC** and **mCi** theorem provers. We had two objectives in mind. 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. These families are not classically valid, since their formulas are in $For^{\circ\bullet}$. However, if we define $\circ X \stackrel{\text{def}}{=} \top$ and $\bullet X \stackrel{\text{def}}{=} \bot$ in **CPL**[3], then all families become **CPL**-valid and can be used for evaluating a **CPL** prover.

The **LFI** families can be divided into two groups: first to fourth families are valid in **mbC** (and also in **mCi**, since this logic extends **mbC**) while seventh to ninth families were designed as valid **mCi** problems. This strange numbering is due to historical reasons (fifth and sixth families are not presented here).

Note: in [18] we can find many simple problems that can be used to evaluate the correctness of provers for **LFI**s, including the so-called contraposition rules. We have used all these problems to evaluate **KEMS**. They were useful to evaluate correctness, but not to evaluate performance since they are rather small.

First family

Here we present 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^1_n) of this problem is:

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

The sequent in (D.1) means that, for every $1 \leq i \leq n$, the following set of assumptions has C as a logical consequence:

1. all $\neg A_i$ are true;
2. if $\circ A_i$ is true, then A_i is also true;
3. either one of the $\circ A_i$s or $\neg A_n \rightarrow C$ is true.

[3] A natural way of extending **CPL** presented in [18].

The explanation for this family is the following: suppose we are working with a system that allows inconsistent information representation. The A_i fact means that someone expressed an opinion A about an individual i and $\neg A_i$ means that someone expressed an opinion $\neg A$ about this same individual. For instance, if A means that a person is nice, $\neg A_3$ means that at least one person finds 3 is not nice, and A_4 means that at least one person finds 4 nice. Then $\circ A_i$ means that either all people think i is nice, or all people think i is not nice, or there is no opinion A recorded about i. '$\circ A_i \rightarrow A_i$' means that if all opinions about a person are the same, then that opinion is A.

For a subset of individuals numbered 1 to n, we have $\neg A_i$ and $\circ A_i \rightarrow A_i$ for all of them. From the fact that either $\neg A_n \rightarrow C$ or for one of them we have $\circ A_i$, we can conclude C.

It is easy to obtain polynomial **mbC KE** proofs for this family of problems. The proofs use most (but not all) **mbC KE** rules and have a comb-like form (see Figure D.1). In this figure, the shown proof of Φ_n^1 closes left branches with the following sequence of signed formulas, which we call $\Phi_C^1(i)$:

$$\begin{array}{l} \mathtt{T}\ \circ A_i \\ \mathtt{T}\ A_i \\ \mathtt{F}\ A_i \\ \quad \times \end{array}$$

Thus, a proof of Φ_n^1 is a $\Phi_p^1(1)$ structure, where $\Phi_p^1(i)$ is depicted below:

1. $\Phi^1_p(1)$ is:

$$\Phi^1_n$$
$$\vdots$$
$$\mathtt{T}\,\neg A_1$$
$$\vdots$$
$$\mathtt{T}\,\neg A_n$$
$$\mathtt{T}\,(\circ A_1) \rightarrow A_1$$
$$\vdots$$
$$\mathtt{T}\,(\circ A_n) \rightarrow A_n$$
$$\Phi^1_C(1) \qquad \Phi^1_p(2)$$

2. For $1 < i \leq n$, $\Phi^1_p(i)$ is defined as:

$$\mathtt{F}\ \circ A_{i-1}$$
$$\mathtt{T}\ [\bigvee_{j=i}^{n}(\circ A_j)] \vee ((\neg A_n) \rightarrow C)$$
$$\Phi^1_C(i) \qquad \Phi^1_p(i+1)$$

3. And when $i = n + 1$, $\Phi^1_p(i)$ is:

$$\mathtt{F}\ \circ A_n$$
$$\mathtt{T}\ (\neg A_n) \rightarrow C$$
$$\mathtt{F}\ \neg A_n$$
$$\times$$

Second Family

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^2_n) is:

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

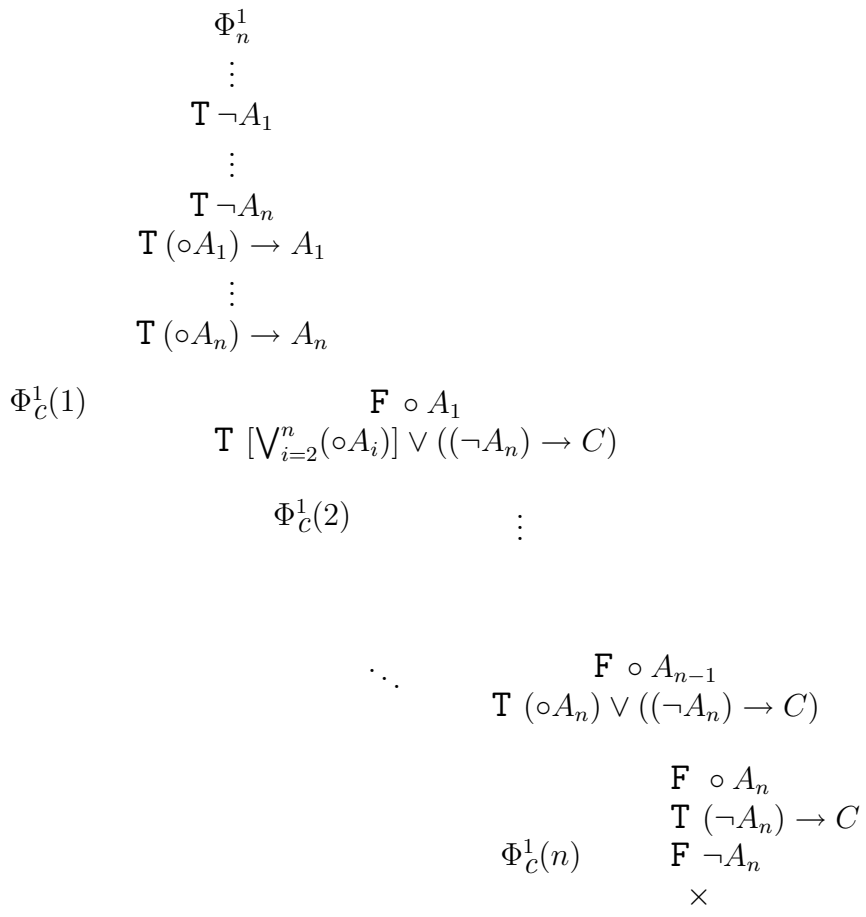
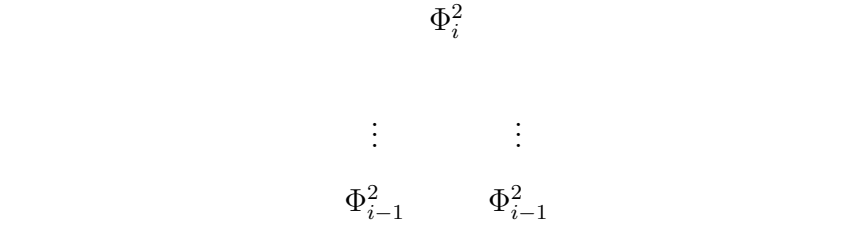


Figure D.1: A proof of Φ^1_n.

In this family, instead of closing left branches easily on $\Phi^1_C(i)$, we obtain two child branches with the same difficulty. Our objective when designing this family was to obtain a problem whose proof contained two proofs of the immediately smaller instance of the same problem:



To achieve this, we had to replace the conjunction of $(\circ A_i) \to A_i$ in (D.1) by a conjunction of

$$(\circ A_i) \to ([\bigvee_{j=i+1}^{n} \circ A_j] \vee ((\neg A_n) \to C)) \tag{D.2}$$

This means that for every person numbered 1 to n, if all opinions about a person are the

same, then either all opinions about some other person with a higher index are the same or $(\neg A_n) \rightarrow C)$ is true.

The proof of Φ^2_n is

$$\Phi^2_n$$
$$\mathtt{T}\,(\neg A_1) \wedge (\neg A_2) \wedge \ldots \wedge (\neg A_n)$$
$$\Phi^2 p(1)$$

Where, for $1 \leq i < n$, $\Phi^2 p(i)$ is:

$$\mathtt{T} \circ A_i \qquad\qquad \mathtt{F} \circ A_i$$
$$\mathtt{T}\,([\bigvee_{j=i+1}^{n} \circ A_j] \vee ((\neg A_n) \rightarrow C)) \qquad\qquad \mathtt{T}\,([\bigvee_{j=i+1}^{n} \circ A_j] \vee ((\neg A_n) \rightarrow C))$$
$$\Phi^2 p(i+1) \qquad\qquad \Phi^2 p(i+1)$$

Finally, $\Phi^2 p(n)$ is

$$\mathtt{T} \circ A_n \qquad\qquad \mathtt{F} \circ A_n$$
$$\mathtt{T}\,(\neg A_n) \rightarrow C \qquad\qquad \mathtt{T}\,(\neg A_n) \rightarrow C$$
$$\mathtt{F}\,\neg A_n \qquad\qquad \mathtt{F}\,\neg A_n$$
$$\times \qquad\qquad \times$$

In Figure D.2 we show a proof of Φ^2_3. The right branch (which we omitted) is a copy of the left branch except that, in the beginning, instead of $\mathtt{T} \circ A_1$ we have $\mathtt{F} \circ A_1$.

Third Family

With the third family of problems we intended to develop a family whose instances required the application of all **mbC KE** rules. To obtain an instance of the third family (Φ^3), we have to make the following changes to an instance of the second family:

1. replace C in the right-side of the sequent by $C' \rightarrow (C'' \vee C)$;

$$\dfrac{\begin{array}{c}\mathtt{T}\,(\neg A_1)\wedge(\neg A_2)\wedge(\neg A_3)\\ \mathtt{T}\,[(\circ A_1)\rightarrow((\circ A_2)\vee(\circ A_3)\vee((\neg A_3)\rightarrow C))]\wedge\\ [(\circ A_2)\rightarrow((\circ A_3)\vee((\neg A_3)\rightarrow C))]\wedge\\ [(\circ A_3)\rightarrow(((\neg A_3)\rightarrow C))]\\ \mathtt{T}\,(\circ A_1)\vee(\circ A_2)\vee(\circ A_3)\vee((\neg A_3)\rightarrow C)\\ \mathtt{F}\,C\end{array}}{\begin{array}{c}\mathtt{T}\,\neg A_1\\ \mathtt{T}\,\neg A_2\\ \mathtt{T}\,\neg A_3\end{array}}$$

$$\dfrac{\mathtt{T}\circ A_1}{\mathtt{T}\,(\circ A_2)\vee(\circ A_3)\vee((\neg A_3)\rightarrow C)} \qquad \dfrac{\mathtt{F}\circ A_1}{\mathtt{T}\,(\circ A_2)\vee(\circ A_3)\vee((\neg A_3)\rightarrow C)}$$

$$\vdots$$

$$\dfrac{\mathtt{T}\circ A_2}{\mathtt{T}\,(\circ A_3)\vee((\neg A_3)\rightarrow C)} \qquad \dfrac{\mathtt{F}\circ A_2}{\mathtt{T}\,(\circ A_3)\vee((\neg A_3)\rightarrow C)}$$

$$\dfrac{\mathtt{T}\circ A_3}{\mathtt{T}\,(\neg A_3)\rightarrow C}\quad\dfrac{\mathtt{F}\circ A_3}{\mathtt{T}\,(\neg A_3)\rightarrow C}\quad\dfrac{\mathtt{T}\circ A_3}{\mathtt{T}\,(\neg A_3)\rightarrow C}\quad\dfrac{\mathtt{F}\circ A_3}{\mathtt{T}\,(\neg A_3)\rightarrow C}$$

$$\dfrac{\mathtt{F}\,\neg A_3}{\times}\quad\dfrac{\mathtt{F}\,\neg A_3}{\times}\quad\dfrac{\mathtt{F}\,\neg A_3}{\times}\quad\dfrac{\mathtt{F}\,\neg A_3}{\times}$$

Figure D.2: A proof of Φ^2_3.

2. replace $[\bigvee_{j=i+1}^{n}\circ A_j]\vee((\neg A_n)\rightarrow C)$ in $(\bigwedge_{i=1}^{n}((\circ A_i)\rightarrow([\bigvee_{j=i+1}^{n}\circ A_j]\vee((\neg A_n)\rightarrow C))))$ (which is left-associated on $\vee$) by a right-associated $(((\neg A_n)\wedge U_l)\rightarrow C)\vee[\bigwedge_{j=i+1}^{n}\circ A_j]$;

3. in (D.2), replace $(\neg A_n)\rightarrow C$ by $(U_r\wedge(\neg A_n))\rightarrow C$;

4. add $U_l\wedge U_r$ to the left-side of the sequent.

The result is the following sequent (Φ^3_n):

$$\begin{array}{l}U_l\wedge U_r,\\ \bigwedge_{i=1}^{n}(\neg A_i),\\ \bigwedge_{i=1}^{n}[(\circ A_i)\rightarrow\ ((((\neg A_n)\wedge U_l)\rightarrow C)\vee\bigvee_{j=i+1}^{n}\circ A_j)],\\ (\bigvee_{i=1}^{n}\circ A_i)\vee\ ((U_r\wedge(\neg A_n))\rightarrow C)\\ \vdash\ C'\rightarrow(C''\vee C)\end{array}$$

Fourth Family

This is a family where negation appears only in the conclusion. The sequent to be proved (Φ^4_n) is:

$$\bigwedge_{i=1}^{n}(A_i), \bigwedge_{i=1}^{n}((A_i \vee B_i) \rightarrow (\circ A_{i+1})), [\bigwedge_{i=2}^{n}(\circ A_i)] \rightarrow A_{n+1} \vdash \neg\neg A_{n+1}$$

The formulas of this family can be explained as follows. We have two formulas to represent two types of opinion: A and B. First we assume A_i for every i from 1 to n. Then we suppose for all j from 1 to n that ($A_i \vee B_j$ implies that $\circ A_{j+1}$. And finally we assume that for every k from 2 to n the conjunction of $\circ A_k$'s implies A_{n+1}. It is easy to see that from these assumptions we can deduce A_{n+1}. So we can also deduce its double negation: $\neg\neg A_{n+1}$.

Seventh family

The seventh family (Φ^7) was designed for testing **mCi** provers. Notwithstanding, if we use the definition $\bullet A \stackrel{\text{def}}{=} \neg \circ A$, it is also valid in **mbC**. The sequent to be proved (Φ^7_n) is:

$$\bigwedge_{i=1}^{n}(A_i), \bigwedge_{i=1}^{n}(B_i \rightarrow (\neg A_i)), \bigvee_{i=1}^{n}(\circ A_i) \vdash \bigwedge_{i=1}^{n}((\bullet A_i) \vee (\neg B_i))$$

In this sequent, the $\bigvee_{i=1}^{n}(\circ A_i)$ formula is actually not essential to arrive at the conclusion. Therefore, we can define a variant (called $\Phi^{7'}$) of this family where this formula does not appear. The Φ^7 family is probably more difficult to prove because of the irrelevant premise.

Eighth family

The eighth family (Φ^8) was also designed as a valid **mCi** problem family. This family is not valid for **mbC**[4]. The sequent to be proved (Φ^8_n) is the following:

$$\bigvee_{i=1}^{n}(\bullet A_i), \bigwedge_{i=1}^{n}(A_i \rightarrow (\neg B_i)), \bigwedge_{i=1}^{n}((\neg A_i) \rightarrow C_{(n-i+1)}) \vdash \bigwedge_{i=1}^{n}((\neg B_i) \wedge C_{(n-i+1)})$$

[4]Not even if we define the inconsistency ('$\bullet$') connective in the standard way.

Ninth family

The ninth family (Φ^9) is also a **mCi** valid family which is not **mbC**-valid[5]. The sequent to be proved (Φ^9_n) is:

$$\bigvee_{i=1}^{n}(\circ A_i), \bigwedge_{i=1}^{n}(B_i \rightarrow (\bullet A_i)) \vdash \bigvee_{i=1}^{n} \neg((\circ(\circ A_i)) \rightarrow B_i)$$

We can have several valid variations of this family, for $m \geq 0$ and $p \geq 0$:

$$\bigvee_{i=1}^{n}(\neg^m(\circ A_i)), \bigwedge_{i=1}^{n}(B_i \rightarrow (\neg^m(\bullet A_i))) \vdash \bigvee_{i=1}^{n} \neg((\circ(\neg^p(\circ A_i))) \rightarrow B_i)$$

where $(\neg^1 A) \stackrel{\text{def}}{=} (\neg A)$ and $(\neg^n A) \stackrel{\text{def}}{=} (\neg(\neg^{n-1} A))$.

D.2 Results Obtained

In this section we exhibit the results obtained by **KEMS** on the problem families we just presented. All results were obtained on a Pentium IV machine with a 3.20G Hz processor and 3775MB memory running Linux version 2.6.15-26-386. The `java -jar kems.jar` command[6] was issued with the `-Xms200m -Xmx2048m` options to set the initial and maximum heap sizes. This allowed **KEMS** to use more of the computer's main memory than the default memory allocated by the java virtual machine.

As we have pointed out in Section C.3, the user can present to **KEMS** a problem and a prover configuration. For the purpose of evaluation, we have developed a command-line version of **KEMS** that accepts a sequence of problems and a set of prover configurations to be run on the problems. The so-called *sequence files* used to evaluate **KEMS** as well as all results obtained are available on [92].

For each prover configuration, the first two parameters are the logical system and the analyzer. In our sequence files, the logical system could be **CPL**, **mbC** or **mCi**, the three logical systems implemented in **KEMS** current version.

[5]Also not even if we give the standard definition of the inconsistency connective in **mbC**.

[6]The command used to execute `kems.jar`, which is the java archive that contains **KEMS** executable version.

First, we have evaluated **CPL** prover configurations with all **CPL** problems presented in Section D.1.1 and also with **LFI** problems families[7] (Section D.1.2). For **CPL**, we had problems written in the format defined in [38] and problems written in SATLIB SAT Format [101]. For the format defined in [38] we used the `sats5` analyzer, an analyzer implemented using JFlex [66] for lexical analysis and CUP [62] for syntactical analysis. And for the SATLIB SAT Format we used the `satcnf2` analyzer, implemented using the same technologies.

After that, we evaluated prover configurations for **mbC** and **mCi** with all **LFI** problems families which are valid in at least one of the two logical systems. We extended the format defined in [38] to deal with **LFI** connectives and implemented the `satlfiinconsdef` analyzer, also using JFlex and CUP.

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.

As we had a lot of problems to evaluate, we fixed the 'number of times the prover must run the proof search procedure with a given problem' parameter to one, and the 'time limit for the proof search procedure' parameter to three minutes or 180.000 milliseconds (the time spent is measured by **KEMS** in milliseconds). We set to true the 'save formula origin' option, and to false the 'discard closed branches' option and the 'save discarded branches' option.

In the tables we display below each prover configuration will be represented by a binary tuple:

```
<strategyId,sorterId>
```

where strategyId and sorterId can vary.

These are the ids for strategies:

SS - **CPL** Simple Strategy;

[7] Except the fifth and the sixth.

MSS - **CPL** Memory Saver Strategy;

BSS - **CPL** Backjumping Simple Strategy;

LS - **CPL** Learning Strategy;

CLS - **CPL** Comb Learning Strategy;

CS - **CPL** Configurable Strategy;

MBCSS - **mbC** Simple Strategy;

MBCES - **mbC** Extended Strategy;

MCISS - **mCi** Simple Strategy;

MCIES - **mCi** Extended Strategy.

And these are the ids for sorters:

ins - insertion order;

rev - reverse order;

and - 'and' connective;

or - 'or' connective;

imp - 'implication' connective;

bi-impli - 'bi-implication' connective;

xor - 'exclusive or' connective;

`T` - 'true' sign;

`F` - 'false' sign;

inc - increasing complexity;

dec - decreasing complexity;

nfo - string order;

rfo - reverse string order.

D.2.1 Gamma Family Results

The biggest Γ family instance solved within the time limit was Γ_{360}, whose size is 3606. In Table D.2 we can see the results obtained by some selected pairs with this instance. The worst pair for this instance lasts approximately the same than the best ones. And its proof size is exactly the same size of the best pairs. In fact, several pairs obtained results similar to the time and size results obtained by the best pairs. In Table D.3 we show two results obtained with the biggest instance solved by all pairs (Γ_{10}, whose size

is 106). Several other pairs obtained results very similar to the one obtained by the best and worst pair. It is easy to see that the differences in time (more that 150 times) and size (more that 25 times) between the best and the worst pair are big.

Pair	Time spent	Proof Size	Comments
<LS,inc>	166068	7926	best in time and size
<BSS,inc>	166120	7926	second best in time
<CLS,rfo>	166631	7926	third best in time
<SS,dec>	174465	7926	worst in time

Table D.2: Γ_{360} results table.

Pair	Time spent	Proof Size	Comments
<BSS,T>	7	226	best in time and size
<CLS,and>	7	226	best in time and size
<CLS,imp>	7	226	best in time and size
<CS,rev>	669	3950	an intermediate result
<CS,ins>	1061	5735	worst in time and size

Table D.3: Γ_{10} results table.

D.2.2 H Family Results

The biggest H family instance solved within the time limit was H_6 (whose size is 954) and it was solved by all pairs. The best results were obtained by MSS pairs. In Table D.4 we can see the results obtained by some selected pairs with the biggest instance solved. The worst pair for this instance lasts approximately two times more than the best ones. And its size is approximately two times higher than the size of best pairs.

Pair	Time spent	Proof Size	Comments
<MSS,F>	29313	33345	best in time and size
<MSS,ins>	29314	33345	second best in time
<MSS,dec>	29321	33345	third best in time
<CLS,rfo>	79890	75887	worst in time and size

Table D.4: H_6 results table.

D.2.3 Statman Family Results

The biggest Statman family instance solved within the time limit was Statman_{29}, whose size is 3338. The best results were obtained by MSS pairs; all of them solved the biggest solved instance. In Table D.5 we can see the results obtained by some selected pairs with the biggest solved instance. The worst pair for this instance lasts approximately three times more than the best ones. And the proof sizes of all pairs that solved the biggest instance are equal. In Table D.6 we show the results obtained with the biggest instance solved by all pairs (Statman_9, whose size is 318). It is easy to see that the differences in time (more that 600 times) and size (more that 150 times) are huge.

Pair	Time spent	Proof Size	Comments
`<MSS,F>`	5838	34366	best in time and size
`<MSS,or>`	5840	34366	second best in time
`<MSS,xor>`	5859	34366	third best in time
`<LS,rev>`	15452	34366	worst in time

Table D.5: Statman_{29} results table.

Pair	Time spent	Proof Size	Comments
`<MSS,or>`	37	1256	best in time and size
`<MSS,xor>`	37	1256	second best in time
`<MSS,dec>`	38	1256	third best in time
`<CS,rfo>`	22722	179991	worst in time
`<CS,or>`	21494	192983	worst in size

Table D.6: Statman_9 results table.

D.2.4 PHP Family Results

The best results for the PHP_6 instance (the biggest PHP instance solved within a 3-minute time limit) were obtained by the `<MSS,or>` and `<MSS,T>` pairs, with approximately 9 seconds and a proof size of 21273. No strategy could solve PHP_7 within the time limit. PHP_6 was solved by 43 pairs, 10 of each had MSS, SS, LS and SS as strategy. Only 3 pairs had CS as strategy and none had CLS as strategy. PHP_4 was the biggest instance that was solved by all pairs.

In Table D.7 we can see the results obtained by some selected pairs with the biggest solved instance (PHP_6, whose size is 455). The worst pair for this biggest instance lasts ten times more than the best ones. And its proof size is approximately 7 times bigger than the best sizes.

Pair	Time spent	Proof Size	Comments
<MSS,or>	9067	21273	best in time and size
<MSS,T>	9086	21273	second best in time
<SS,T>	9543	21273	third best in time
<LS,xor>	101522	148743	worst in time and size

Table D.7: PHP_6 results table.

We also tried PHP_7 (whose size is 692) with a 20-minute time limit. The results are presented in Table D.8. Surprisingly, the best pairs solved the biggest solved instance in less than three minutes. Again the best pair was <MSS,or>.

Pair	Time spent	Proof Size	Comments
<MSS,or>	150007	153988	best in time and size
<MSS,rfo>	154587	153988	second best in time and size
<SS,T>	162933	153988	third best in time and size
<BSS,T>	169517	153988	worst in time
<LS,or>	166666	181704	worst in size

Table D.8: PHP_7 results table.

In Table D.9 we can see the results obtained by two pairs with the biggest instance solved by all pairs (PHP_4, whose size is 155). The worst pair for this instance lasts 43 times more than the best ones. And its proof size is approximately 13 times bigger than the best sizes.

Pair	Time spent	Proof Size	Comments
<BSS,or>	70	720	one of the several best in time and size
<CS,F>	3024	9689	worst in time and size

Table D.9: PHP_4 results table.

D.2.5 U Family Results

The biggest U family instance solved within the time limit was U_{13} (whose size is 51). The best results were obtained by MSS pairs; all of them solved the biggest solved instance.

In Table D.10 we can see the results obtained by some selected pairs with the biggest solved instance. The worst pair for this instance lasts approximately 1.5 times more than the best ones. And the proof sizes of all pairs that solved the biggest instance are equal. In Table D.11 we show the results obtained with the biggest instance solved by all pairs (U_8, whose size is 31). It is easy to see how big are the differences in time (more than 400 times) and size (more than 18 times) between the best and worst pairs.

Pair	Time spent	Proof Size	Comments
`<MSS,bi-impli>`	33686	483158	best in time and size
`<MSS,rfo>`	33790	483158	second best in time
`<MSS,T>`	33793	483158	third best in time
`<SS,rev>`	49541	483158	worst in time

Table D.10: U_{13} results table.

Pair	Time spent	Proof Size	Comments
`<MSS,inc>`	314	9554	best in time and size
`<MSS,F>`	319	9554	second best in time
`<MSS,ins>`	319	9554	third best in time
`<CLS,rev>`	134411	174624	worst in time
`<CLS,rfo>`	127820	174880	worst in size

Table D.11: U_8 results table.

D.2.6 Square Tseitin Family Results

The biggest ST family instance solved within the time limit was ST_4, whose size is 80. The best time results were obtained by MSS pairs. Besides that, all of them solved the biggest solved instance. Only CLS pairs could not solve this instance.

In Table D.12 we can see the results obtained by some selected pairs with the biggest solved instance. The worst pair for this instance lasts approximately 3 times more than

the best ones. And its proof size is approximately 3 times biggest than the best pair size. In Table D.13 we show the results obtained with the biggest instance solved by all pairs (ST_3, whose size is 39). It is easy to see that the differences in time (more than 10 times) and size (approximately 4 times) between the best and worst pairs are not so big.

Pair	Time spent	Proof Size	Comments
<MSS,rev>	1087	13676	best in time
<MSS,inc>	1156	14661	second best in time
<CS,dec>	1454	9216	best in size
<CS,ins>	1503	12712	second best in size
<CS,rev>	3933	30690	worst in time and size

Table D.12: ST_4 results table.

Pair	Time spent	Proof Size	Comments
<BSS,T>	17	407	best in time
<BSS,F>	17	409	best in time
<BSS,ins>	17	410	best in time
<CS,dec>	22	283	best in size
<CLS,nfo>	188	1110	worst in time
<CLS,rev>	113	1123	worst in size

Table D.13: ST_3 results table.

D.2.7 Backjumping Family Results

As expected, the Backjumping Simple Strategy achieved the best results with B_PHP family instances. The biggest instance solved within the time limit was $B\_PHP_6^3$, whose size is 465; no strategy could solve $B\_PHP_7^3$ within the time limit. $B\_PHP_6^3$ was solved by 25 pairs. Of these, in 10 pairs the strategy was the Backjumping Simple Strategy. The other fifteen pairs had Simple Strategy, Learning Strategy and Memory Saver Strategy (5 each) as strategy. $B\_PHP_4^3$ was the biggest instance that was solved by all pairs.

In Table D.14 we can see the results obtained by some selected pairs with the biggest solved instance ($B\_PHP_6^3$). And in Table D.15 we present the results obtained by some selected pairs with the biggest instance solved by all pairs ($B\_PHP_4^3$, whose size is 165).

It is interesting to compare these results: the worst pair that solved B_PHP$^3_4$ took more time than the best pair for B_PHP$^3_6$ and produced a bigger proof.

Pair	Time spent	Proof Size	Comments
<BSS,or>	11006	21291	best in time and size
<BSS,T>	11116	21296	second best in time and size
<BSS,rfo>	11705	21296	third best in time and size
<BSS,dec>	21425	39483	fourth best in time and size
<BSS,bi-impli>	105920	110306	worst in time
<LS,nfo>	99591	186361	worst in size

Table D.14: B_PHP$^3_6$ results table.

Pair	Time spent	Proof Size	Comments
<BSS,T>	91	743	best in time
<BSS,or>	93	738	best in size
<CS,and>	3615	9702	an intermediate result
<CS,inc>	11870	26683	worst in time and size

Table D.15: B_PHP$^3_4$ results table.

It is also interesting to compare B_PHP with PHP results: the results obtained by Backjumping Simple Strategy pairs with B_PHP instances were only slightly worst than those obtained with PHP instances. But the results obtained by other strategies were much worst (for example, the time spent by <MSS,T> with B_PHP was more than four times the time spent by the same pair with PHP).

D.2.8 Random SAT Family Results

CPL strategies were able to give an answer in the time limit only for random K-SAT (cnf) instances 1 to 3 (see Table D.1). The results are exhibited in Table D.16. These results are much worse than those obtained by state-of-the-art SAT solvers.

Instance	Pair	Time spent	Proof size	Comments
cnf_1	<MSS,inc>	444	3778	best in time
cnf_1	<CS,inc>	969	3698	best in size
cnf_1	<CS,rfo>	1684	4654	worst in size and time
cnf_2	<MSS,nfo>	8252	19438	best in size and time
cnf_2	<CS,rfo>	53208	29428	worst in time and size
cnf_3	<MSS,rev>	8540	19243	best in time
cnf_3	<CS,nfo>	18308	19155	best in size
cnf_3	<CS,rfo>	73356	31204	worst in time
cnf_3	<CS,bi-impli>	60204	31988	worst in size

Table D.16: Random K-SAT results table.

D.2.9 First family results

mbC

The biggest first family (see Section D.1.2) instance solved within the time limit was Φ^1_{90}, whose size is 993. And this instance was solved by all pairs. In Table D.17 we can see the results obtained by some selected pairs with the biggest solved instance. The worst pair in time for this instance lasts approximately 1.4 times more than the best one. And the worst pair in size (which is the best in time) has almost the same size of the best pairs.

Pair	Time spent	Proof Size	Comments
<MBCES,dec>	66955	58681	best in time and worst in size
<MBCSS,ins>	67258	58591	second best in time
<MBCSS,inc>	78871	58416	one of the best in size
<MBCES,nfo>	97387	58416	worst in time

Table D.17: **mbC** Φ^1_{90} results table.

mCi

The biggest instance solved by **mCi** pairs with first family (see Section D.1.2) instances within the time limit was Φ^1_{90}. And this instance was solved by all pairs. In Table D.18 we can see the results obtained by some selected pairs with the biggest solved instance.

The worst pair in time for this instance lasts approximately 1.4 times than the best one. And the worst pair in size (which is the best in size) has almost the same size of the best pairs.

Pair	Time spent	Proof Size	Comments
`<MCIES,dec>`	67042	58681	best in time and worst in size
`<MCISS,ins>`	67145	58591	second best in time
`<MCISS,and>`	67199	58591	third best in time
`<MCIES,nfo>`	96930	58416	worst in time and one of the best in size

Table D.18: **mCi** Φ^1_{90} results table.

CPL

We also posed instances of the first family to **CPL** pairs. The The biggest instance solved within the time limit was Φ^1_{120} (size 1083 for **CPL**[8]). And Φ^1_{90} (size 813 for **CPL**) was the biggest instance solved within the time limit by all **CPL** pairs. Φ^1_{120} was solved by `<MSS,bi-impli>` (the best pair for this instance) in 42008 milliseconds and with a proof size of 51550. The best pair for Φ^1_{90} was `<MSS,rfo>`, which solved this instance in 12980 milliseconds and produced a proof with a size of 29209. These results make it very clear that this family is much easier for **CPL** than for **mbC** and **mCi**. The same observation applies to all other **LFI** families submitted to **CPL** pairs.

D.2.10 Second family results

mbC

The biggest second family instance solved by **mbC** pairs within the time limit was Φ^2_{14}, whose size is 472. And the biggest instance solved by all pairs was Φ^2_{10} (whose size is 278). In Table D.19 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 D.20 we show some results obtained with the biggest instance solved by all pairs (Φ^2_{10}). The difference in time between the best and worst pairs is more

[8]If we define $\circ X \stackrel{\text{def}}{=} \top$ and $\bullet X \stackrel{\text{def}}{=} \bot$, as we have shown in Section D.1.2, **CPL** versions of **LFI** problems have a smaller size.

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 D.19: **mbC** Φ^2_{14} 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 D.20: **mbC** Φ^2_{10} results table.

mCi

The biggest second family instance solved by **mCi** pairs within the time limit was Φ^2_{17}, whose size is 649. And the biggest instance solved by all pairs was Φ^2_{11} (whose size is 322). In Table D.21 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 D.22 we show some results obtained with the biggest instance solved by all pairs (Φ^2_{11}). The difference in time between the best and worst pairs is more than 19 times. And the difference in size between the best and worst pairs is more than 22 times.

Pair	Time spent	Proof Size	Comments
<MCISS,rev>	112586	181333	best in time
<MCIES,rev>	116751	181333	second best in time

Table D.21: **mCi** Φ^2_{17} results table.

Pair	Time spent	Proof Size	Comments
<MCISS,rev>	4376	15785	best in time and size
<MCIES,rev>	4550	15785	second best in time
<MCISS,inc>	27681	52226	third best in time
<MCIES,and>	86858	285642	worst in time
<MCISS,imp>	65654	360515	worst in size
<MCIES,imp>	66834	360515	worst in size

Table D.22: **mCi** Φ^2_{11} results table.

CPL

We also posed instances of the second family to **CPL** pairs. The biggest instance solved within the time limit was Φ^2_{20} (whose size is 623), and it was solved by all pairs. It was solved by <MSS,T> (the best pair for this instance) in 2123 milliseconds and with a proof size of 5784.

D.2.11 Third family results

mbC

The biggest instance solved by some **mbC** pairs with third family instances within the time limit was Φ^3_{14}, whose size is 509. And the biggest instance solved by all pairs was Φ^3_{11} (size 353). In Table D.23 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 D.24 we show some results obtained with the biggest instance solved by all pairs. The difference in time between the best and worst pairs is more than 10 times. And the difference in size between the best and worst pairs is more than 8 times.

Pair	Time spent	Proof Size	Comments
<MBCSS,rev>	96055	163470	best in time and size
<MBCES,rev>	102098	163470	second best in time

Table D.23: **mbC** Φ^3_{14} results table.

Pair	Time spent	Proof Size	Comments
<MBCSS,rev>	7358	21915	best in time and size
<MBCES,rev>	7798	21915	second best in time
<MBCSS,dec>	10715	27942	third best in time
<MBCES,ins>	75160	115467	worst in time
<MBCSS,imp>	49971	192386	worst in size
<MBCES,imp>	52308	192386	worst in size

Table D.24: **mbC** Φ^3_{11} results table.

mCi

The biggest instance solved by **mCi** pairs with third family instances. within the time limit was Φ^3_{12} (size 402). And the biggest instance solved by all pairs was Φ^3_{10} (size 307).

In Table D.25 we can see the results obtained by some pairs that solved the biggest solved instance. The interesting fact is that the two best pairs use the same sorter: Reverse Insertion Order. The difference in time between the best and worst pairs is approximately two times. And the difference in size between the best and worst pairs is less than 1.5 times.

In Table D.26 we show some results obtained with the biggest instance solved by all pairs. The difference in time between the best and worst pairs is more than 7 times. And the difference in size between the best and worst pairs is approximately 8 times.

Pair	Time spent	Proof Size	Comments
<MCISS,rev>	16925	42590	best in time and in size
<MCIES,rev>	18063	42590	second best in time
<MCISS,or>	30769	54903	worst in size
<MCIES,or>	32555	54903	worst in time and in size

Table D.25: **mCi** Φ^3_{14} results table.

CPL

We also posed instances of the second family to **CPL** pairs. The biggest instance solved within the time limit was Φ^3_{20} (size 672), and it was solved by all pairs. It was solved by <MSS,F> (the best pair for this instance) in 2513 milliseconds and with a proof size of 6311.

Pair	Time spent	Proof Size	Comments
<MCISS,rev>	3103	11786	best in time and size
<MCIES,rev>	3338	11786	second best in time
<MCIES,T>	28029	52540	worst in time
<MCISS,imp>	19634	87029	worst in size
<MCIES,imp>	20557	87029	worst in size

Table D.26: **mCi** Φ_{10}^{3} results table.

D.2.12 Fourth family results

mbC

The biggest instance solved by **mbC** pairs with fourth family instances within the time limit was Φ_{90}^{4} (size 1079). And the biggest instance solved by all pairs was Φ_{80}^{4} (size 959).

In Table D.27 we can see the results obtained by some pairs that solved the biggest solved instance. In Table D.28 we show some results obtained with the biggest instance solved by all pairs. In both cases, the best pairs' results are only slightly better than the worst pair results. The interesting fact is that the two best pairs use the same sorter: Reverse String Order.

Pair	Time spent	Proof Size	Comments
<MBCES,rfo>	55417	40142	best in time
<MBCSS,rfo>	55486	40140	second best in time and best in size
<MBCSS,dec>	74809	51202	worst in time and in size

Table D.27: **mbC** Φ_{90}^{4} results table.

Pair	Time spent	Proof Size	Comments
<MBCSS,rfo>	36033	31865	best in time and in size
<MBCES,rfo>	36315	31869	second best in time
<MBCSS,imp>	57975	40480	worst in time
<MBCSS,dec>	48496	40712	worst in size

Table D.28: **mbC** Φ_{80}^{4} results table.

mCi

The biggest instance solved by **mCi** pairs with fourth family instances within the time limit was Φ^4_{90}. And the biggest instance solved by all pairs was Φ^4_{80}.

In Table D.29 we can see the results obtained by some pairs that solved the biggest solved instance. The interesting fact is that the two best use the same sorter: Reverse String Order. In Table D.30 we show some results obtained with the biggest instance solved by all pairs. In both cases, the best pairs' results are only slightly better than the worst pair results. The interesting fact is that the two best pairs use the same sorter: Reverse String Order.

Pair	Time spent	Proof Size	Comments
`<MCISS,rfo>`	55026	40140	best in time and size
`<MCIES,rfo>`	55542	40142	second best in time
`<MCISS,dec>`	75296	51202	worst in time and size

Table D.29: **mCi** Φ^4_{90} results table.

Pair	Time spent	Proof Size	Comments
`<MCISS,rfo>`	35624	31865	best in time and size
`<MCIES,rfo>`	36177	31869	second best in time
`<MCISS,imp>`	58403	40480	worst in time
`<MCISS,dec>`	48500	40712	worst in size

Table D.30: **mCi** Φ^4_{80} results table.

CPL

We also posed instances of the fourth family to **CPL** pairs. The biggest instance solved within the time limit was Φ^4_{100} (size 1000). And Φ^4_{90} was the biggest instance solved within the time limit by all **CPL** pairs. Φ^4_{100} was solved by `<MSS,F>` (the best pair for this instance) in 39236 milliseconds and with a proof size of 41006. The best pair for Φ^4_{90} was `<MSS,rfo>`, which solved this instance in 26162 milliseconds and produced a proof with a size of 33306.

D.2.13 Seventh family results

mbC

The biggest instance solved by **mbC** pairs with seventh family instances within the time limit was Φ^7_{20} (size 336). And the biggest instance solved by all pairs was Φ^7_7 (size 115).

In Table D.31 we can see the results obtained by some pairs that solved the biggest solved instance. The best pair in time is more than 45 times faster than the worst pair. And the proof produced by the worst pair is more than 35 times bigger than the proof produced by the pair with the smallest proof.

In Table D.32 we show some results obtained with the biggest instance solved by all pairs. Here the best pair in time is more than 1200 times faster than the worst pair. And the proof produced by the worst pair is more than 480 times bigger than the proof produced by the pair with the smallest proof.

Pair	Time spent	Proof Size	Comments
<MBCES,F>	845	3524	best in time
<MBCES,and>	847	3524	second best in time
<MBCSS,nfo>	875	3504	best in size
<MBCSS,F>	951	3504	best in size
<MBCSS,and>	957	3504	best in size
<MBCES,rev>	40319	105872	worst in time and size

Table D.31: **mbC** Φ^7_{20} results table.

Pair	Time spent	Proof Size	Comments
<MBCES,F>	30	586	best in time
<MBCES,nfo>	32	586	second best in time
<MBCES,and>	32	586	second best in time
<MBCSS,and>	33	579	best in size
<MBCSS,nfo>	34	579	best in size
<MBCSS,F>	35	579	best in size
<MBCSS,rfo>	38006	279070	worst in time and size

Table D.32: **mbC** Φ^7_7 results table.

mCi

The biggest instance solved by **mCi** pairs with seventh family instances within the time limit was Φ^7_{20}. And the biggest instance solved by all pairs was Φ^7_8 (size 132).

In Table D.33 we can see the results obtained by some pairs that solved the biggest solved instance. The best pair in time is more than 45 times faster than the worst pair. And the proof produced by the worst pair is more than 30 times bigger than the proof produced by the pair with the smallest proof.

In Table D.34 we show some results obtained with the biggest instance solved by all pairs. Here the best pair in time is more than 2400 times faster than the worst pair. And the proof produced by the worst pair is more than 870 times bigger than the proof produced by the pair with the smallest proof.

Pair	Time spent	Proof Size	Comments
<MCIES,and>	849	3524	best in time
<MCISS,nfo>	883	3504	best in size
<MCISS,F>	963	3504	best in size
<MCISS,and>	964	3504	best in size
<MCIES,rev>	40445	105872	worst in time and size

Table D.33: **mCi** Φ^7_{20} results table.

Pair	Time spent	Proof Size	Comments
<MCIES,F>	45	728	best in time
<MCISS,rfo>	7209	720	best in size
<MCISS,nfo>	49	720	best in size
<MCISS,F>	51	720	best in size
<MCISS,and>	51	720	best in size
<MCISS,rfo>	110192	629303	worst in time and size

Table D.34: **mCi** Φ^7_8 results table.

CPL

We also posed instances of the second family to **CPL** pairs. The biggest instance solved within the time limit was Φ^7_{25} (size 272), and it was solved by all pairs. It was

solved by <MSS,or> (the best pair for this instance) in 317 milliseconds and with a proof size of 3246.

D.2.14 Eighth family results

mbC

As all eighth family instances are not valid for **mbC**, the biggest instance we submitted for **mbC** pairs was Φ^8_{10} (size 186). This was found to be not valid by all pairs. All proofs took approximately the same time and have almost the same size. The best result for the biggest instance was obtained by the <MBCSS,dec> pair which took 148 milliseconds and produced a refutation of size 1257.

mCi

The biggest instance solved by **mCi** pairs with eighth family instances within the time limit was Φ^8_{50} (size 946). And the biggest instance solved by all pairs was Φ^8_7 (size 129).

In Table D.35 we can see the results obtained by some pairs that solved the biggest solved instance. The time and size of best and worst pairs are almost the same.

In Table D.36 we show some results obtained with the biggest instance solved by all pairs. Here the best pair in time is more than 2000 times faster than the worst pair. And the proof produced by the worst pair is more than 530 times bigger than the proof produced by the pair with the smallest proof.

Pair	Time spent	Proof Size	Comments
<MCISS,or>	18430	25407	best in time and in size
<MCISS,dec>	18442	25407	second best in time
<MCIES,or>	18635	25507	worst in size
<MCIES,dec>	18969	25507	worst in time and size

Table D.35: **mCi** Φ^8_{50} results table.

Pair	Time spent	Proof Size	Comments
<MCISS,or>	39	682	best in time and in size
<MCISS,dec>	40	682	second best in time and best in size
<MCIES,dec>	41	696	third best in time
<MCIES,and>	78662	365630	worst in time and one of the worst in size

Table D.36: **mCi** Φ_7^8 results table.

CPL

We also posed instances of the eighth family to **CPL** pairs. The biggest instance we submitted was Φ_{10}^8 (size 166), and it was solved by all pairs. It was solved by <MSS,rev> (the best pair for this instance) in 17 milliseconds and with a proof size of 1006.

D.2.15 Ninth family results

mbC

As all ninth family instances are not valid for **mbC**, the biggest instance we submitted for **mbC** pairs was Φ_{25}^9 (size 397). This was found to be not valid by all pairs. All proofs took approximately the same time and have almost the same size. The best result in time for the biggest instance was obtained by the <MBCSS,nfo> pair which took 1807 milliseconds and produced a refutation of size 6219. And the best result in size for the biggest instance was obtained by the <MBCES,or> pair which took 1897 milliseconds and produced a refutation of size 5740.

mCi

The biggest instance solved by **mCi** pairs with ninth family instances within the time limit was Φ_{75}^9 (size 1197). And the biggest instance solved by all pairs was Φ_{40}^9 (size 637).

In Table D.37 we can see the results obtained by some pairs that solved the biggest solved instance. The size of best and worst pairs are almost the same, but the time taken to finish the worst pair was approximately 1.6 higher than the time taken by the best pair.

In Table D.38 we show some results obtained with the biggest instance solved by all

pairs. Here the best pair in time is more than 9 times faster than the worst pair. And the proof produced by the worst pair is approximately 1.8 times bigger than the proof produced by the pair with the smallest proof.

Pair	Time spent	Proof Size	Comments
<MCIES,inc>	59644	47691	best in time and size
<MCIES,imp>	59818	47691	second best in time
<MCISS,rev>	100869	48142	worst in time and size

Table D.37: **mCi** Φ^9_{75} results table.

Pair	Time spent	Proof Size	Comments
<MCIES,imp>	7547	14231	best in time and size
<MCIES,inc>	7549	14231	second best in time
<MCIES,rev>	68886	25534	worst in time and size

Table D.38: **mCi** Φ^9_{40} results table.

CPL

We also posed instances of the second family to **CPL** pairs. The biggest instance we submitted was Φ^9_{25} (size 272), and it was solved by all pairs. It was solved by <MSS,T> (the best pair for this instance) in 209 milliseconds and with a proof size of 3246.

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.