Complementarity of natural deduction and resolution principle in empirically automated theorem proving

Dominique Pastre
Crip5 - Université René Descartes - Paris

March, 2006

Abstract : MUSCADET is a knowledge-based theorem prover based on natural deduction. The results obtained during the CASC competitions of theorem provers show its complementarity with regard to resolution-based provers. This paper presents some MUSCADET proofs of theorems proposed at the last competition (2005) and points out some of its characteristics which may account for its successes.

Keywords : automated theorem proving, natural deduction, knowledge-based system

1 Introduction

In 1971, Bledsoe published a paper where he compared some proofs obtained either by the resolution principle [Robinson1965] or by natural deduction [Bledsoe 1971]. He showed that the proofs obtained by natural deduction[1] were shorter than those obtained by resolution.

Since then, several theorem provers based on resolution, were improved by many strategies, became powerful and very few provers were based on natural deduction.

In 2005, we have observed that at least one theorem could be proved by MUSCADET [Pastre1989, Pastre1993, Pastre2001b] a theorem prover based on natural deduction, but could not be proved by any of the participating resolution-based theorem provers, given the time limit imposed by the CASC competitions [Sutcliffe2005].

This does not mean that MUSCADET is better than other provers. Several theorems were proved by all entrants' systems in the last competition (2005) except by MUSCADET. This only tends to show that MUSCADET may be complementary to other provers. MUSCADET still needs to be improved. The improvements will be obtained by some new good heuristics and know-how, not by the increase of computer speed. We also see, looking at the results of the competition, that MUSCADET is faster than other provers on the problems it was able to solve. In case of a success, the proof is obtained at least as quickly as with the others, and much more quickly in many cases. If it fails it often quickly stops itself. The cases of "timeout" are generally due to infinite creations of objects, or due to too many sub-theorems.

For a long time, many theoretical results regarding resolution were published but the new strategies were not very satisfying from an practical perspective. Moreover, theorems were generally given to the provers as sets of clauses instead of first-order formulas. There may be one reason for this : although translating sets of first-order formulas into sets of clauses is easily automatizable, writing a *good* set of clauses was not so easy. There may be several possibilities, some of them being better for the resolution based provers. So it was better to do the transformation by hand.

In the TPTP Library [Sutcliffe, Suttner1998], created in 1993, all problems were expressed as sets of clauses up to 1997. Nowadays, more than two thirds of the problems are still given as sets of clauses. In the CASC competitions [Sutcliffe2005], there are five divisions. For four of them,

[1] For Bledsoe, the meaning of the terms "natural deduction" differs from what it is in Gentzen's rules; it is both larger and less formal.

comprising of nine categories, problems are given as sets of clauses. In only one division, divided into two categories, problems are given as sets of first order formulas. The consequence is that resolution-based provers can compete in all divisions, or may be specialized in one or another division, but provers that do not work with clauses can compete only in the FOF division (First Order Formula), which is the most general division of them all.

Moreover, the library grows on the contributions of researchers. As more researchers work with resolution-based provers, more new problems are more adapted to resolution-based provers than to natural deduction ones.

2 Main characteristics of Muscadet

2.1 Facts, rules and metarules

Muscadet is a knowledge-based system. It works with facts to which it applies rules.

Facts are the conclusion to be proved, the hypotheses, the objects of a theorem or a sub-theorem to be proved, links to concepts which appeared in the initial conjecture or in the definitions of the preceding concepts, sub-theorems, definitions, axioms and lemmas, and all sorts of facts which give relevant information during the proof searching process. At the beginning, there is no hypothesis and the conclusion to be proved is the first-order formula of the initial conjecture.

Rules are written in the form

rule <name> : if <list of conditions> then <list of actions>

By applying rules, hypotheses are added, objects may be created, the conclusion may be replaced by another one, the theorem to be proved may be split into one or more sub-theorems to be proved, independant or not.

Some rules are general and express logical or mathematical knowledge and usual mathematical know-how. Here are some examples of such rules.

rule $\forall$: if the conclusion is $\forall X\, P(X)$
then create a new object $X1$ and the new conclusion is $P(X1)$

rule $\rightarrow$: if the conclusion is $H \rightarrow C$
then add the hypothesis H and the new conclusion is C

rule `stop1` : if the conclusion is one of the hypotheses
then the new conclusion is `true`

rule $\wedge$: if the conclusion is a conjunction
then successively prove all elements of the conjunction

rule `defconcl` : if the predicate of the conclusion has a definition
then replace the conclusion by its definition

Other rules are automatically built by metarules from the definitions, lemmas and universal hypotheses. For example, from the definition of inclusion

$\forall A\ \forall B(A \subset B \leftrightarrow\ \forall X(X \in A \rightarrow X \in B)$

the following rule is built :

rule $\subset$: if $A \subset B$ and $X \in A$ are hypotheses
then add the hypothesis $X \in B$ if it is not yet a hypothesis.

From the definition[2] of intersection

$\forall A\ \forall B\ \forall X(X \in A \cap B \leftrightarrow X \in A \wedge X \in B)$

the following rules are built :

[2] Muscadet also accepts the following notation $A \cap B = \{X \mid X \in A \wedge X \in B\}$ but it cannot be used in the TPTP context.

rule $\cap 1$: if $A \cap B : C$ and $X \in C$ are hypotheses
then add the hypothesis $X \in A$ if it is not yet a hypothesis.
rule $\cap 2$: if $A \cap B : C$ and $X \in C$ are hypotheses
then add the hypothesis $X \in B$ if it is not yet a hypothesis.
rule $\cap 3$: if $A \cap B : C$, $X \in A$ and $X \in B$ are hypotheses
then add the hypothesis $X \in C$ if it is not yet a hypothesis.

where $A \cap B : C$ expresses that C is the intersection of A and B. This means that $A \cap B$ has already been introduced.

Some actions are elementary, such as replacing the conclusion by its definition by the rule `defconcl` above. A conclusion of the form
$A \subset B$
will simply be replaced by
$\forall X(X \in A \rightarrow X \in B)$

Some other actions are more sophisticated and are defined by packs of rules, such as adding a hypothesis which is defined by the following rules :
To add a hypothesis H :
if H is already a hypothesis or is of the form $X = X$, then do nothing
if H is a conjunction,
then successively add all the elements of the conjunction
if H is $\forall X P(X)$ then create local rules for this theorem
... [some others examples will be given in next sections]
in all other cases, add H as a new hypothesis.

Note that hypotheses which are added are only elementary hypotheses, disjunctive hypotheses and existential hypotheses. Conjunctive hypotheses are split before being added and universal hypotheses are treated as definitions or lemmas and replaced by rules. Disjunctive and existential hypotheses are first stored as hypotheses without being subjected to any particular treatment because their treatment may be useless or even expansive and it will be done only later if necessary.

2.2 Elimination of functional symbols

Strategies of Muscadet are designed to work with predicates rather than with functional symbols. In a formula with functional symbols, it "eliminates" them by giving names to the terms. These objects will replace these terms in the predicative formula. So, there remains no hypothesis or conclusion such as $p(f(a))$ but instead the hypothesis or conclusion $p(b)$ where b is a constant defined by the hypothesis $f(a) : b$.

The symbol ":" is used to express that b is the object $f(a)$, and the formula $f(a) : b$ will be handled as if it were a predicative formula.

These transformations cannot be done directly on the initial statement of the theorem to be proved, since some of the functional expressions will become hypotheses while others will become conclusions. Moreover, those expressions may contain variables which complicate things and they also appear in some definitions. So the expressions are first transformed by using a new quantifier noted "!", which means
"for the only ... equal to ...".
$p(f(a))$ is replaced by $!A{:}f(a), p(A)$ which means
"for the only A equal to $f(a)$ then $p(A)$ holds"
where A is a variable.

This mechanism is recursive. In the example given in the next section, the formula
$\text{inv}(f, a \cap b) = \text{inv}(f, a) \cap \text{inv}(f, b)$
is replaced by

$$!A{:}a \cap b, !B{:}\text{inv}(f, A), !C{:}\text{inv}(f, a), !D{:}\text{inv}(f, b), !E : C \cap D, B = E \quad .$$

This work is done by the (recursive) rules of the action `elifun` which is called for the first conclusion to be proved (and also for the definitions and lemmas) by the rule `elifun`.

Then the expressions !A:<term>, <property> are handled by rules specific to hypotheses or conclusion, or to building rules from definitions.

Here are examples of such rules :

rule ! : if the conclusion is of the form $!Y{:}F(...), P(Y)$
  then if there is already a hypothesis $F(...){:}Y1$
    then the new conclusion is $P(Y1)$
    else create a new object
      add the hypothesis $F(...)$:<this new object>
      and the new conclusion is P(<this new object>).

To add a hypothesis H :
  if H is of the form $!Y{:}F(...), P(Y)$
  then if there is already a hypothesis $F(...){:}Y1$
    then add the hypothesis $P(Y1)$
    else create a new object
      and add the hypotheses $F(...)$:<this newobject>
        and P(<this new object>)

2.3 Equality handling

Because of elimination of functional symbols, equality may occur in hypotheses only as equality of objects. Each time two objects of a (sub)theorem to be proved are found to be equal, one of them is replaced everywhere in the sub-theorem by the other, then it is removed, and so is the equality.

In the example given in the next section, during the proof searching process of the second sub-theorem, the object t is in $f^{-1}(b1)$ and in $f^{-1}(b2)$, so t has an image u in $b1$ and an image v in $b2$.

The following hypotheses have been added :
  $f(t){:}u$, $f(t){:}v$,
  $u \in b1$ hence $u \in b$ since $b1 \subset b$,
  $v \in b2$ hence $v \in b$ since $b2 \subset b$.

The unicity of the image of t in b implies that $u = v$.

Then v is replaced by u and we have the new hypotheses $u \in b2$.

Now the same object u belongs to $b1$ and to $b2$, hence to $b1 \cap b2$ and it is possible to conclude that t belongs to $f^{-1}(b1 \cap b2)$.

2.4 Negation handling

MUSCADET works with positive properties as much as possible. Rules built from definition are made to work with positive properties rather than negative properties. MUSCADET usually does not add negative properties, on the contrary there are general rules to eliminate negations in most cases.

If the conclusion is a negation $\neg A$, then a new hypothesis A is added and the new conclusion is `false`. This is one of the few cases of proofs by contradiction, the (sub-)theorem will be proved if a contradiction is found (hypothesis `false` added) or if a new conclusion appears, for example after removing a negative hypothesis due to the following rule :
  if a hypothesis is a negation $\neg A$
  and if the conclusion to be proved is `false`
  then the hypothesis is removed and the new conclusion is A.

In other cases, negative hypotheses may be rewritten, for example
  $\neg\neg A := A$

$\neg(A \rightarrow B) := A \wedge \neg B$ which gives two hypotheses A and $\neg B$
$\neg(A \leftrightarrow B) := (A \wedge \neg B) \vee (\neg A \wedge B)$
There are also some rare built rules which test negative hypothesis (see section 3.2).

There are two instances where MUSCADET is required to handle negations,
- firstly if the statement of the theorem to be proved contains itself negations, for example $(p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$ (SYN046+1),
- secondly if the handled concepts concern negation, for example complements or differences of sets, empty or disjoint sets.

The example given in section 3.2 concerns the difference of sets, the definition of which is
$\forall E\ \forall A(X \in (E - A) \leftrightarrow X \in E \wedge \neg(X \in A))$
We will see the four rules which are built from these definitions. Three of them do not contain any negation, this was obtained by moving formulas in conditions, actions, hypotheses and conclusion parts. The fourth rule contains only a negative hypothesis in the part condition.

2.5 Examples

Some detailed examples of simple proofs of easy theorems may be found in [Pastre1993, Pastre2001a, Pastre2001b]. Proofs of other theorems may be found in [Pastre1993, Pastre2001c, Pastre2002]. The following section gives detailed proofs of more difficult theorems and provides comments.

3 Examples of MUSCADET proofs

3.1 Handling images and inverse images : a property of inverse images

Here is an example of a theorem which was proved by MUSCADET during the last competition (2005), but by none of the other provers.

Theorem SET757+4. *If f maps A into B and X and Y are two subsets of B, then the inverse image by f of the intersection of X and Y is equal to the intersection of the inverse images by f of X and of Y.*

that is, in usual mathematical notation $f^{-1}(X \cap Y) = f^{-1}(X) \cap f^{-1}(Y)$

Its formal statement in first order predicate calculus is
$\forall F\ \forall A\ \forall B\ \forall X\ \forall Y$
$(\text{maps}(F, A, B) \wedge X \subset B \wedge Y \subset B$
$\rightarrow \text{inv}(F, X \cap Y, A) =_{\text{set}} \text{inv}(F, X, A) \cap \text{inv}(F, Y, A))$
with the following definitions [3]
$\forall F\ \forall A\ \forall B(\texttt{maps}(F, A, B) \leftrightarrow$
$(\forall X(X \in A \rightarrow \exists Y(Y \in B \wedge \texttt{apply}(F, X, Y)))$
$\wedge\ \forall X\ \forall Y1\ \forall Y2(X \in A \wedge Y1 \in B \wedge Y2 \in B \rightarrow$
$(\texttt{apply}(F, X, Y1) \wedge \texttt{apply}(F, X, Y2) \rightarrow Y1 = Y2))))$
(every element in the domain A of F has an image in the range B and this image is unique).
$\forall A\ \forall B(A =_{\text{set}} B \leftrightarrow A \subset B \wedge B \subset A)$
$\forall F\ \forall A\ \forall B\ \forall X(X \in \text{inv}(F, B, A) \leftrightarrow X \in A\ \wedge\ \exists Y(Y \in B\ \wedge\ \text{apply}(F, X, Y))$

From these definitions the following rules was built :
rule $=_{\text{set0}}$: if $A =_{\text{set}} B$ is a hypothesis
then add the hypothesis $A \subset B$ [4]

[3]the definitions of $\subset$ and $\cap$ have already been given in section 2.1
[4]for all rules, add "if it is not yet a hypothesis"

rule $=_{\text{set1}}$: if $A =_{\text{set}} B$ is a hypothesis
then add the hypothesis $B \subset A$
rule maps1 : if map(F, A, B), $X \in A$ are hypotheses
then add the hypothesis $\exists Y(Y \in B \wedge$ apply$(F, X, Y))$
rule maps2 : if maps(F, A, B), $X \in A$, $Y1 \in B$, $Y2 \in B$,
apply$(F, X, Y1)$ and apply$(F, X, Y2)$ are hypotheses
then add the hypothesis $Y1 = Y2$
rule inv1 : if inv(F, B, A):C and $X \in C$ are hypotheses
then add the hypothesis $X \in A$
rule inv2 : if inv(F, B, A):C and $X \in C$ are hypotheses
then add the hypothesis $\exists Y(Y \in B \wedge$ apply$(FX, Y))$
rule inv3 : if inv(F, B, A):C, $Y \in B$ and apply(F, X, Y) are hypotheses
then add the hypothesis $X \in C$

Here is the MUSCADET proof [5]. The initial theorem to be proved in numbered 0. Its conclusion is the given conjecture :
$\forall F\ \forall A\ \forall B\ \forall X\ \forall Y$
(maps$(F, A, B) \wedge X \subset B \wedge Y \subset B$
$\rightarrow$ inv$(F, X \cap Y, A) =_{\text{set}}$ inv$(F, X, A) \cap$ inv$(F, Y, A))$
five applications of the rule $\forall$ remove the universal quantifier and create correspondant objects [6] f, a, b, $b1$, $b2$
and the new conclusion is
maps$(f, a, b)\ \wedge\ b1 \subset b\ \wedge\ b2 \subset b) \rightarrow$
inv$(f, b1 \cap b2, a) =_{\text{set}}$ inv$(f, b1, a) \cap$ inv$(f, b2, a)$
.... rule $\forall$
elimination of functional symbols
new conclusion
maps$(f, a, b) \wedge b1 \subset b \wedge b2 \subset b) \rightarrow$
$!A$:$b1 \cap b2$, $!B$:inv(f, A, a),
$!C$:inv$(f, b1, a)$, $!D$:inv$(f, b2, a)$, $!E$:$C \cap D$,
$B =_{\text{set}} E$
.... rule elifun
separation of hypotheses and conclusion
add hypotheses maps(f, a, b), $b1 \subset b$ and $b2 \subset b$
new conclusion
$!A$:$b1 \cap b2$, $!B$:inv(f, A, a),
$!C$:inv$(f, b1, a)$, $!D$:inv$(f, b2, a)$, $!E$:$C \cap D$,
$B =_{\text{set}} E$
.... rule $\rightarrow$
three applications of the rule ! create objects and their definition
add objects : $b3$, $a0$, $a1$, $a2$ and $a3$
add hypotheses $b1 \cap b2 : b3$, inv$(f, b3, a) : a0$,
inv$(f, b1, a)$:$a1$, inv$(f, b2, a) : a2$, $a1 \cap a2 : a3$
new conclusion $a0 =_{\text{set}} a3$
.... rule !
definition of the conclusion
new conclusion $a0 \subset a3 \wedge a3 \subset a0$
.... rule defconcl
as the conclusion is now a conjunction, the theorem 0 is split into two sub-theorems 1 and 2 which will be proved one after the other

[5]Some useless actions have been removed.
[6]the system names the objects `o, o1, o2, o3`, and so on, but I give here the names f, a, b, $b1$ and so on, which make the proof more easily readable by a human reader.

. . . . sub-theorem 1
the conclusion of the first sub-theorem is the first sub-formula of the conjonction ; hypotheses and other facts are copied from theorem 0 to sub-theorem 1
new conclusion $a0 \subset a3$ . . . . creation of sub-theorem 1
definition of the conclusion
new conclusion $\forall A(A \in a0 \rightarrow A \in a3)$ . . . . rule defconcl
add object x
new conclusion $x \in a0 \rightarrow x \in a3$ . . . . rule $\forall$
add hypothesis $x \in a0$
new conclusion $x \in a3$ . . . . rule $\rightarrow$
add hypothesis $x \in a$ . . . . rule inv1
add hypothesis $\exists A(A \in b3 \wedge \text{apply}(f, x, A))$
x has an image in $b1 \cap b2$ because it belongs to $f^{-1}(b1 \cap b2)$
. . . . rule inv2
treatment of the existential hypothesis : creation of an object y in $b1 \cap b2$ which is the image of x
add object y
add hypotheses $y \in b3$ and $\text{apply}(f, x, y)$
add hypothesis-treated $\exists A(A \in b3 \wedge \text{apply}(f, x, A))$
this fact is used to memorize the fact that the hypothesis has been treated (it cannot be removed because the rule which has added it would add it again, and infinitely loop)
. . . . rule $\exists$
add hypothesis $y \in b1$ (*since* $y \in b1 \cap b2$) . . . . rule $\cap 1$
add hypothesis $y \in b$ (*since* $b1 \subset b$) . . . . rule $\subset$
add hypothesis $y \in b2$ (*since* $y \in b1 \cap b2$) . . . . rule $\cap 2$
add hypothesis $x \in a1$
(*x belongs to $f^{-1}(b1)$ since its image y belongs to $b1$*)
. . . . rule inv3
add hypothesis $x \in a2$
(*x belongs to $f^{-1}(b2)$ since its image y belongs to $b2$*)
. . . . rule inv3
add hypothesis $x \in a3$
(*x belongs to $f^{-1}(b1) \cap f^{-1}(b2)$ since its belongs to $f^{-1}(b1)$ and to $f^{-1}(b2)$*)
. . . . rule $\cap 3$
the last hypothesis added is the actual conclusion to be proved, then the sub-theorem 1 is proved and its conclusion is put at `true` *to memorize the fact that it is proved*
new conclusion (of theorem 1) `true` . . . . rule stop1
theorem 1 proved

since sub-theorem 1 is proved, the first sub-formula of the conclusion of theorem 0 is removed
new conclusion (of theorem 0) $a3 \subset a0$
and the second sub-theorem is now being proved

. . . . sub-theorem 2
new conclusion $a3 \subset a0$
. . . . creation of sub-theorem 2
definition of the conclusion
new conclusion $\forall A(A \in a3 \rightarrow A \in a0)$ . . . . rule defconcl
add object t
new conclusion $t \in a3 \rightarrow t \in a0$ . . . . rule $\forall$
add hypothesis $t \in a3$
new conclusion $t \in a0$ . . . . rule $\rightarrow$
add hypothesis $t \in a1$
(*since t belongs to $f^{-1}(b1) \cap f^{-1}(b2)$ it belongs to $f^{-1}(b1)$*)

. . . . rule $\cap 1$
add hypothesis $t \in a2$ ($and\ to\ f^{-1}(b2)$)
. . . . rule $\cap 2$
add hypothesis $t \in a$ (*and to the domaine* a)
. . . . rule inv3
add hypothesis $\exists A(A \in b1 \wedge \text{apply}(f,t,A))$
(t *has an image in* $b1$ *since it belongs to* $f^{-1}(b1)$)
. . . . rule inv2
add hypothesis $\exists A(A \in b2 \wedge \text{apply}(f,t,A))$
(*and an image in* $b2$ *since it belongs to* $f^{-1}(b2)$)
. . . . rule inv2
treatment of the first existential hypothesis of the sub-theorem :
creation of u *image of* t *in* $b1$
add object u
add hypotheses $u \in b1$ and $\text{apply}(f,t,u)$
add hypothesis-treated $\exists A(A \in b1 \wedge \text{apply}(f,t,A))$
. . . . rule $\exists$
add hypothesis $u \in b$
(u *belongs to* b *since* $b1 \subset b$) . . . . rule $\subset$
treatment of the second existential hypothesis of the sub-theorem :
creation of v *image of* t *in* $b2$
add object v
add hypothesis $v \in b2$ and $\text{apply}(f,t,v)$
add hypothesis-treated $\exists A(A \in b2 \wedge \text{apply}(f,t,A))$
. . . . rule $\exists$
add hypothesis $v \in b$
v *belongs to* b *since* $b2 \subset b$ . . . . rule $\subset$
add hypothesis $u = v$ (*since the image is unique*) . . . . rule maps2
replace v by u propagate and remove v
add hypothesis $u \in b2$ . . . . rule =
add hypothesis $u \in b3$
(u *belongs to* $b1 \cap b2$ *since* u *belongs to* $b1$ *and to* $b2$)
. . . . rule $\cap 3$
add hypothesis $t \in a0$
(t *belongs to* $f^{-1}(b1 \cap b2)$ *since its image* u *belongs to* $b1 \cap b2$)
. . . . rule inv3
new conclusion (of theorem 2) `true` . . . . rule stop1
theorem 2 proved

since sub-theorem 2 is proved, there is no more formula to prove in the conclusion of theorem 0, so it is proved
new conclusion (of theorem 0) `true` . . . . rule concl$\wedge$
theorem 0 proved

3.2 Handling concepts with negation : a theorem about complements

Here is an example of a theorem which was proved by MUSCADET in less than 0.01 second during the last competition (2005), and also by Vampire [Riazanov,Voronkov2002] but in 99 seconds, and by none of the other provers.

Theorem SET012+4. *If* A *is a subset of* E *then the complement in* E *of its complement* *is equal to itself,*

Its formal statement in first order predicate calculus is
$\forall E\ \forall A(A \subset E \rightarrow (E - (E - A)) =_{\text{set}} A)$

The definition of complement in a set (or difference) is
$\forall E\ \forall A(X \in (E - A) \leftrightarrow X \in E \wedge \neg(X \in A)$

The rules built from this definition are the followings :
rule `diff1` : if $(E - A) : B$ and $X \in B$ are hypotheses
then add the hypothesis $X \in E$ [7]
rule `diff2` : if $(E - A) : B$, $X \in B$ and $X \in A$ are hypotheses
then add the hypothesis `false`
The hypothesis `false` means that there is a contradiction in the hypotheses, hence the sub-theorem to which such a rule may be applied is proved.
rule `diff3` : if $(E - A) : B$, $X \in E$ and $\neg(X \in A)$ are hypotheses
then add the hypothesis $X \in B$
rule `diff4` : if $(E - A) : B$ and $X \in E$ are hypotheses
then add the hypothesis $X \in A \vee X \in B$.

By manipulations analogous to those executed in the preceding example, MUSCADET creates objects e, a, b and c with their properties stored as hypotheses :
$a \subset e$
$(e - a) : b$
$(e - b) : c$
and the conclusion to be proved is
$c =_{\text{set}} a$
which is replaced by its definition
$c \subset a \wedge a \subset c$.

Two sub-theorems are to be proved.

PROOF OF SUB-THEOREM 1
The conclusion
$c \subset a$
is replaced by its definition
$\forall X(X \in c \rightarrow X \in a)$
an object x is created such that
$x \in c$
and the conclusion is
$x \in a$.
The rules `diff1` and `diff4` add the hypotheses
$x \in e$ and $x \in a \vee x \in b$.
After splitting of this disjunctive hypothesis, two sub-theorems have to be proved

SUB-THEOREM 11 with the hypothesis $x \in a$ which is the conclusion to be proved, and

SUB-THEOREM 12 with the hypothesis $x \in b$ and the rule `diff2` brings the contradiction

PROOF OF SUB-THEOREM 2
The conclusion

[7] for all rules, add "if it is not yet a hypothesis"

$a \subset c$
is replaced by its definition then an object y is created such that
$y \in a$
and the conclusion is
$y \in c$.
The rules `diff1` and `diff4` add the hypotheses
$y \in e$ and $y \in b \vee y \in c$.
then two sub-theorems 21 and 22 are proved as before.

3.3 Handling intersections and unions

The following theorem

Theorem SET601+3.p.
$\forall A\ \forall B\ \forall C\ ((A \cap B) \cup (B \cap C) \cup (C \cap A) =_{\text{set}} (A \cup B) \cap (B \cup C) \cap (C \cup A))$

was proved by MUSCADET in less than 0.01 second at the last competition. It was also proved by Prover9 [McCune2005] in less than 0.01 second. Three other provers also proved it, but in at least 146 seconds.

MUSCADET splits this theorem into five final sub-theorems. We can see that Prover9 also splits this theorem since, as the author wrote in its CASC system description, "a preprocessing step attempts to reduce the problem to independent subproblems".

Here, these splittings are particularly efficient.

The proof of MUSCADET looks like an elementary proof given by a unexperienced mathematician. More advanced mathematicians would give a more direct and smart proof by using his/her knowledge about distributivity of intersections and unions.

The rules which are applied by MUSCADET are those which were described in previous sections and rules built from the definition of union.

definition : $\forall A\ \forall B\ \forall X (X \in A \cup B \leftrightarrow X \in A \vee X \in B)$

built rules :
rule $\cup 1$: if $A \cup B : C$ and $X \in C$ are hypotheses
then add the hypothesis $X \in A \vee X \in B$ [8]
rule $\cup 2$: if $A \cup B : C$ and $X \in A$ are hypotheses
then add the hypothesis $X \in C$
rule $\cup 3$: if $A \cup B : C$ and $X \in B$ are hypotheses
then add the hypothesis $X \in C$

Note that the two last rules directly lead to elementary belongings whereas the first rule leads to disjunctive hypotheses which will lead to splittings.

The initial theorem is split into two sub-theorems by splitting the equality of sets into two inclusions.

SUB-THEOREM 1
to prove the conclusion
$(a \cap b) \cup (b \cap c) \cup (c \cap a) \subset (a \cup b) \cap (b \cup c) \cap (c \cup a)$
where a, b, c are objects and all the intersections and unions such as $a \cap b$ have been created as

[8] for all rules, add "if it is not yet a hypothesis"

new objects with their own names, an object x is taken in $(a \cap b) \cup (b \cap c) \cup (c \cap a)$ and it must be proved that x belongs to $(a \cup b) \cap (b \cup c) \cap (c \cup a)$

Belonging to the union leads, in two steps, to three sub-theorems which are easily proved.

- split of $x \in (a \cap b) \cup (b \cap c) \cup (c \cap a)$
- SUB-THEOREM 11 : $x \in (a \cap b) \cup (b \cap c)$
  - -split
  - -SUB-THEOREM 111 : x belongs to $a \cap b$ hence to a and to b hence to $a \cup b$, $c \cup a$, $b \cup c$ and to their intersection
  - -SUB-THEOREM 112 : x belongs to $b \cap c$ hence to b and to c hence to $(b \cup c)$, $(a \cup b)$, $c \cup a)$ and to their intersection
- SUB-THEOREM 12 : x belongs to $c \cap a$ hence to c and to a, hence to $c \cup a$, $b \cup c$, $a \cup b$ and to their intersection

SUB-THEOREM 2
to prove the conclusion
$(a \cup b) \cap (b \cup c) \cap (c \cup a) \subset (a \cap b) \cup (b \cap c) \cup (c \cap a)$
an object y is taken in $(a \cup b) \cap (b \cup c) \cap (c \cup a)$
y belongs to $a \cup b$, $b \cup c$ and $c \cup a$
belonging to these unions leads to four final sub-theorems

- split of $y \in a \cup b$
- SUB-THEOREM 21 : y belongs to a
  - split of $y \in b \cup c$
  - SUB-THEOREM 211 : y belongs to b hence to $a \cap b$ and to $(a \cap b) \cup (b \cap c) \cup (c \cap a)$
  - SUB-THEOREM 212 : y belongs to c hence to $c \cap a$ and to $(a \cap b) \cup (b \cap c) \cup (c \cap a)$
- SUB-THEOREM 22 : y belongs to b
  - $y \in b \cup c$ does not have to be split
  - split of $y \in c \cup a$
  - SUB-THEOREM 221 : y belongs to c hence to $(b \cap c)$ and to $(a \cap b) \cup (b \cap c) \cup (c \cap a)$
  - SUB-THEOREM 222 : y belongs to a hence to $(c \cap a)$ and to $(a \cap b) \cup (b \cap c) \cup (c \cap a)$

3.4 Handling singletons and pairs

Here is another example of a theorem which was proved by MUSCADET in less than 0.01 second during the last competition (2005), and also by Vampire [Riazanov,Voronkov2002] but in 240 seconds, and by none of the other provers.

Theorem SET703+4
The union of two singletons $\{A\}$ *and* $\{B\}$ *is equal to the unordered pair* $\{A, B\}$

Its formal statement is
$\forall A\ \forall B\ (\{A\} \cup \{B\} =_{\text{set}} \{A, B\})$

The definitions of singleton $\{A\}$ and unordered-pair $\{A, B\}$ are
$\forall A\ \forall X (X \in \{A\} \leftrightarrow X = A)$
$\forall A\ \forall B\ \forall X (X \in \{A, B\} \leftrightarrow X = A \vee X = B)$

The rules built from these definitions are
rule singleton1 : if $\{A\} : S$ and $X \in S$ are hypotheses
then add the hypothesis $X = A$

rule singleton2 : if $\{A\} : S$ is a hypothesis
then add the hypothesis $A \in S$

rule pair1 : if $\{A, B\} : P$ and $X \in P$ are hypotheses
then add the hypothesis $X = A \vee X = B$

rule pair2 : if $\{A, B\} : P$ is a hypothesis
then add the hypothesis $A \in P$

rule pair3 : if $\{A, B\} : P$ is a hypothesis
then add the hypothesis $B \in P$

Proof

The conclusion of the first theorem to be proved is the statement of the conjecture.

add objects a and b
new conclusion $\{a\} \cup \{b\} =_{\text{set}} \{a, b\}$ .... rule ∀
new conclusion
$!A{:}\{a\},\ !B{:}\{b\},\ !C{:}A \cup B,\ !D{:}\{a, b\},\ C =_{\text{set}} D$ .... rule elifun
add object $a1$, $b1$, $a1b1$ and ab
add hypotheses $\{a\} : a1$, $\{b\} : b1$, $a1 \cup b1 : a1b1$ and $\{a, b\} : ab$
new conclusion $a1b1 =_{\text{set}} ab$ .... rule !
add hypothesis $a \in a1$ .... rule singleton2
add hypothesis $a \in a1b1$ .... rule ∪2
add hypothesis $b \in b1$ .... rule singleton2
add hypothesis $b \in a1b1$ .... rule ∪3
add hypothesis $a \in ab$ .... rule pair2
add hypothesis $b \in ab$ .... rule pair3
definition of the conclusion
new conclusion $a1b1 \subset ab \wedge ab \subset a1b1$ .... rule defconcl

.... SUB-THEOREM 1
new conclusion $a1b1 \subset ab$
.... creation of sub-theorem 1
definition of the conclusion
new conclusion $\forall A(A \in a1b1 \rightarrow A \in ab)$ .... rule defconcl
add object x
new conclusion $x \in a1b1 \rightarrow x \in ab$ .... rule ∀
add hypothesis $x \in a1b1$
new conclusion $x \in ab$ .... rule →
add hypothesis $x \in a1 \vee x \in b1$ .... rule ∪1
treatment of the disjunctive hypothesis $x \in a1 \vee x \in b1$
new conclusion $(x \in a1 \rightarrow x \in ab) \wedge (x \in b1 \rightarrow x \in ab)$
add hypothesis-treated $x \in a1 \vee x \in b1$ .... rule ∨

.... SUB-THEOREM 11
new conclusion $x \in a1 \rightarrow x \in ab$
.... creation of sub-theorem 11
add hypothesis $x \in a1$
new conclusion $x \in ab$ .... rule →
add hypothesis $x = a$ .... rule singleton1
replace a by x propagate and remove a
add hypothesis $x \in ab$ .... rule =
new conclusion (of theorem 11) `true` .... rule stop1
theorem 11 proved

new conclusion (of theorem 1) $x \in b1 \rightarrow x \in ab$

. . . . SUB-THEOREM 12
new conclusion $x \in b1 \rightarrow x \in ab$
. . . . creation of sub-theorem 12
proved in an analogous manner
theorem 12 proved

new conclusion (of theorem 1) `true` . . . . rule concl_and
theorem 1 proved

new conclusion (of theorem 0) $ab \subset a1b1$

. . . . SUB-THEOREM 2
new conclusion $ab \subset a1b1$
. . . . creation of sub-theorem 2
definition of the conclusion
new conclusion
$\forall A(A \in ab \rightarrow A \in a1b1)$ . . . . rule defconcl
add object y
new conclusion $y \in ab \rightarrow y \in a1b1$ . . . . rule $\forall$
add hypothesis $y \in ab$
new conclusion $y \in a1b1$ . . . . rule $\rightarrow$
add hypothesis $y = a \vee y = b$ . . . . rule pair1
treatment of the disjunctive hypothesis $y = a \vee y = b$
new conclusion $(y = a \rightarrow y \in a1b1) \wedge (y = b \rightarrow y \in a1b1)$
add hypothesis-treated $y = a \vee y = b$ . . . . rule $\vee$

. . . . SUB-THEOREM 21
new conclusion $y = a \rightarrow y \in a1b1$
. . . . creation of sub-theorem 21
add hypothesis $y = a$
new conclusion $y \in a1b1$ . . . . rule $\rightarrow$
replace a by y propagate and remove a
add hypothesis $y \in a1b1$ . . . . rule =
new conclusion (of theorem 21) `true` . . . . rule stop1
theorem 21 proved

new conclusion (of theorem 2) $y = b \rightarrow y \in a1b1$

. . . . SUB-THEOREM 22
new conclusion $y = b \rightarrow y \in a1b1$
. . . . creation of sub-theorem 22
proved in an analogous manner
theorem 22 proved

new conclusion (of theorem 2) `true` . . . . rule concl_and
theorem 2 proved

new conclusion (of theorem 0) `true` . . . . rule concl_and
theorem 0 proved

4 Why MUSCADET may be an efficient system

First, I will point out a major difference between the resolution principle and the natural methods of MUSCADET.

With the resolution principle, there is a very theoretically efficient deduction rule which may generate so many clauses that the proof, even theoretically obtainable, may be not found in a reasonable time. Strategies are written to limit the exponential “combinatorial explosion”.

With the natural methods of MUSCADET, there is a high but reasonable number of rules, given or automatically built. The conditions for their application are strict and they usually lead to a linear growth. Some of them may be expansive (infinite creation of objects) or exponential (splittings) but they have low priority and if time is out, this often means that a proof could not be obtained. This may also mean that priorities are not right and prevent an efficient rule to be applied. Strategies must therefore be written to define further rules and metarules, and consequently the number of useful deduced facts.

The efficiency of some MUSCADET characteristics will now be commented.

4.1 Representations

The first order formula of the conjecture to be proved is decomposed in various facts.

There are first the hypotheses and the conclusion to be proved. The simpler of them look like clauses but there are differences.

- hypotheses and conclusion are closed formulas;
- a hypothesis P (or $\neg P$) is not handled as a conclusion (or a part of a disjunctive conclusion) $\neg P$ (or P);
- quantifiers are not systematically removed and stay as long as possible near the scope of the quantified variable;
- there are no universal hypotheses, instead of them there are new local rules
- a conclusion may be disjunctive, there are rules to handle them, for example
  - if A is found to be a new hypothesis,
    then the conclusion $A \vee B$ is true
  - if the conclusion is $\neg A \vee B$,
    then the new hypothesis A is added
    and the new conclusion is B

Many objects are created, they will be useful for example to simply verify a conclusion of the form $\exists X P(X)$. If there is not yet such an object a more complex mechanism must be used.

Objects correspond to Skolem constants, but there are no other Skolem functions. We will see in subsection 4.3 the mechanism which replaces them.

Other facts are the definitions and lemmas. Metarules build rules from their formal statements. They contain rewriting rules and build step by step the conditions and the actions of these new rules. The mechanism is describe in [Pastre1989]. We have seen several such rules in preceding sections. Formal statements of the lemmas are afterwards not useful. For the definitions, their statements, after elimination of functional symbols (see 2.2), are still useful (for the definition of the conclusion).

Hypotheses are never removed. When an existential or disjunctive hypothesis H is treated, a new fact is added to register that this hypothesis has been treated and will prevent its being added again.

4.2 Splittings

As [Bledsoe 1971] already pointed out, splitting is very efficient. We also saw in section 3.3 the efficiency of splitting since Prover9 [McCune2005] and MUSCADET proved a theorem in less than 0.01 second whereas others provers needed at least 146 seconds or failed.

In MUSCADET splitting is not only a preprocessing step but may be done at all levels of the proof.

Vampire [Riazanov,Voronkov2002] uses another sort of splitting rule which applies to clauses [Riazanov,Voronkov2001]. But this splitting needs to introduce new predicates, thus it has to be limited. The authors say "Although the use of splitting results in degradation of performance on the average, there exist many problems which VAMPIRE can solve in reasonable time only with splitting".

There are various versions of this splitting rule. This explains perhaps why Vampire 7.0 proved theorem SET711+4 [9] in less than 0.01 second, as well as MUSCADET, and Vampire 8.0 needed 170 seconds (no other prover succeded), whereas theorem SET601+3, which we have seen in section 3.3, is proved by Vampire 8.0 (146 seconds needed) but not by Vampire 7.0 (timeout).

4.3 treatment of functional symbols

We have already seen in sections 2.2 and 3.1 how a formula of the form $P(F(X))$ is treated when it appears in a conclusion, after it has been instanciated. But such a formula may appear as a sub-formula anywhere in a conjecture or in a definition or lemma and it may contain variables. There is no skolemisation.

$P(F(X))$
might be replaced by
$\forall Y(Y{=}F(X)) \rightarrow P(Y))$ or by $\exists Y(Y{=}F(X) \wedge P(Y))$
and one or the other of these formulas would be more adequate later when it is handled. It will depend on its position in relation to implications and negations. It is the reason why the quantifier "!" ("for the only ... equal to ...") was introduced.

$P(F(X))$
is replaced by
$!Y{:}F(X), P(Y)$
which means
"for the only Y equal to $F(X)$, $P(Y)$"

It is only when the sub-formula $!Y{:}F(X), P(Y)$ appears as a conclusion to be proved or as a hypothesis to be added or as the sub-formula to be processed during building rules that this sub-formula will be dealt with.

The rule ! : if the conclusion is of the form $!Y{:}F(...), P(Y)$
then if there is already a hypothesis $F(...){:}Y1$
then the new conclusion is $P(Y1)$
else create a new object
add the hypothesis $F(...)$:<this new object>
and the new conclusion is $P($<this new object>$)$
treats "!" in the same manner as $\forall$.

The rule
To add a hypothesis H :
if H is of the form $!Y{:}F(...), P(Y)$

[9] This theorem states the unicity of the inverse image of a one-to-one mapping.

then if there is already a hypothesis $F(...)$:$Y1$
then add the hypothesis $P(Y1)$
else create a new object
and add the hypotheses $F(...)$:<this new object>
and P(<this new object>)
treats "!" in the same manner as $\exists$ but immediately without waiting any longer.

Note that these treatments are done on closed formulas, the objects are Skolem constants, there is no need for other Skolem functions.

In metarules building rules from definitions and lemmas, "!" is also treated either as $\forall$ or as $\exists$ according to its position in the sub-formula which is treated.

We also saw in sections 2.2 and 3.1 that the mechanism is recursive.
A formula $f(g(a))$ leads to objects b and c and to hypotheses $g(a)$:b and $f(b)$:c.

The first advantage of this flattening is that all intermediary terms are created as objects and may be considered in a rule, as we have seen with the rules $\cap i$ and $\cup i$

The second advantage is that if $f(a)$ and $f(b)$ are found to be equal, this will be memorized by the fact that there will be now only one object c to design both terms, with the hypotheses $f(a)$:c and $f(b)$:c. $f(a)$ and $f(b)$ have now the same properties thanks to the intermediary of their same name c.

On the downside, if the succession of functional symbols is characteristic in a term $f(g(a))$ then it is not easily visible. It is the reason why MUSCADET is not adapted to work in group theory in particular.

4.4 Building efficient rules

The building of rules has already been mentionned several times. It is a complex mechanism which is described in details in [Pastre1989] involving many rules. I will here only develop a crucial point which is in relation with the treatment of functional symbols.

Firstly, rules are adapted to the choice of working with positive properties. If we have two sets such that $A \subset B$, the rule $\subset$ (see section 2.1) will be applied each time an element is found to be in A. These elementary properties will certainly be useful. If there are no sets such as $A \subset B$, the concept $\subset$ is probably not pertinent. Surely, it must not have priority.

This set of rules is not complete, but some failures are widely balanced by the successes.

Secondly, if a definition or a lemma leads to a statement of the form
$H \rightarrow p(f(X))$
(for example $X \in A \rightarrow X \in A \cup B$)
the following rule could be built
if H is a hypothesis then add the new hypothesis $p(f(X))$
but this would have introduced the object $f(X)$ which has perhaps nothing to do in the actual context. Moreover, this mechanism could be expansive by introducing $f(X)$, then $f(f(X))$, and so on (for example introducing unions of unions of unions, and so on, of sets),

Instead of this, the formula
$H \rightarrow p(f(X))$
which has been rewritten
$H \rightarrow !Y{:}f(X), p(Y)$
leads to the condition
$!Y{:}f(X)$ is a hypothesis
and to the action

add the hypothesis $p(Y)$
that is to the rule
if <conditions built from H >
and $!Y\!:\!f(X)$ is a hypothesis
then add the hypothesis $p(Y)$.

So the hypothesis $p(Y)$ is added only if $f(Y)$ already exists. And if $f(Y)$ is pertinent, it probably will be introduced by other rules. This sort of rules is efficient and, above all, it is not dangerous, so it could have a high priority. This explains why MUSCADET is efficient in proving theorems where many concepts F are defined by formulas of the form
$\forall X(P(F(X)) \leftrightarrow ...)$,
more especially in proving theorems about unions, intersections, mappings, images, power sets, inverse images, etc.

But there are situations where these restrictions are too severe while building rules. This is the case if the functional symbol F does not correspond to a defined concept but to a term which appears only in axioms (for example the functional symbol "growth_rate" in axioms of MGT problems). In these situations, the sub-formulas of the form
$H \rightarrow p(f(X))$
will lead to the building of other rules, in which there is not the condition
$!Y\!:\!f(X)$ is a hypothesis
but there is the action
add the hypothesis $\exists Y(f(X) : Y \wedge p(Y))$.
The hypothesis will be treated later as the other existential hypotheses.

4.5 Treatment of existential hypotheses

Another feature of MUSCADET explains its efficiency when handling images, inverse images, such as nonempty or disjoint sets.
From a sub-formula of the form
$H \rightarrow \exists X P(F(X)))$
where
$P(F(X))$
denotes a sub-formula involving the functional symbol F
(for example every element has an image or every element has an antecedant, (see section 3.1))
the following rule might be built :

if <conditions built from H >
then create a new object
and add the hypotheses $F(X)$:<this new object>
and $P($<this new object>

But this rule might be expansive. Instead the built rule is just

if <conditions built from H > then add the hypothesis $\exists X P(X))$

So existential hypotheses are first stored without being treated, and they will be treated later, one by one, in the order in which they have been added (for example an image, then an antecedant, then an image, etc, or if there are several mappings, images by each of them successively).

This explain the successes of MUSCADET in the proof of theorems

- SET751+4, proved only by MUSCADET in less than 0.01 second and by Vampire [Riazanov,Voronkov2002] in more than 100 seconds during the last competition (2005)
- HAL002+1, proved only by MUSCADET and E [Schulz2002] in less than 0.01 second and by Vampire in more than 50 seconds.

During the CASC-19 competition (2003), four theorems about mappings were proved only by MUSCADET and by no other prover. (SET723+4, SET743+4, SET750+4, SET752+4). These theorems may perhaps have been proved in 2005, but there is another theorem, SET741+4, which is also proved by MUSCADET and which is more difficult than those four theorems. Indeed, my own machine needs 7.5 seconds to prove it, whereas the four others are proved in less than 0.24 second.

In this theorem SET741+4, there are 3 sets, 9 mappings, injective or surjective, and MUSCADET creates 31 elements (16 necessary and 15 useless).

4.6 Equality handling

We saw in section 2.3 the treatment of equality, combined with the elimination of functional symbols. In hypotheses, equalities occur only between constants (objects) and are quickly removed, so as one of the objects, by replacing everywhere one of the constants by the other. Consequently, there are no longer equalities in hypotheses. The equality between two terms is only implicit : they are named by the same constant. This simplification is successful.

A consequence of this fact is that an equality in a hypothesis must not appear in the condition of a rule. Such a rule will not be applied, except during the short time between the addition of this hypothesis and its removal ! So, the treatment of equalities in the building of rules must be specific.

As we saw in section 3.4, the definitions of singleton and unordered-pair does not lead to the rules

if $\{A\}$:S and $X = A$ are hypotheses
then add the hypothesis $X \in S$

and

if $\{A, B\}$:P and X=A are hypotheses
then add the hypothesis $X \in P$

but to the rules

if $\{A\}$:S is a hypothesis
then add the hypothesis $A \in S$

and

if $\{A, B\}$:P is a hypothesis
then add the hypothesis $A \in P$

This is successful, not only for the theorem SET703+4, which is relatively easy, but also for theorem SET707+4

$\forall A\ \forall B(\{\{A\}, \{A, B\}\} = \{\{U\}, \{U, V\}\} \leftrightarrow A = U \wedge B = V)$

the difficulty of which was emphasized by [Brown1986] in his paper about his system based on the fundamental deduction principle.

This is successful also for all theorems relying on a concept which includes, among other things, the assertion of the uniqueness of objects, for example mapping, injection or partition.

5 Conclusion

MUSCADET functions in a manner which is quite different from resolution-based provers. It uses methods based on natural deduction and is a knowledge-based system. We have seen some of these methods which are crucial and explain why MUSCADET is able to prove some theorems that resolution-based provers are not yet able to prove. We have also seen that, in cases where theorems are also proved by other provers, MUSCADET proof can be obtained at least as fast as with the other provers and much faster in many cases.

However, MUSCADET cannot prove some theorems that resolution-based provers can easily prove.

MUSCADET is efficient for everyday mathematical problems which are expressed in a natural manner, for example in naive set theory. It is not efficient for problems which are defined axiomatically, from a logician's point of view, for instance in the fields of axiomatic geometry or axiomatic set theory.

MUSCADET is efficient to solve problems which involve many axioms, definitions or lemmas. It is not efficient at all to solve problems which involve only one large conjecture and no intermediary definitions.

Improvements of MUSCADET will not come from the increase of computer speed. Most of the time, either it succeeds or it fails and stops quickly. In MUSCADET the role of heuristics is not to limit the number of deduced facts which generally increases in a linear manner. Their role is to enlarge the scope of situations that it is able to handle efficiently. The analysis of the failures of MUSCADET will be pursued in order to help to refine its heuristics.

Nevertheless there are two cases of time out. One is due to too many useless splittings, the other concerns problems containing too many existential axioms leading to too many objects being created. Until now MUSCADET does not backtrack, as it is more difficult to decide when it should backtrack than to refine heuristics so that it chooses the right path. Two potential improvements would lead to better results : to implement backtracking and to improve heuristics to choose the right path, especially in the case of problems in axiomatic theories.

Another way to improve theorem proving is to have provers cooperate. [Sutcliffe2001] has already worked to have the best resolution-based provers cooperate, and the performance was better than the performance of each of the components.

It would certainly be possible to have MUSCADET cooperate with a resolution-based prover. Some provers may begin with a preprocessing step, such as attempting to split a problem into sub-problems before clausifying it [Schulz2002]. MUSCADET could first analyse the given problem and choose between searching itself for a proof or calling for the resolution-based prover. In the event of failure on a sub-problem or after having exceeded a time limit, it could also call for the resolution-based prover for this sub-problem.

Acknowledgements

I would like to thank Geoff Sutcliffe and Christian Suttner for their interest in empirically successful automated reasoning, for having created the TPTP Problem Library and for updating it, and finally for organizing the CASC competitions.

References

[Bledsoe 1971] W. W. Bledsoe, Splitting and Reduction Heuristics in Automatic Theorem Proving, *Journal of Artificial Intelligence*, 2:55–77, 1971.

[Bledsoe1977] W. W. Bledsoe, Non-Resolution Theorem Proving, *Journal of Artificial Intelligence*, 9:1–35, 1977.

[Brown1986] F.M.Brown, An Experimental Logic based on the Fundamental Deduction Principle, *Journal of Artificial Intelligence*, 30:117–263, 1986

[McCune2005] W. McCune, http://www.mcs.anl.gov/∼mccune/prover9, 2005

[Pastre1989] D. Pastre, Muscadet: An Automatic Theorem Proving System using Knowledge and Metaknowledge in Mathematics, *Journal of Artificial Intelligence*, 38(3):257–318, 1989.

[Pastre1993] D. Pastre, Automated Theorem Proving in Mathematics, *Annals on Artificial Intelligence and Mathematics*, 8(3-4):425–447, 1993

[Pastre2001a] D. Pastre, Muscadet version 2.3 : User's Manual, *http://www.math-info.univ-paris5.fr/∼pastre/muscadet/manuel-en.ps*, 16p, 2001a

[Pastre2001b] D. Pastre, Muscadet2.3 : A Knowledge-based Theorem Prover based on Natural Deduction, *International Joint Conference on Automated Reasoning - Conference on Automated Deduction*, 685–689, 2001b, http://www.math-info.univ-paris5.fr/∼pastre/muscadet

[Pastre2001c] D. Pastre, Implementation of Knowledge Bases for Natural Deduction, *2nd International Workshop on Implementation of Logics, 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, 49–68, 2001c

[Pastre2002] D. Pastre, Strong and weak points of the MUSCADET theorem prover, *AI Communications*, 15(2-3):147-160, 2002, http://www.cs.miami.edu/∼tptp

[Riazanov,Voronkov2001] A. Riazanov, A. Voronkov A. Splitting without backtracking, *17th Inernational Joint Conference on Artificial Intelligence, IJCAI 01*, 611-617, 2001

[Riazanov,Voronkov2002] A. Riazanov, A. Voronkov A. The Design and Implementation of Vampire, *AI Communications*, 15(2-3):91-110, 2002

[Robinson1965] J.A. Robinson, A machine oriented logic based on the resolution principle, *J.ACM* 12:23-41, 1965

[Schulz2002] S. Schulz. E: A Brainiac Theorem Prover, *AI Communications*, 15(2-3):111-126,2002

[Sutcliffe, Suttner1998] G. Sutcliffe, C.B. Suttner, The TPTP Problem Library: CNF Release v1.2.1, *Journal of Automated Reasoning* 21(2):177–203.1998

[Sutcliffe2001] G. Sutcliffe, The Design and Implementation of a Compositional Competition-Cooperation Parallel ATP System, *2nd International Workshop on Implementation of Logics, 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, 92–102,2001

[Sutcliffe2005] G. Sutcliffe, The IJCAR-2004 Automated Theorem Proving Competition, *AI Communications*, 18(1):33–40.2005, http://www.cs.miami.edu/∼tptp/CASC