AI Communications 15 (2002) 147–160
IOS Press

Strong and weak points of the MUSCADET theorem prover – examples from CASC-JC

Dominique Pastre
Crip5, Université René Descartes – Paris 5, 45 rue des Saints Pères, F-75270 Paris Cedex, France
E-mail: pastre@math-info.univ-paris5.fr

Abstract. MUSCADET is a knowledge-based theorem prover based on natural deduction. It has participated in CADE Automated theorem proving System Competitions. The results show its complementarity with regard to resolution-based provers. This paper presents some of its crucial methods and gives some examples of MUSCADET proofs from the last competition (CASC-JC in IJCAR 2001).

Keywords: Theorem proving, knowledge bases, natural deduction

1. Introduction

Most theorem provers nowadays are based on the resolution principle [16]. The properties of such provers have been studied from a theoretical point of view and much progress has also been made from a practical point of view. However, it is also useful to continue to improve theorem provers based on natural deduction, following the terminology of Bledsoe [2–4]. MUSCADET [12–14] is such a natural deduction type system. As for the UT interactive theorem prover implemented by Bledsoe [5], MUSCADET does not contain a standard set of Gentzen natural deduction rules but implements natural-deduction-like techniques and uses heuristics. Bledsoe also calls such systems 'natural' systems or goal oriented systems [4]. Moreover, MUSCADET is built as a knowledge-based system; theorems (and sub-theorems) to be proved are decomposed and represented as sets of facts and all methods are expressed as rules which are either given to the system or automatically built by metarules.

MUSCADET has participated in the last three CADE Automated theorem proving System Competitions (CASC-16/17/JC). It was the only prover based on natural deduction and the results show its complementarity with regard to resolution-based provers. For example, at CASC-JC, MUSCADET was the only prover which was able to prove five problems out of the fifty in the FEQ category (First order with EQuality) and it was also the only prover which was unable to prove two other problems. The FEQ and FNE categories constitute the FOF division (First Order Formula) which is the only division in which MUSCADET could compete since, in all other divisions (except SEM in CASC-17 only), the problems are not stated by first order formulas but by clauses.

Section 2 quickly presents the main characteristics of MUSCADET and especially its main strategy which is the use of rules automatically built from axioms. Section 3 illustrates this strategy with a theorem from CASC-JC which was proved only by MUSCADET and one other entrant. Sections 4 and 5 present two of the crucial strategies, the processing of existential properties and the processing of negation, and illustrate them by two theorems which were proved only by MUSCADET. The construction of rules is explained in Section 6. Section 7 provides a commentary on forward and backward chaining, and Section 8 exposes problems for which MUSCADET is not adapted or has to be improved. Section 9 mentions some other provers which work with knowledge bases, and Section 10 concludes on the complementarity of MUSCADET with regard to resolution-based provers.

2. Main characteristics of MUSCADET

MUSCADET is composed of rules and metarules. It works with facts and (mathematical) objects to which it applies rules. Consequently, the main structure of MUSCADET is not an algorithm previously studied from a theoretical point of view. If it contains some algorithms, they are only small algorithms which have been written for technical and minor reasons.

Some of the rules are automatically built by metarules.

Examples of facts are

'$p(a)$ is a hypothesis of Theorem 1',
'a is a mathematical object which has been introduced in Theorem 1',
'$q(b)$ is the conclusion of Theorem 2.1 which has to be proved',
'Theorem 2.1 is a sub-theorem of Theorem 2',
'$\exists X p(X)$' is a hypothesis of Theorem 2.2',
'the existential hypothesis $\exists X p(X)$ of Theorem 2.2 has been treated' (*i.e., a mathematical object which verifies p already existed or had been created*),
'concept p is pertinent for theorem 0',
'active rules (*i.e., rules which are pertinent for the* (*sub-*)*theorem to be proved*) *are $r1, r2, \ldots$'.*

Facts result from the initial theorem to be proved or are added by rules.

Rules are either logical rules or rules which have been built from definitions, lemmas or universal hypotheses.

Examples of logical rules are

'If the conclusion to be proved is $H \Rightarrow C$, then add the new hypothesis H and the new conclusion is C',
'If the conclusion to be proved is a conjunction, then successively prove all elements of this conjunction'.

Several other rules are given in Fig. 1. Some of the automatically built rules are given in Fig. 2.

Rules contain a list of conditions which are easily verified, for example

'$p(X)$ is a hypothesis',
'$\exists X p(X)$ is a hypothesis',
'there is no hypothesis of the form $p(X)$',

and a list of actions which may be elementary, for example

'set the new conclusion to C'
'replace all occurrences of X by Y in expression E',

or more complicated actions which are called super-actions and are defined by packs of rules, for example

'To add a hypothesis H,
if H is already a hypothesis, 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
...

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 $\vee$: if $A \vee B$ is a hypothesis
and the conclusion is C
then the new conclusion is
$(A \Rightarrow C) \wedge (B \Rightarrow C)$
rule *stop*: 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 *exists*: if $\exists X P(X)$ is the first
existential hypothesis
which has not been treated
and there is no object satisfying P
then create an object X satisfying P
rule *concl_exist*: if the conclusion is $\exists P(X)$
then search for an object X satisfying P

Elimination of functional symbols:
rule *elifun*: for each subformula $P(\ldots, F(\ldots), \ldots)$
of the conclusion where the term
$F(\ldots)$ is totally instantiated,
then create a new object Y,
add the hypothesis $F(\ldots) : Y$
and the new conclusion is $P(\ldots, Y, \ldots)$

Definition of the conclusion:
rule *defconcl*1: if the conclusion is $C(A1, \ldots, An)$
and there is a definition of the form
$C(X1, \ldots, Xn) \leftrightarrow D(X1, \ldots, Xn)$
then the new conclusion is $D(A1, \ldots, An)$
rule *defconcl*2: if the conclusion is $R(A, B)$,
there is a hypothesis of the form
$F(A1, \ldots, An) : B$
and there is a definition of the form
$R(X, F(X1, \ldots, Xn) \leftrightarrow D(X, X1, \ldots, Xn)$
then the new conclusion is $D(A, A1, \ldots, An)$

Fig. 1. Some universal and logical rules.

in all other cases, add H as a new hypothesis
(this concerns elementary hypotheses and existential hypotheses)'

There is no backtracking. Adding useless facts such as useless hypotheses does not matter because their number grows only in a linear manner. Some actions may lead to failure or to the creation of infinitely many objects or of a very large number of sub-theorems. Priorities and heuristics are used but there is no backtracking because it would be more difficult to decide when the system should backtrack than to search for heuristics to avoid these problems. Mostly, in the case of

$\forall A \forall B(A \subset B \Leftrightarrow \forall X(X \in A \Rightarrow X \in B))$ $\forall A \forall B \forall X(X \in A \cup B \Leftrightarrow X \in A \vee X \in B)$ $\forall A \forall X(X \in \mathcal{P}(A) \Leftrightarrow X \subset A)$
rule $\subset$: if $A \subset B$ and $X \in A$ are hypotheses, and $X \in B$ is not a hypothesis then add the hypothesis $X \in B$ rule $\cup$1: if $A \cup B : C$ and $X \in C$ are hypotheses, and $X \in A \vee X \in B$ is not a hypothesis then add the hypothesis $X \in A \vee X \in B$ rule $\cup$21: if $A \cup B : C$ and $X \in A$ are hypotheses, and $X \in C$ is not a hypothesis then add the hypothesis $X \in C$ rule $\cup$22: if $A \cup B : C$ and $X \in B$ are hypotheses, and $X \in C$ is not a hypothesis then add the hypothesis $X \in C$ rule $\mathcal{P}$: if $P(A) : B$ and $X \in B$ are hypotheses, and $X \subset A$ is not a hypothesis then add the hypothesis $X \subset A$

Fig. 2. Definitions occurring in problem SET694+4 and rules built from these definitions.

too many objects or too many splittings, MUSCADET would not be able anyway to prove the theorem.

Metarules are used to build rules from definitions. We will see such rules in Sections 3 and 4 and examples of constructions in Section 6.

Metarules are also used to activate rules, i.e., to select rules which concern the concepts that occur in the theorem to be proved, and to order rules.

At the beginning of the proof, there is only one fact: the conclusion of theorem numbered 0 is the statement of the initial theorem to be proved.

A theorem is proved if its conclusion becomes *true*. If it is only a sub-theorem, the fact that it has been proved is returned to the theorem which created it.

A conclusion is set to *true* for example, if the conclusion to be proved is also a hypothesis, or if the conclusion is an existential property which is verified by the objects and their properties.

The system gives a trace of all the successful applications of rules, i.e., the objects which are introduced, the hypotheses which are added, the new conclusions which replace the old ones, the splitting of theorems into two or more sub-theorems, the subtheorems which are proved, etc. As all rules express natural properties, the traces are easily comprehensible to the human reader. They may however be a little tedious on occasion because all the details are given and repeated even if they look like preceding steps.

Examples of such traces are given in [15].

3. Applying rules

We have seen that MUSCADET works with objects, hypotheses, a conclusion and rules. It has to prove the conclusion by applying rules which may, for example, add new hypotheses, modify the conclusion, create new objects, split the theorem into two or more subtheorems.

To illustrate this mechanism, in this section, I will give the proof of TPTP problem SET694+4 and comment on it.

The union of the power set of two sets A and B is included in the power set of the union of A and B

I chose this theorem as the first example because it was one of the CASC-JC problems and, although it is rather easy, it was proved only by MUSCADET and by one other entrant. I was surprised that all resolution-based provers (except this one) ran over the time. The reasons why MUSCADET is successful is that the rules perform actions which are restricted to actions which would be carried out by a mathematician. This is the case for logical rules and also for rules which are built from definitions. Here, these rules quickly lead to the proof. These actions often resemble actions which are carried out by resolution, but only an effective subset of these actions is considered.

Notice that the first example given by Bledsoe in [2] to show the efficiency of his natural-deduction-like prover also concerned power sets. (This example involved intersection which is easier that union because it leads to less splitting.)

MUSCADET receives the first order statement of this conjecture

$$\forall A \forall B(\mathcal{P}(A) \cup \mathcal{P}(B) \subset \mathcal{P}(A \cup B)).$$

It first builds rules from the definitions in the SET006+0.ax file which is included in the problem statement. Figure 2 shows the definitions which are used in this problem and the rules which have been automatically built from these definitions. I will show in section 6 how these rules are built. Once rules are built, they may be stored in order to be used for another problem without being built again. This possibility is not used in the TPTP context.

Then MUSCADET considers the conjecture and activates the rules that are pertinent for the conjecture. This means that, later, it does not have to try to apply many rules that have nothing to do with the context of the conjecture. To decide which rules are pertinent, MUSCADET first links the conjecture to the concepts (predicate and functional symbols other than logical

rules	objects	hypotheses	conclusion
			$\forall A \forall B(A \cup B \subset \mathcal{P}(A \cup B))$
$\forall$	a, b		$\mathcal{P}(a) \cup \mathcal{P}(b) \subset \mathcal{P}(a \cup b)$
elifun	pa, pb pab c, pc	$\mathcal{P}(a) : pa, \mathcal{P}(b) : pb$ $pa \cup pb : pab$ $a \cup b : c, \mathcal{P}(c) : pc$	$pab \subset pc$
*defconcl*1			$\forall X(X \in pab \Rightarrow X \in pc)$
$\forall$	d		$d \in pab \Rightarrow d \in pc$
$\Rightarrow$		$d \in pab$	$d \in pc$
$\cup 1$		$d \in pa \vee d \in pb$	
$\vee$			$(d \in pa \Rightarrow d \in pc) \wedge (d \in pb \Rightarrow d \in pc)$
$\wedge$			split into two sub-theorems
sub-theorem 1			
			$d \in pa \Rightarrow d \in pc$
$\Rightarrow$		$d \in pa$	$d \in pc$
$\mathcal{P}$		$d \subset a$	
*defconcl*1			$d \subset c$
*defconcl*1			$\forall X(X \in d \Rightarrow X \in c)$
$\forall$	x	$x \in d$	$x \in c$
$\subset$		$x \in a$	
$\cup 21$		$x \in c$	
stop			*true*
			sub-theorem 1 proved
sub-theorem 2			
			$d \in pb \Rightarrow d \in pc$
is proved	in a	analogous manner	sub-theorem 2 proved initial theorem proved

Fig. 3. MUSCADET proof of theorem SET694+4.

symbols) that occur in the conjecture, and recursively to the concepts which occur in the definitions of the preceding concepts. Pertinent rules are universal rules and rules which have been built from the definitions of the concepts linked to the conjecture. Here the rules in Fig. 2 are active because they involve $inclusion$, $union$ and *power set*.

At the beginning, there is no hypothesis and no object, and the first conclusion to be proved is the first order statement of the conjecture

$$\forall A \forall B(\mathcal{P}(A) \cup \mathcal{P}(B) \subset \mathcal{P}(A \cup B)).$$

Figure 3 presents an outline of the proof. One can see in the first line the first conclusion to be proved, which is the given conjecture. Then each line shows the name of the rule which is applied, the objects which are introduced, the hypotheses which are added and the new conclusion.

Prolog conventions are used for mathematical variables and constants. Variables start with upper-case letters whereas constants start with lower-case letters.

Here follows a commentary on this proof.

Rule $\forall$ is a classic rule in natural deduction. It is applied twice. It instantiates the variables $X1$ by a new constant a (then by b), it creates objects a and b and the new conclusion is

$$\mathcal{P}(a) \cup \mathcal{P}(b) \Leftrightarrow \mathcal{P}(a \cup b).$$

MUSCADET accepts statements with functional symbols but works as if there were predicate symbols. Rule *elifun* calls a super-action which creates objects pa, pb, pab, c and pc,[1] adds the hypotheses

$$\mathcal{P}(a) : pa,\ \mathcal{P}(b) : pb,\ pa \cup pb : pab,\ a \cup b : c$$

[1]The naming conventions have been simplified to make the paper more readable.

and

$$\mathcal{P}(c) : pc$$

and the conclusion is

$$pab \subset pc$$

The formula $a \cup b : c$, for example, will be manipulated as if it was a predicate $p(a, b, c)$. The symbol ':' means that c is the name of the union of a and b.

Then rule *defconcl*1 replaces the conclusion by its definition

$$\forall X(X \in pab \Rightarrow X \in pc).$$

In MUSCADET there are no axioms or hypotheses – following the terminology of TPTP – but definitions and lemmas. The building of rules is slightly different for definitions and for lemmas. The replacement of the conclusion only occurs for definitions. In a standard use of MUSCADET, the user defines properties as definitions or lemmas. In the TPTP context, MUSCADET has to recognize definitions among the axioms. Roughly, definitions are formulas containing an equivalence in which one argument is of the form $P(X1, \ldots, Xn)$ or $Q(X1, \ldots, F(X1, \ldots, Xn), \ldots, Xn)$ where P or F has no other occurrence in the formula.

Rule $\forall$ is applied again and creates the object d in pab and the new conclusion is

$$d \in pab \Rightarrow d \in pc.$$

Rule $\Rightarrow$ is also a classic rule in natural deduction. It decomposes the statement into a hypothesis $d \in pab$ and a conclusion $d \in pc$.

Then rule $\cup 1$ adds the hypothesis

$$d \in pab \vee d \in pb$$

and rule $\vee$ replaces the conclusion by

$$(d \in pa \Rightarrow d \in pc) \wedge (d \in pb \Rightarrow d \in pc).$$

Rule $\vee$ has a low priority. It is applied here because no other rule can be applied. Otherwise, rules with higher priority would be applied first in order to avoid useless splittings.

Then rule $\wedge$ splits the theorem into two sub-theorems. The conclusion of the first sub-theorem is

$$(d \in pa \Rightarrow d \in pc).$$

Rule $\Rightarrow$ is applied again and adds the hypothesis $d \in pa$, and the new conclusion is $d \in pc$.

Rule $\mathcal{P}$ adds the hypothesis $d \subset a$.

Rule *defconcl*1 is applied twice and replaces the conclusion by

$$d \subset c$$

then by

$$\forall X(X \in d \Rightarrow X \in c).$$

Then rules $\forall$ and $\Rightarrow$ are applied again as previously described. A new object x is created in d and the conclusion is $x \in c$.

Then rule $\subset$ adds the hypothesis $x \in a$, and rule $\cup 21$ adds $x \in c$ which is the conclusion to be proved. So, rule $stop$ sets the conclusion to $true$ and ends the proof by stating that the sub-theorem is proved. This information is returned to the initial theorem and rule $\wedge$ continues the proof with the second sub-theorem which is proved in an analogous manner.

As both sub-theorems are proved, the initial theorem is proved.

4. Processing of existential properties

In this section I will comment on MUSCADET proofs of some problems that have been proved only by MUSCADET in CASC-JC.

While commenting on the proofs I will explain some crucial MUSCADET strategies that enabled it to prove theorems that other CASC entrants failed to prove.

In particular, the creation of objects verifying existential hypotheses may be expansive.

The reasons for the efficiency of MUSCADET are the following.

First, as before, the rules which are automatically built only execute actions which would have been carried out by humans.

Secondly, the processing of existential hypotheses is delayed until no other rule with a higher priority may be applied.

Thirdly, after the creation of the object(s) verifying an existential hypothesis, all the other rules are tried again before the next existential hypothesis is treated. So all elementary properties of the objects which have just been introduced will be used before new objects are created.

Fourthly, existential hypotheses are treated in the order in which they appear. This allows, if f maps A

onto itself, images and pre-images to be created alternatively instead of creating only images for example and missing the right element if it is a pre-image. Moreover, if there are several sets and many mappings, it allows all sets and all mappings to be considered.

Lastly, another reason for the efficiency of MUSCADET in this type of problem is that MUSCADET does not skolemize the formulas. It does not handle the term $A \cup B$ where the functional symbol $\cup$ corresponds to a concept in the same manner as an object which is created because of the existential property. When it is treated this object is a constant because all other variables have necessarily been instantiated before. If the formulas had been skolemized, this object would have been written as a term with a Skolem function and the difference of nature would have disappeared.

4.1. Proof and comments on problem SET722+4

The English statement of problem SET722+4 is

Consider two mappings $f : A \to B$ and $g : B \to C$. If $g_o f$ is surjective then g is surjective.

Its first order statement is

$$\forall F \forall G \forall A \forall B \forall C (maps(F, A, B) \wedge maps(G, B, C) \wedge surjective(compose(G, F, A, B, C), A, C) \Rightarrow surjective(G, B, C))$$

and its MUSCADET proof is given in Fig 6.

MUSCADET first builds rules from the axioms of the files SET006+0.ax and SET006+1.ax which are included in the problem statement. Figure 4 shows some definitions about mappings. Figure 5 shows some of the rules which were automatically built from these definitions.

Then MUSCADET activates rules that are pertinent for the conjecture, i.e., universal rules, and also rules *maps*1, *maps*2, *surjective*, *compose*1 and *compose*2 which involve *mappings*, *surjective mappings* and *composition of mappings.*

The first conclusion to be proved is the first order statement of the conjecture. At the beginning, there is no hypothesis and no object.

Notice that the first order formulas are not skolemized. They are not in prenex form either; on the contrary, quantifiers remain the most interior possible. Only the most exterior and universal quantifier is removed. For the axioms, all free variables are then implicitly universal.

For the conjecture, rule $\forall$ creates objects, i.e., constants. This corresponds to the skolemization of the most exterior and existential quantifier of the negation of the conjecture. In the example, the objects a, b, c, f, g are added and the conclusion is

$$maps(f, a, b) \wedge maps(g, b, c) \wedge surjective(compose(g, f, a, b, c), a, c) \Rightarrow surjective(g, b, c)).$$

Rule *elifun* creates an object h, adds the hypothesis $compose(g, f, a, b, c) : h$, and the conclusion is

$$maps(f, a, b) \wedge maps(g, b, c) \wedge surjective(h, a, c) \Rightarrow surjective(g, b, c).$$

Rule $\Rightarrow$ decomposes the statement into hypotheses and a conclusion. The new conclusion is

$$surjective(g, b, c).$$

MUSCADET does not add a conjunctive hypothesis but as many hypotheses as there are arguments in the conjunction

$$maps(f, a, b),\ maps(g, b, c) \text{ and } surjective(h, a, c).$$

Adding hypotheses is a super-action which is defined by a pack of rules; it involves performing specific actions depending on the formula which has to be added. For conjunctions, as here, this super-action adds elements of the conjunction (and it is recursive). To give another example, for universal properties and implications, it creates rules which are local for the (sub-)theorem being proved.

Rule *defconcl*1 replaces the conclusion by its definition

$$\forall Y (Y \in c \Rightarrow \exists X (X \in b \wedge apply(g, X, Y)).$$

Then rules $\forall$ and $\Rightarrow$ are applied again as described before. A new object y is introduced in c and the conclusion to be proved is

$$\exists X (X \in b \wedge apply(g, X, y)),$$

i.e., the prover has to search for a element X in b such that $apply(g, X, y)$.

Then rule *surjective* adds the hypothesis

$$\exists X (X \in a \wedge apply(h, X, y)).$$

As no other rule can be applied, rule *exists* treats this one existential hypothesis and creates x in a such that $apply(h, x, y)$.

$\forall F \forall A \forall B$(**maps**($F$, A, B) $\Leftrightarrow \forall X(X \in A \Rightarrow \exists Y(Y \in B \wedge$ *apply*$(F, X, Y))$
$\wedge \forall X \forall Y1 \forall Y2(X \in A \wedge Y1 \in B \wedge Y2 \in B \wedge$ *apply*$(F, X, Y1) \wedge$ *apply*$(F, X, Y2) \Rightarrow Y1 = Y2))))$
$\forall F \forall A \forall B$(**surjective**($F$, A, B) $\Leftrightarrow \forall Y(Y \in B \Rightarrow \exists X(X \in A \wedge$ *apply*$(F, X, Y))))$
$\forall F \forall G \forall A \forall B \forall C \forall X \forall Y$(*maps*$(F, A, B) \wedge$ *maps*$(G, B, C) \wedge X \in A \wedge Y \in C \Rightarrow$
(*apply*(**compose**(G, F, A, B, C), X, Y) $\Leftrightarrow \exists Z(Z \in B \wedge$ *apply*$(F, X, Z) \wedge$ *apply*$(G, Z, Y))))$

Fig. 4. Some definitions.

rule *maps*1: if *maps*(F, A, B) and $X \in A$ are hypotheses,
and $\exists Y(Y \in B \wedge$ *apply*$(F, X, Y))$ is not a hypothesis
then add the hypothesis $\exists Y(Y \in B \wedge$ *apply*$(F, X, Y))$
rule *maps*2: 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 *surjective*: if *surjective*(F, A, B) and $Y \in B$ are hypotheses
and (X ($X \in A$ (*apply* (F,X,Y)) is not a hypothesis
then add the hypothesis $\exists X(X \in A \wedge$ *apply*$(F, X, Y))$
rule *compose*1: if *maps*(F, A, B), *maps*(G, B, C), *compose*$(G, F, A, B, C) : H$, $X \in A$, $Y \in C$
and *apply*(H, X, Y) are hypotheses
and $\exists Z(Z \in B \wedge$ *apply*$(F, X, Z) \wedge$ *apply*(G, Z, Y) is not a hypothesis
then add the hypothesis $\exists Z(Z \in B \wedge$ *apply*$(F, X, Z) \wedge$ *apply*(G, Z, Y)

Fig. 5. Some of the rules built from definitions outlined in Fig. 4.

Because of this new element x in a, rules $maps1$ and $compose1$ can now be applied again and they add the hypotheses

$$\exists Y(Y \in b \wedge \textit{apply}(f, x, Y))$$

and

$$\exists Z(Z \in b \wedge \textit{apply}(f, x, Z) \wedge \textit{apply}(g, Z, y)).$$

Again, no other rule can be applied and rule *exists* treats the first existential hypothesis which has not yet been treated

$$\exists Y(Y \in b \wedge \textit{apply}(f, x, Y)).$$

The first rule to be applied depends on the order of the rules, which is the order in which the definitions are given. We will see that the useful existential hypothesis is the second one but this does not matter.[2] This existential hypothesis $\exists Y(Y \in b \wedge$ *apply*$(f, x, Y))$ gives $y1$ in b such that *apply*$(f, x, y1)$.

Because of this new element $y1$ in b, rule *maps*1 can now be applied again and it adds the hypothesis

$$\exists Y(Y \in c \wedge \textit{apply}(g, y1, Y)).$$

Again, no other rule can be applied and rule *exists* treats the first existential hypothesis which has not yet been treated. This is now

$$\exists Z(Z \in b \wedge \textit{apply}(f, x, Z) \wedge \textit{apply}(g, Z, y))$$

which gives the object $y2$ in b such that

apply$(f, x, y2)$ and *apply*$(g, y2, y)$.

Lastly, rule *concl_exist* verifies that the conclusion is satisfied with $y2$ as an instantiation for X. So the conclusion is now *true* and rule *stop* ends the proof by stating that the theorem is proved.

If, at this stage, the theorem was not proved, rule *maps*2 would have added the hypothesis

$$y1 = y2$$

and all occurrences of $y2$ would have been replaced by $y1$. In fact, MUSCADET removes hypotheses which are equalities between two objects and replaces all occurrences of one object by the other one. So the useless element $y1$ would have disappeared as it would be now the same as $y2$, and $y2$ would satisfy all the properties that $y1$ satisfied. This illustrates the way MUSCADET handles the equality predicate in hypotheses. In the conclusions, the equality predicate is handled in the same way as other predicates.

[2] There is in MUSCADET a metarule which can reorder rules. Here rule *compose*1 is moved before rule $maps1$ because it is recognized as more specific, the proof is then shorter and nicer. But this metarule must know the *member relation* (which I consider as a universal symbol). Thus, this possibility is not used in the TPTP context.

rules	objects	hypotheses	conclusion
			$\forall F \forall G \forall A \forall B \forall C \forall X \forall Y$ *maps*(F, A, B) $\wedge$*maps*(G, B, C) $\wedge$*surjective(compose*$(G, F, A, B, C), A, C)$ $\Rightarrow$ *surjective*(G, B, C))
$\forall$	a, b, c, f, g		*maps*$(f, a, b) \wedge$ *maps*(g, b, c) $\wedge$*surjective(compose*$(g, f, a, b, c), a, c)$) $\Rightarrow$ *surjective*(g, b, c)
elifun	h	*compose*$(g, f, a, b, c) : h$	*maps*$(f, a, b) \wedge$ *maps*(g, b, c) $\wedge$*surjective*$(h, a, c) \Rightarrow$ *surjective*(g, b, c)
$\Rightarrow$		*maps*(f, a, b) *maps*(g, b, c) *surjective*(h, a, c)	*surjective*(g, b, c)
*defconcl*1			$\forall Y(Y \in c \Rightarrow \exists X(X \in b \wedge$ *apply*$(g, X, Y))$
$\forall$	y		$y \in c \Rightarrow \exists X(X \in b \wedge$ *apply*$(g, X, y))$
$\Rightarrow$		$y \in c$	$\exists X(X \in b \wedge$ *apply*$(g, X, y))$
surjective		$\exists X(X \in a \wedge$ *apply*$(h, X, y))$	
exists	x	$x \in a$ *apply*(h, x, y)	
*maps*1		$\exists Y(Y \in b \wedge$ *apply*$(f, x, Y))$	
*compose*1		$\exists Z(Z \in b \wedge$ *apply*$(f, x, Z))$ $\wedge$*apply*(g, Z, y)	
exists	$y1$	$y1 \in b$ *apply*$(f, x, y1)$	
*maps*1		$\exists Y(Y \in c \wedge$ *apply*$(g, y1, Y))$	
exists	$y2$	$y2 \in b$ *apply*$(f, x, y2)$ *apply*$(g, y2, y)$	
concl_exist			*true* theorem proved

Fig. 6. MUSCADET proof of theorem SET722+4.

4.2. *Problem* SET737+4

This problem is not conceptually more difficult than problem SET722+4 but it is much more complicated and deserves to be given as another example. There are more mappings and more elements to be created: images, pre-images and intermediary elements. As this process may be expansive, it is important not to develop one direction to the detriment of the others. The creation of elements is delayed to ensure that the elements necessary for the proof are created.

Here is the English statement of the theorem:[3]

Consider three mappings $f : A \rightarrow B$, $g : B \rightarrow C$ and $h : C \rightarrow A$. If $h_o g_o f$ and $f_o h_o g$ are injective and $g_o f_o h$ is surjective then h is one-to-one.

Here is the outline of the MUSCADET proof. In addition to the definitions and rules in Fig. 4 and 5, the definitions and rules in Fig. 7 are used.

By rules $\forall$, *elifun*, $\Rightarrow$ and *defconcl*1 it first creates sets and mappings stated in the theorem with their properties as hypotheses. It then has to prove that h is *one-to-one*, that is that h is *injective* (first sub-theorem) and *surjective* (second sub-theorem).

For the first sub-theorem, by rules *defconcl*1, $\forall$ and $\Rightarrow$, it creates two elements $b1$ and $b2$ in B with the same image $c1$ by g and it has to prove that $b1 = b2$.

Then rules *maps*1, *injective*, *surjective*, *compose*1, *compose*2 and *exists* add successively the following hypotheses:

[3] This theorem is one of the six theorems obtained by replacing f by g or h and by inverting $injective$ and $surjective$. There is no symmetry among them and they are more or less difficult. Theorem SET737+4 is one of the easiest.

$\forall F \forall A \forall B$(**injective**($F$, A, B) $\Leftrightarrow$ $\forall X1 \forall X2 \forall Y(X1 \in A(X2 \in A(Y \in B($*apply*$(f, X1, Y)($*apply*$(f, X2, Y) => X1 = X2))$ $\forall F \forall A \forall B$(**one-to-one**($F$, A, B) $\Leftrightarrow$ *injective*(F, A, B) *and surjective*(F, A, B)
rule *injective*: if *injective*(F, A, B), $X1 \in A$, $X2 \in A$,$Y \in B$, *apply*(f, $X1$, Y) and *apply*(f, $X2$, Y) are hypotheses then add the hypothesis $X1 = X2$ rule *compose*2: if *maps*(F, A, B), *maps*(G, B, C), *compose*(G, F, A, B, C) : H, $X \in A$, $Y \in B$, $Z \in C$, *apply*(F, X, Y) and *apply*(G, Y, Z) are hypotheses and *apply*(H, X, Z) is not a hypothesis then add the hypothesis *apply*(H, X, Z)

Fig. 7. Some more definitions and rules.

$\exists Y(Y \in C \wedge$ *apply*$(g, b1, Y)$ (1)
$\exists Y(Y \in C \wedge$ *apply*$(g, b2, Y)$ (2)
$\exists Y(Y \in A \wedge$ *apply*$(h, c1, Y)$ (3)
$\exists X(X \in C \wedge$ *apply*$(g_o f_o h, X, c1)$ (4)
(1) and (2) are not treated because they are already satisfied
$a1 \in A$ and *apply*$(h, c1, a1)$ (treatment of (3))
$\exists Y(Y \in B \wedge$ *apply*$(f, a1, Y)$ (5)
apply$(h_o g, b1, a1)$)
$\exists Z(Z \in C \wedge$ *apply*$(g, b1, Z) \wedge$ *apply*$(h, Z, a1))$(6)
apply$(h_o g, b2, a1)$)
$\exists Z(Z \in C \wedge$ *apply*$(g, b1, Z) \wedge$ *apply*$(h, Z, a1))$ (6)
apply$(h_o g, b2, a1)$)
$\exists Z(Z \in C \wedge$ *apply*$(g, b2, Z) \wedge$ *apply*$(h, Z, a1))$ (7)
$c2 \in C$ and *apply*$(g o f o h, c2, c1)$ (treatment of (4), useless)
$\exists Y(Y \in A \wedge$ *apply*$(h, c2, Y)$ (8)
$\exists X(X \in C \wedge$ *apply*$(g_o f_o h, X, c2)$ (9)
$\exists Z(Z \in B \wedge$ *apply*$(f_o h, c2, Z) \wedge$ *apply*$(g, Z, c1))$ (10)
$b3 \in B$ and *apply*$(f, a1, b3)$ (treatment of (5))
$\exists Y(Y \in C \wedge$ *apply*$(g, b3, Y)$ (11)
apply$(f_o h_o g, b1, b3)$
$\exists Z(Z \in A \wedge$ *apply*$(h_o g, b1, Z) \wedge$ *apply*$(f, Z, b3))$ (12)
apply$(f_o h_o g, b2, b3)$
$b1 = b2$.

This is the conclusion of sub-theorem 1 which is then proved.

For the second sub-theorem, by rules *defconcl*1 and $\forall$, it creates $c1$ in C and it has to prove that $\exists X(X \in B \wedge$ *apply*$(g, X, c1))$.

Then rules *maps*1, *surjective*, *compose*1, *compose*2, *exists* and *concl_exist* successively add the following hypotheses:

$\exists Y(Y \in A \wedge$ *apply*$(h, c1, Y)$ (1)
$\exists X(X \in C \wedge$ *apply*$(g_o f_o h, X, c1)$ (2)
$a1 \in A$ and *apply*$(h, c1, a1)$ (treatment of (1))
$\exists Y(Y \in B \wedge$ *apply*$(f, a1, Y)$ (3)
$c2 \in C$ and *apply*$(g o f o h, c2, c1)$ (treatment of (2))
$\exists Y(Y \in A \wedge$ *apply*$(h, c2, Y)$ (4)
$\exists X(X \in C \wedge$ *apply*$(g_o f_o h, X, c2)$ (5)
$\exists Z(Z \in C \wedge$ *apply*$(f_o h, c2, Z) \wedge$ *apply*$(g, Z, c1))$ (6)
$b1 \in B$ and *apply*$(f, a1, b1)$ (treatment of (3))
$\exists Y(Y \in C \wedge$ *apply*$(g, b1, Y)$ (7)
apply$(f_o h, c1, b1)$
$\exists Z(Z \in B \wedge$ *apply*$(f_o h, c1, Z) \wedge$ *apply*$(g, Z, b1))$ (8)
$a2 \in A$ and *apply*$(h, c2, a2)$ (treatment of (4), useless)
$\exists Y(Y \in B \wedge$ *apply*$(f, a2, Y)$ (9)
$c3 \in C$ and *apply*$(g_o f_o h, c3, c2)$ (treatment of (5), useless)
$\exists Y(Y \in A \wedge$ *apply*$(h, c3, Y)$ (10)
$\exists X(X \in C \wedge$ *apply*$(g_o f_o h, X, c3)$ (11)
$\exists Z(Z \in C \wedge$ *apply*$(f_o h, c3, Z) \wedge$ *apply*$(g, Z, c2))$ (12)
$b2 \in B$, *apply*$(f_o h, c2, b2)$ and *apply*$(g, b2, c1)$ (treatment of (6)).

The conclusion

$$\exists X(X \in B \wedge \textit{apply}(g, X, c1))$$

is now satisfied, so sub-theorem 2 is proved.

As both sub-theorems are proved, the initial theorem is proved.

5. Processing of negation – positive and negative properties

MUSCADET works as much as possible without negations. Not only does it work with hypotheses and conclusions instead of negative literals but it eliminates negations whenever it is possible.

If the conclusion to be proved is a negation $\neg C$, it adds C as a new hypothesis and the new conclusion is *false*. This means that it will have to find a contradiction, i.e., to deduce the hypothesis *false*. If the conclu-

sion is $\neg C1 \vee C2$, it adds $C1$ and the new conclusion is $C2$.

From definitions in which a negation $\neg P$ occurs, MUSCADET builds several rules which are logically equivalent. Some rules contain the negative literal $\neg P$ in the places where it appeared in the definition. Other rules contain the positive literal P as a condition (resp. conclusion) if $\neg P$ was on the right (resp. left) or an implication.

For example, from the definition

$\forall A \forall B \forall X (X \in \textit{compl}(E, A) \Leftrightarrow X \in E \wedge \neg X \in A))$

it builds the following rules

rule *compl*1: if *compl*$(E, A) : B$ and $X \in B$ are hypotheses
then add the hypothesis $X \in E$

rule *compl*2: if *compl*$(E, A) : B$, $X \in B$ and $X \in A$ are hypotheses
then add the hypothesis *false*

rule *compl*3: if *compl*$(E, A) : B$, $X \in E$ and $\neg X \in A$ are hypotheses
then add the hypothesis $X \in B$

rule *compl*4: if *compl*$(E, A) : B$ and $X \in E$ are hypotheses
then add the hypothesis $X \in A \vee X \in B$.

Negations occur every time a problem contains the *empty set* ϕ or $complements$ or *set differences*. In these cases, proofs are often proofs by contradiction and resolution-based provers are well adapted to them.

Problems SET763+4 and SET764+4 involve the empty set ϕ. In CASC-JC they were proved, respectively, by three and five entrants out of seven.

To prove these theorems, MUSCADET applies in particular the following rules

rule ϕ*: if* $X \in \phi$ *is a hypothesis*
then add the hypothesis false

which has been built from the definition of the empty set ϕ,

$\forall X \neg X \in \phi$

and the universal logical rule

rule stop*1: if* false *is a hypothesis*
then the new conclusion is true.

I will rather describe the MUSCADET proof of problem SET770+4 which involves $disjoint$ sets and which was proved only by MUSCADET in CASC-JC.
If R *is an equivalence relation on* E *and* A *and* B *two elements of* E *then the equivalence classes* $\overline{A}$ *and* $\overline{B}$ are equal or disjoint.

In this problem the definition itself of $disjoint$ is a negation.

$\forall A \forall B (\textit{disjoint}(A, B) \Leftrightarrow \neg \exists X (X \in A \wedge X \in B))$.

It is the property *non disjoint* which is a positive property. If the definition of a concept C is a negation, MUSCADET creates the concept $nonC$ the definition of which is positive and defines C as $\neg nonC$. For this reason, it defines and uses the concept *nondisjoint*.

It adds the two following definitions

$\forall A \forall B (\textit{disjoint}(A, B) \Leftrightarrow \neg \textit{nondisjoint}(A, B))$
$\forall A \forall B (\textit{nondisjoint}(A, B) \Leftrightarrow \exists X (X \in A \wedge X \in B)$

and builds the rules

rule *disjoint*1: if *disjoint*(A, B) and *nondisjoint*(A, B) are hypotheses
then add the hypothesis *false*

rule *disjoint*2: if *disjoint*(A, B) is a hypothesis
then add the hypothesis $\neg$*nondisjoint*(A, B)

rule *nondisjoint*:
if *nondisjoint*(A, B) is a hypothesis and $\exists X (X \in A \wedge X \in B)$ is not a hypothesis
then add the hypothesis
$\exists X (X \in A \wedge X \in B)$.

MUSCADET introduces a set e, an equivalence relation r, two elements a and b, names the classes *class*$(a, e, r) : aa$ and *class*$(b, e, r) : bb$ which contain respectively a and b. The conclusion is

$aa = bb \vee$ *disjoint*(aa, bb)

which is replaced (definition de $\subset$) by

$(aa \subset bb \wedge bb \subset aa) \vee$ *disjoint*(aa, bb)

then (logical rule) by

$(aa \subset bb \vee$ *disjoint*$(aa, bb))$
$\wedge (bb \subset aa \vee$ *disjoint*$(aa, bb))$.

The conclusion of the first sub-theorem is

$aa \subset bb \vee$ *disjoint*(aa, bb)

which is replaced by

$\forall X(X \in aa \Rightarrow X \in bb) \vee$ *disjoint*(aa, bb)

then by

$\forall X(X \in aa \Rightarrow X \in bb) \vee \neg$*nondisjoint*$(aa, bb)$.

Then the hypothesis

nondisjoint(aa, bb)

is added and the new conclusion is

$\forall X(X \in aa \Rightarrow X \in bb)$.

A new object x is created in aa and the conclusion is $x \in bb$. Then the hypothesis xra is added.

Then rule *nondisjoint* adds the hypothesis

$\exists X(X \in aa \wedge X \in bb)$

which is treated and gives y which belongs to aa and to bb. It is then easy to conclude, by rules built from the definitions of *equivalence relation* and of *classes*, that yra, yrb, ary, xry, xrb, and $x \in bb$ which is the conclusion.

The proof of the second sub-theorem is analogous.

The efficiency of MUSCADET comes from its methods which are adapted to 'positive' properties and to its ability to create the intermediate 'positive' $nondisjoint$ property.

6. Building rules

The construction of rules is performed by the metarule $buildrules$ which is recursive and composed of metarules which analyze axioms. A frame is first built which depends on the nature of the axiom (definition of a predicate or functional symbol, or lemma) and it is filled step by step by the recursive super-action.

The arguments of the call

buildrules(E, R)

are the statement E to analyze and the (partially) built rule R.

In the TPTP context, it is not possible to define statements as definitions or lemmas, but the system recognizes them because they are universal closures of equivalences between an expression of the form $P(X_1, X_2, \ldots)$ or $Q(\ldots, F(X_1, X_2, \ldots), \ldots)$ and another expression which does not contain occurrences of P or F. The names of the built rules are formed with the name P or F and a number if several rules are built.

6.1. First case: building rules from definitions of predicates

From the definition of the concept $\subset$,

$\forall A \forall B(A \subset B \Leftrightarrow \forall X(X \in A \Rightarrow X \in B))$

the first call to *buildrules* is

buildrules$(A \subset B \Rightarrow \forall X(X \in A \Rightarrow X \in B))$,
rule $\subset$:).

Then the left part of $\Rightarrow$ gives the first condition, the right part gives a new expression to analyze, and the next calls are

buildrules$(\forall X(X \in A \Rightarrow X \in B)$,
rule $\subset$*: if* $A \subset B$ *is a hypothesis*)

then, because the universal variables which are now implicit,

buildrules$(X \in A \Rightarrow X \in B$,
rule $\subset$*: if* $A \subset B$ *is a hypothesis*)

and, as before because of the $\Rightarrow$,

buildrules$(X \in B$,
rule $\subset$*: if* $A \subset B$ *and* $X \in A$
are hypotheses)

The expression is now elementary and the last call, with an empty expression,

buildrules(),
rule $\subset$*: if* $A \subset B$ *and* $X \in A$
are hypotheses
and $X \in B$ *is not a hypothesis,*
then add the new hypothesis $X \in B$)

adds the rule

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

which we have seen in Section 3.

Applying such built rules amounts to the same thing as applying resolution on clauses, but only in a restricted manner. In particular, 'positive' properties are favored and this is one of the reasons why MUSCADET quickly attains its goals in many situations. Nevertheless, MUSCADET is able to handle negations whenever necessary, especially if negations occur in the defined concepts. We have seen an example of this in Section 5.

6.2. Second case: building rules from definitions of functional definitions

For the definition of union,

$$\forall A \forall B \forall X (X \in A \cup B \Leftrightarrow X \in A \vee X \in B)$$

the first call to *buildrules* is

buildrules($X \in C \Leftrightarrow X \in A \vee X \in B$,
rule $\cup$*: if* $A \cup B : C$ *is a hypothesis*)

the existence of the union is stated in the first condition and $A \cup B$ is replaced by C in the definition. Because of the $\Leftrightarrow$, the expression is split into two sub-formulas

$$X \in C \Leftrightarrow X \in A \vee X \in B$$

and

$$X \in A \vee X \in B \Leftrightarrow X \in C$$

and there are two calls to $buildrules$.
The first call

buildrules($X \in C \Rightarrow X \in A \vee X \in B$,
rule $\cup$1*: if* $A \cup B : C$ *is a hypothesis*)

leads to the rule

rule $\cup$1*: if* $A \cup B : C$ *and* $X \in C$ *are hypotheses,*
and $X \in A \vee X \in B$ *is not a hypothesis*
then add the hypothesis $X \in A \vee X \in B$.

The second call

buildrules($X \in A \vee X \in B \Rightarrow X \in C$,
rule $\cup$2*: if* $A \cup B : C$ *is a hypothesis*)

splits the expression because of the $\vee$ and there are again two calls to *buildrules*.

buildrules($X \in A \Rightarrow X \in C$,
rule $\cup$21*: if* $A \cup B : C$ *is a hypothesis*)

leads to the rule

rule $\cup$21*: if* $A \cup B : C$ *and* $X \in A$ *are hypotheses,*
and $X \in C$ *is not a hypothesis*
then add the hypothesis $X \in C$.

buildrules($X \in B \Rightarrow X \in C$,
rule $\cup$22*: if* $A \cup B : C$ *is a hypothesis*)

leads to the rule

rule $\cup$22*: if* $A \cup B : C$ *and* $X \in B$ *are hypotheses,*
and $X \in C$ *is not a hypothesis*
then add the hypothesis $X \in C$.

This example shows the importance of eliminating functional symbols by flattening the terms. If the system worked with terms, the construction of rules would have led to rules such as the rule

if $X \in A$ *is a hypothesis*
then add the hypothesis $X \in A \cup B$

which is expansive and could introduce infinitely many unions. Instead of this, unions are considered only if they have already been introduced for another reason.

6.3. Other cases: building rules from axioms and universal hypotheses

From axioms which are not recognized as definitions and are considered as lemmas and from universal hypotheses, the first call to $buildrules$ is simply the following

buildrules(E,
rule ri:)

Rules built from universal hypotheses are only local to the (sub-)theorem in which they were built because they could contain objects which were created in this (sub-)theorem.

7. Forward-chaining and backward-chaining

Most strategies are forward-chaining. But there are two exceptions.

The first exception concerns the rules *defconcl*1 and 2 seen in Section 3 which replace the conclusion by a definition (direct or indirect in the case of functional symbols).

The second backward strategy is used in the case where a universal condition would be generated in a rule. This arises with some rather complicated axioms such as the axiom

$\forall E \forall To$ (*environment*(*E*)
$\wedge$ (*growth_rate*(*efficient_producers,To*)
$>$ *growth_rate*(*first_movers,To*))
$\wedge$ *in_environment*(*E,To*)
$\wedge \forall T$ (*subpopulations*(*first_movers,*
efficient_producers,E,T)
$\wedge\ T > To$
$\Rightarrow$ *growth_rate*(*efficient_producers,T*)
$>$ *growth_rate*(*first_movers,T*))
$\Rightarrow To =$ *critical_point*(*E*))

of TPTP problem MGT023+1 (which was proved by MUSCADET and all CASC-JC entrants except one).

From this axiom (a lemma for MUSCADET), the following rule is automatically built.

if *environment*(E),
  growth_rate(*efficient_producers*,*To*):A,
  growth_rate(*first_movers*,*To*):B,
  $\neg A > B$
    are hypotheses,
  the conclusion is $C = P(X1, \ldots, Xn)$
    and there is a hypothesis
    $H = P(Y1, \ldots, Yn)$
then the new conclusion is
  $\forall T$ (*subpopulations*(*first_movers*,
    efficient_producers,*E*,*T*)
    $\wedge\ T > To$
    $\Rightarrow$ *growth_rate*(*efficient_producers*,*T*)
      $>$ *growth_rate*(*first_movers*,*T*))
    $\wedge\ To =$ *critical_point*(E) $\Rightarrow C$)

This means that, after having verified that there is a hypothesis H which involves the same property P as the conclusion, the prover will have to verify all the conditions of the axiom except H and the universal condition and to prove two properties. The first property is the instantiated universal condition. The second property is the fact that the conclusion of the axiom implies the conclusion of the theorem to be proved.

This backward strategy still needs to be improved and generalized.

8. Weaknesses of MUSCADET

MUSCADET has very poor results in the FNE category. It proved only two theorems of this category at CASC-JC. The reason is not the presence or absence of equality in FEQ or FNE. The crucial difference is the following. FEQ contains problems which are real problems (mathematical or not) with many axioms, definitions or lemmas. These problems are more or less expressed as human beings express them. This is the case for the problems in Sections 4 and 5. The more definitions there are, the more rules are built and the more MUSCADET strategies are efficient. The MUSCADET strategies, in particular the construction of rules from the axioms and the processing of existential properties, are rather well adapted to these domains. By contrast, FNE contains logical problems which are sometimes expressed in a non-natural manner. They often contain only one very large formula, the conjecture, and no axioms or intermediate definitions. There are also very large conjectures which probably result from and were automatically generated from problems expressed in non-classical logics. In these circumstances, MUSCADET cannot build enough rules. Humanlike methods are not useful and MUSCADET performs so many splittings that it cannot prove the theorems.

Even in FEQ the performances of MUSCADET depend on the domains. MUSCADET is efficient for mathematical everyday problems which are expressed in a natural manner, for example naive set theory. It is less efficient for problems which are defined axiomatically, from a logician's point of view (e.g., in axiomatic set theory, or axiomatic geometry). In everyday mathematical problems, we may use *mappings*, consider that they are primitive concepts and only use their properties (cf. examples in Section 4). In axiomatic set theory, *mappings* are particular *relations* which are subsets of *cross-products of sets*.

In axiomatic theories, MUSCADET has difficulty with some axioms that state the existence of objects and generate infinitely many objects. Improvements will be made for the handling of such axioms.

MUSCADET only proves about half of the MGT problems, although low priority rules – as described in Section 7 – are efficient for some of them.

MUSCADET does not prove any theorem about groups. The 'elimination' of functional symbols is not adapted to this field in which the terms may be deep and in which it is often useful to consider them as a whole.

9. Related work

MUSCADET is not the only prover which uses natural deduction or knowledge bases.

The theorem prover Isabelle has been used in set theory with natural deduction [11].

The Ω-MKRP system has been developed over the years [7–9] and uses knowledge of the particular mathematical fields and knowledge about its potential use.

The study of tactics is another related field. They are applications of calculus rules, abbreviations for sequences of calculus rules [6], operationalization of a reformulation [10], programs that construct proofs [1] and may perhaps be compared to metarules.

These provers use types to express theorems and higher order logic. They may also give specific knowledge in their own language and representations for spe-

cific mathematical knowledge. This is not possible in the TPTP context and especially in the CASC competitions.

Although MUSCADET is also able to receive specific knowledge expressed by rules and to work in specific mathematical domains, its strength in the TPTP context is that it is able to receive first order formulas and to build efficient rules from axioms without knowing anything about the particular symbols used.

10. Conclusion

MUSCADET works in a manner which is quite different from resolution-based provers. It uses methods based on natural deduction and knowledge-based systems. We have seen some of these methods which are crucial and which explain why MUSCADET is able to prove some theorems that resolution-based provers are not able to prove. However, it is not able to prove some theorems that all resolution-based provers are able to prove. Why can't we make them cooperate? We, human beings, do not use the same methods for all problems that we have to solve. We can choose one method or another to solve a problem. In some cases, we begin with a method, then try another, and perhaps even another and sometimes we come back to the first one and succeed with it. For some problems, we reason at a metalevel or in higher order logic. To veritably improve theorem provers, it would be interesting to have several provers working together, in sequence, as we do when we successively try several methods or, in parallel, as computers are able to do, or better still to have a top-level mechanism which analyses the problem and chooses one of the provers. This analysis was begun in Section 8, and should be continued. MUSCADET itself should be able to select the most appropriate prover If it was provided with the necessary knowledge.

Acknowledgements

I would like to thank Geoff Sutcliffe and Christian Suttner for creating and updating the TPTP Problem Library, and the referees for providing numerous suggestions for improving this paper.

References

[1] D.A. Basin and R.L. Constable, Metalogical frameworks, in: *Logical Environments*, G. Huet and G. Plotkin, eds, Cambridge University Press, 1993, pp. 1–29.

[2] W.W. Bledsoe, Splitting and reduction heuristics in automatic theorem proving, *Journal of Artificial Intelligence* **2** (1971), 55–77.

[3] W.W. Bledsoe and P. Bruell, A man-machine theorem-proving system, *Journal of Artificial Intelligence* **5** (1974), 51–72.

[4] W.W. Bledsoe, Non-resolution theorem proving, *Journal of Artificial Intelligence* **9** (1977), 1–35.

[5] W.W. Bledsoe and M. Tyson, The UT interactive theorem prover, The University of Texas at Austin, Dept. Memo ATP-17, 1978.

[6] R.L. Constable et al., *Implementing Mathematics with the Nuprl Proof Development System*, Prentice-Hall, Englewood Cliffs, 1986.

[7] N. Eisinger, J. Siekmann and G. Smolka, The Markgraf Karl refutation procedure, *IJCAI* (1981), 511–518.

[8] N. Eisinger and H.J. Ohlbach, The Markgraf Karl Refutation Procedure (MKRP), in: *Procedings of the 8th CADE*, J.H. Siekmann, ed., Springer Verlag, Berlin, 1986.

[9] X. Huang, M. Kerber, M. Kohlhase, E. Melis, D. Nesmith, J. Richts and J. Siekmann, The Ω-MKRP proof development environment, in: *ECAI 94 Wokshop From Theorem Provers to Mathematical Assistants: Issues and Possible Solutions*.

[10] M. Kerber and A. Pracklein, Using tactics to reformulate formulae for resolution theorem proving, *Annals of Mathematics and Artificial Intelligence* **18**(2) (1996).

[11] P.A.J. Noel, Experimenting with Isabelle in ZF set theory, *Journal of Automating Reasoning* **10** (1993), 15–58.

[12] D. Pastre, MUSCADET: an automatic theorem proving system using knowledge and metaknowledge in mathematics, *Journal of Artificial Intelligence* **38**(3) (1989).

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

[14] D. Pastre, MUSCADET version 2.3, User's manual, 2001, http://www.math-info.univ-paris5.fr/~pastre/muscadet/manual-en.ps.

[15] D. Pastre, MUSCADET version 2.3, Examples, 2001, http://www.math-info.univ-paris5.fr/~pastre/muscadet/examples.

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

[17] G. Sutcliffe and C. Suttner, The TPTP Problem Library for automated theorem proving, http://www.cs.miami.edu/~tptp.