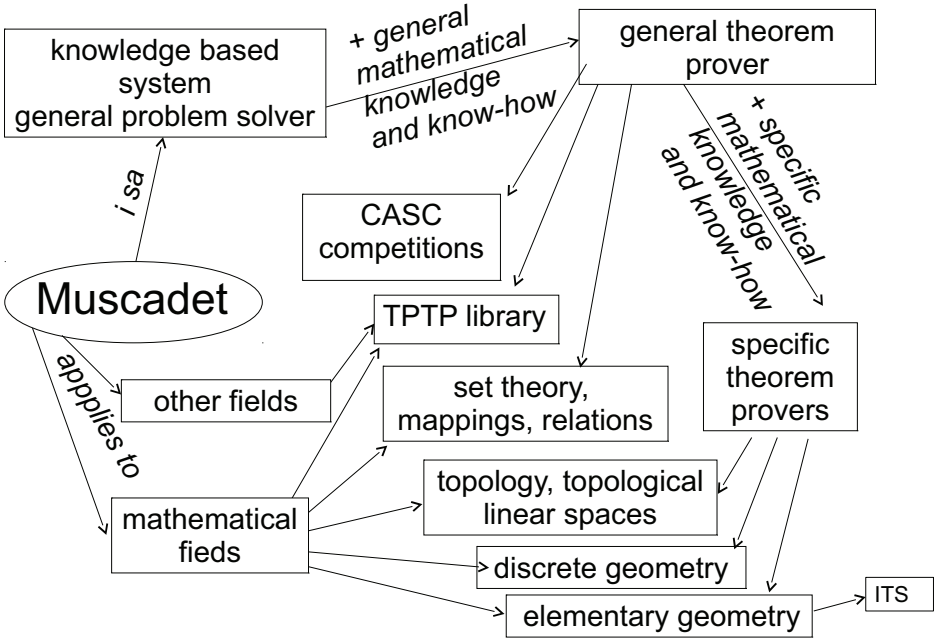
Mathematical Theorem Proving from Muscadet0 to Muscadet4 Why and How

Dominique Pastre

Université Paris Descartes
pastre@www.math-info.univ-paris5.fr
http://www.math-info.univ-paris5.fr/~pastre/muscadet/muscadet.html

Workshop about Sets and Tools (SETS 2014)
Toulouse, June 3rd 2014



Muscadet0 to Muscadet4

- A bit of history
- Muscadet
  - 0 - A first program based on “natural deduction”
  - 1 - A knowledge-based prover
  - 2 - A Prolog prover
  - 3 - A TPTP prover
  - 4 - Relevant proofs and tools
- Efficiency
- Conclusion

A famous paper, a good team

- Woody Bledsoe

  Splitting and reduction heuristics in automatic theorem proving,

  Journal of Artificial Intelligence, 1971

- Jacques Pitrat's team

  Artificial Intelligence, knowledge and metaknowledge

A bit of history

- Logic Theorist : first theorem prover
  Newel, Shaw, Simon (1957)
  then NSS (chess playing program, 1958) and
  GPS (General Problem Solver, 1961)
- Gelertner (geometry, 1959)
- Wang (logics (sequents), 1960)
- Slagle (integration, 1961)

Revolution and counter-revolution
Resolution Principle vs natural deduction

- Robinson, A machine oriented logic based on the Resolution Principle, 1965
  - Skolemisation
  - CNF
  - clauses, literals
  - one rule
  - completude for predicate calculus
- Bledsoe, Splitting and reduction heuristics in automatic theorem proving, 1971
  - sequents
  - hypotheses and conclusions
  - rewriting rules
  - splitting
  - reduce

SPLIT

general rules

	INPUT	OUTPUT
• splitting :	$A \wedge A$	two theorems A and B
• rewriting rules		
	$p \rightarrow (A \wedge B)$	$(p \rightarrow A) \wedge (p \rightarrow B)$
	$p \vee q \rightarrow A$	$(p \rightarrow A) \wedge (q \rightarrow A)$
	$p \rightarrow (A \rightarrow B)$	$p \wedge A \rightarrow B$
	$A \rightarrow \forall x\, P(x)$	$A \rightarrow P(y)$ (new variable)
	$\exists x\, P(x) \rightarrow D$	$P(y) \rightarrow D$ "
	$x = y \rightarrow P(x,y)$	P(y,y) if x is a variable

REDUCE

rules for set theory

INPUT	OUTPUT
$s \in A \cap B$	$s \in A \wedge s \in B$
$s \in A \cup B$	$s \in A \vee s \in B$
$s \in \mathcal{P}(A)$	$s \subset A$
$C \subset A \cap B$	$C \subset A \wedge C \subset B$
$A \cup B \subset C$	$A \subset B \wedge A \subset$

general rules

$\neg (A \wedge B)$	$\neg A \vee \neg B$
$\neg (A \vee B)$	$\neg A \wedge \neg B$
$\neg \forall x\, A$	$\exists x \neg A$
$\neg \exists x\, A$	$\forall x \neg A$

Theorem to be proved

$\mathcal{P}(A)\cap\mathcal{P}(B) = \mathcal{P}(A\cap B)$

$\mathcal{P}(A)\cap\mathcal{P}(B) \subset \mathcal{P}(A\cap B) \wedge \mathcal{P}(A\cap B) \subset \mathcal{P}(A)\cap\mathcal{P}(B)$

(1)

$\mathcal{P}(A)\cap\mathcal{P}(B) \subset \mathcal{P}(A\cap B)$

$t \in \mathcal{P}(A)\cap\mathcal{P}(B) \rightarrow t \in \mathcal{P}(A\cap B)$

$t \subset A \wedge t \subset B \rightarrow t \subset A \cap B$

$t \subset A \wedge t \subset B \rightarrow t \subset A \wedge t \subset B$

true

(2)

$\mathcal{P}(A\cap B) \subset \mathcal{P}(A)\cap\mathcal{P}(B)$

...

proof similar

...

true

A first programme based on "natural deduction"

(Datte, alias Muscadet0)

General rules

- to prove $\forall$x P(x) take any x1 and prove P(x1)
- to prove A $\Rightarrow$B, assume A and prove B
- to prove $A_1 \wedge A_2 \wedge ... \wedge A_n$ prove all the A_i one after the other
- to prove C, if A$\vee$B is a hypothesis, prove (A$\Rightarrow$C)$\wedge$(B$\Rightarrow$C)
- to prove $\neg$ A, assume A and search for a contradiction

To assume A

- if A is elementary, or is an existential formula, add it as a new hypothesis
- if A is a conjunction $A_1 \wedge A_2 \wedge ... \wedge A_n$ assume each A_i
- if A is a universal hypothesis, it is not added as a new hypothesis, new rules are built

hypotheses may be

- elementary (p(...))
- a disjunction
- an existential formula
- false

the conclusion may be (only one)

- elementary (p(...))
- a disjunction
- an existential formula
- true

Definitions and building rules

- A conclusion may be replaced by its definition
- Hypotheses are never replaced by their definition

- Rules are automatically built from definitions

  Example

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

  Rule : if $A \subset B$ and $X \in A$ are hypotheses

  then add the hypothesis $X \in B$

Theorem to be proved

$\forall A \forall B\ \forall C\ (A \subset B \wedge B \subset C \Rightarrow A \subset C)$

elements	hypotheses	conclusion
		$\forall C \forall A \forall B\ \forall C\ (A \subset B \wedge B \subset C \Rightarrow A \subset C)$
a, b, c		$a \subset b \wedge b \subset c \Rightarrow a \subset c$
	$a \subset b$	
	$b \subset c$	$a \subset c$
		$\forall X\ (X \in a \Rightarrow X \in c)$
x		$x \in a \Rightarrow x \in c$
	$x \in a$	$x \in c$
	$x \in b$	
	$x \in c$	
		true

Functional symbols definitions and rules

Examples

definition of $\mathcal{P}(A)$ $\quad \forall A\ \forall B \in X\ (X \in \mathcal{P}(A) \Leftrightarrow X \subset A)$

For Datte $\quad \forall A \forall B(\text{powerset}(B,A) \Rightarrow \forall X(X \in B \Leftrightarrow X \subset A))$

powerset(B,A) will be noted here $B:\mathcal{P}(A)$

rules

- to prove $X \in B$, if $B:\mathcal{P}(A)$ is a hypothesis then prove $X \subset A$
- if $B:\mathcal{P}(A)$ and $X \in B$ are hypotheses,
  then add the hypothesis $X \subset A$

definition of intersection

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

rules :

- if $C:A \cap B$ and $x \in C$ are hypotheses
  then add the hypothesis $x \in A$
- if $C:A \cap B$ and $x \in C$ are hypotheses
  then add the hypothesis $x \in B$
- if $C:A \cap B$, $x \in A$ and $x \in B$ are hypotheses
  then add the hypothesis $x \in C$
- to prove $x \in C$, if $C:A \cap B$ is a hypothesis
  then prove $X \in A \wedge X \in B$

Remark : there is no rule of the form

if $x \in A$ and $x \in B$ are hypotheses
then add the hypothesis $x \in A \cap B$

which would be expansive

Proof of the theorem
$\forall A \forall B(\mathcal{P}(A \cap B) \subset \mathcal{P}(A) \cap \mathcal{P}(B))$

	objects	hypotheses	conclusion
			$\forall A \forall B(\mathcal{P}(A \cap B) \subset \mathcal{P}(A) \cap \mathcal{P}(B))$
	a, b		$\mathcal{P}(a \cap b) \subset \mathcal{P}(a) \cap \mathcal{P}(b)$
	c, pc	c:a∩b, pc:$\mathcal{P}$(c)	
	pa, pb	pa:$\mathcal{P}$(a), pb:$\mathcal{P}$(b)	
	pd	pd:pa∩pb	pc ⊂ pd
			$\forall X\ (X \in pc \Rightarrow X \in pd)$
	x		$x \in pc \Rightarrow x \in pd$
		$x \in pc$	$x \in pd$
		$x \subset c$	
			$x \in pa \wedge x \in pb$

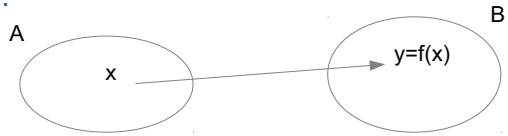
split in Theorem 1 and Theorem 2

	objects	hypotheses	conclusion
Theorem 1		...	$x \in pa$
			$x \subset a$
			$\forall X\ (X \in x \Rightarrow X \in a)$
	t	$t \in x$	$t \in a$
		$t \in c$	
		$t \in a$	Theorem 1 proved
Theorem 2		...	$x \in pb$
			...
			Theorem 2 proved
			Theorem proved

Processing of the existential hypotheses

Systematically creating objects could be expansive. So, the processing of existential hypotheses has a low priority and these hypotheses are handled one after the other, in the order where they appeared, and all the other rules are tried again before processing the next one.

Example : If f maps A into B, then each element x of A has an image in B.

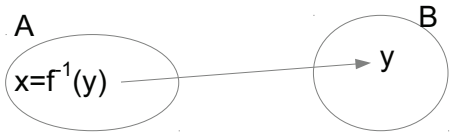


Special case, if f maps A into A :

$a \rightarrow a_1=f(a) \rightarrow a_2=f(a_1) \rightarrow a_3=f(a_2) \rightarrow \ldots$

All that can be deduced from the l'hypothesis $a_i=f(a_{i-1})$ is deduced before the creation of a_{i+1}.

If moreover f is surjective, each element y of B has an antecedent in A.



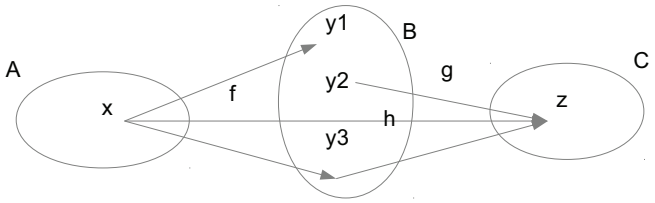
Special case, if f maps A onto A :

$\ldots \rightarrow a_4=f^{-1}(a_2) \rightarrow a_2=f^{-1}(a) \rightarrow a \rightarrow a_1=f(a) \rightarrow a_3=f(a_1) \rightarrow \ldots$

an image and an antecedent are created alternately.

Moreover, if there are several mappings, images and antecedents are created alternately for all mappings.

If f maps A into B,
then each element x in A has an image y_1 in B.
If g maps B onto C,
then each element z in C has an antecedent y_2 in B.
If h is the composition (from A into C) of f and g, and if z=h(x),
then there is an element y_3 in B such that y=f(x) and z=g(y)



Then $y_1=y_3$ and, if g is injective, $y_2=y_3$.
Rather than creating y_1, then y_2 and y_3, it is better to only create y_3 which verifies the three properties.

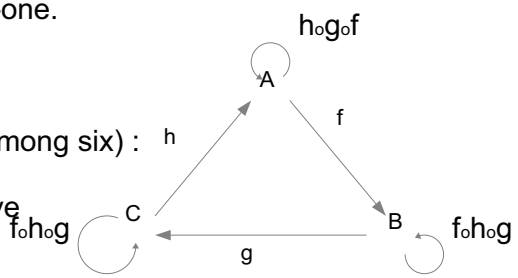
Reordering rules

The rules which may create more specific objects must have higher priority than others
Metarule : if the rule R may create an element a such that P
the rule R' may create an element b such that Q
P is more general than Q
then R' must be applied before R
More precisely, the metarule is the following (of which it is a restriction) :
if the rule R contains the action add-hyp $\exists x \in A\ C$
the rule R' contains the action add-hyp $\exists x' \in A\ C'$
C' is a conjunction of terms and
one of them is equal to C modulo x and x'
then apply R' before R

Example in set theory on mappings

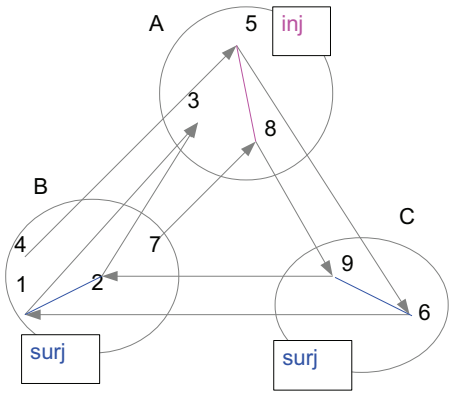
Theorem : Consider three mappings f, g, h from A into B, B into C, C into A; if among the three mappings $h \circ g \circ f$, $g \circ f \circ h$, $f \circ h \circ g$, two are injective (resp. surjective) and the third is surjective (resp. injective), then f, g and h are one-to-one.

For example (one case among six) :
$h \circ g \circ f$ injective
$g \circ f \circ h$ and $f \circ h \circ g$ surjective

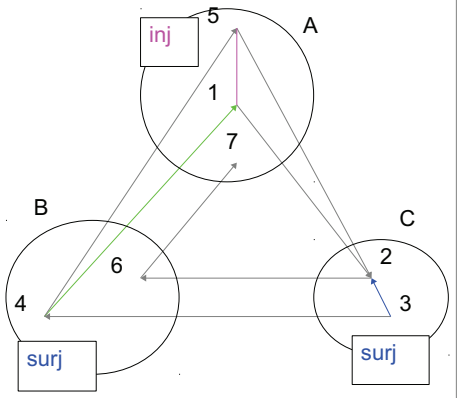


Case $h \circ g \circ f$ injective, $g \circ f \circ h$ and $f \circ h \circ g$ surjective (one case among six)

h injective
if 1 and 2 have the same image 3, then they are equal



h surjective
4 is a pre-image de 1 because 1 is equal to its image 5



Processing of disjunctive hypotheses

- rule : To prove C, if $A \vee B$ is a hypothesis
  prove $((A \Rightarrow C) \wedge (B \Rightarrow C))$

this gives then two subtheorems with either A or B as a hypothesis, and C as the conclusion

- Applied too early, this rule could give two subtheorems with the same proof.
- If the hypothesis $A \vee B$ is useless, this rule may be expansive,

- Disjunctive hypotheses are handled after existential ones.
  No real reason, only experiments.
  It is easy to modify the priority to choose the opposite

Negation – positive and negative properties

Negations are removed as much as possible.

exceptions (mathematical notations and/or natural language)

- different $x \neq y$ for $\neg (x=y)$ frequently useful
- not-in $x \notin y$ for $\neg (x \in y)$ useful for complements of sets
- non-empty(A) $\Leftrightarrow \exists x\ (x \in A)$
- non-disjoint(A,B) $\Leftrightarrow \exists x\ (x \in A \wedge x \in B)$

In following versions, Muscadet receives the definitions

empty(A) $\Leftrightarrow \neg \exists x\ (x \in A)$

disjoint(A,B) $\Leftrightarrow \neg \exists x\ (x \in A \wedge x \in B)$

and replaces them by

non-empty(A) $\Leftrightarrow \exists x\ (x \in A)$

empty(A) $\Leftrightarrow \neg$ non-empty(A)

non-disjoint(A,B) $\Leftrightarrow \exists x\ (x \in A \wedge x \in B)$

disjoint(A,B) $\Leftrightarrow \neg$ non-disjoint(A,B)

High-order formulas

One may define properties of relations, then applies to specific relations.

- transitive(R) $\Leftrightarrow \forall X\ \forall Y\ \forall Z\ (R(X,Y) \wedge R(Y,Z) \Rightarrow R(X,Z))$
- transitive($\subset$)
- transitive($\leq$)

For Datte, it is only a pseudo--high-order

- transitive(R) $\Leftrightarrow \forall X \forall Y \forall Z\ (\ .\ R\ X\ Y \wedge .\ R\ Y\ Z) \Rightarrow .\ R\ X\ Z))$

For Muscadet1, it will be really high-order

Results

Datte has proved about 150 theorems in set theory, mappings, orderings, congruence relations, ordinal numbers.

Some of them are difficult and were not proved by other provers until a few years ago.

A knowledge-based prover using knowledge and metaknowledge
(Muscadet1)

The facts are the internal representation of the theorem during the proof

- Facts
  - conclusion to be proved
  - hypotheses
  - objects
  - subtheorems
  - active rules
  - …
  - all sort of facts which give relevant information during the proof searching progress

- Rules
  - logic and mathematics
  - built from definitions and axioms
  - dynamically built from hypotheses
- Metarules

- definitions, axioms, lemmas

- inference engine (PL1 then Pascal)

All strategies which were programmed in Datte are written by rules and metarules

+
- readability, modularity
- second order, variable arity
- writing knowlege bases, especially know-how is easier

-
Some specific efficient representations loose their efficiency (Ex : graphes de Datte et de Merialdo)

the quantifier !
"for the only ... such that ..."

Rule : if the expression P(F(A)) occurs where F is a functional symbol
then replace it by !B:f(A), P(B)
where !B:f(A), P(B) means for the only B equal to f(A), p(B) is true

!B:f(A), P(B) is equivalent to $\forall B(f(A):B \Rightarrow p(B))$
and to $\exists B\,[f(A):B \wedge P(B)]$

The first expression is better for conclusions (positive position),
to prove !B:f(A), P(B),
create B1, add the hypothesis B1:f(A) and prove P(B1)

The second one is better for hypotheses (negative position), no such hypothesis is added, at the place we have the *super-action*
to add !B:f(A), P(B)
create an objet B1and add the hypothesis P(B1)

Applications to difficult domains

- Topological Linear Spaces
- Kuratowski theorem
- Cellular Automata
- concepts
- splittings
- formalisation symbolic manipulation
- Auto-observations
- Observation of mathematicians thinking aloud while solving some problems

Topological Linear Spaces

ThA. If U is a neighbourhood of the origin O in a TLS (Topological Linear Space), then there exists a neighbourhood V of O, included in U, star-shaped and symmetric relative to O

ThB. If U is a neighbourhood of the origin O in a TLS, then there exists a neighbourhood V of O, included in U, star-shaped and symmetric relative to O, such that $x \in V$, $y \in V$ implies $[x,y] \subset U$.

Knowledge bases

definition of a TLS
- linear space, operations + and ×
- topological space : + and × are continuous

properties of + and ×, topological space (continuity, open sets and/or neighborhoods)

know-how knowledge
- bases of neighborhoods
- product sets
- usual bases of neighborhoods for elements (x,x) of the diagonal
- symbolic manipulation of + and ×
- reasons to choose to first apply the continuity of + or ×.

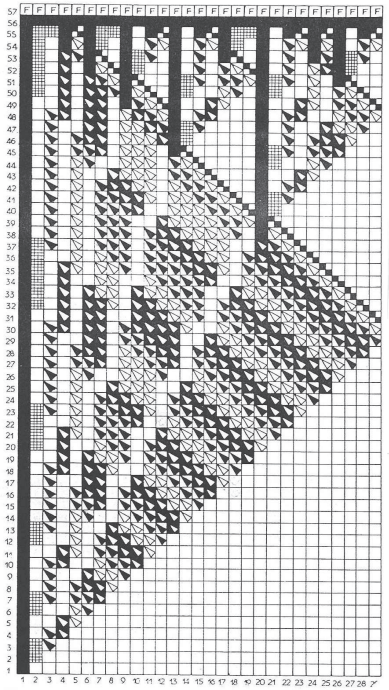
specific but general and used by mathematicians

Kuratowski theorem $(a,b)=(c,d) \Rightarrow a=c \wedge c=d$

Definitions : ordered pair : $(a,b)=\{\{a\},\{a,b\}\}$,
singleton : $\{a\}=\{x \mid x=a\}$
unordered pair : $\{a,b\}=\{x \mid x=a \vee x=b\}$

"The definition of (a,b) does not have any intrinsic intuitive meaning. It is just a convenient way (discovered by Kuratowski) to define ordered pairs so than one can prove the characteristic property of ordered pairs expressed in the proposition". (Mendelson)

Proofs by
- Mendelson 4 splittings, 6 subtheorems
- Resolution
- Brown (new method, 1996), very long
- Muscadet 8 splittings, 13 subtheorems
- Muscadet + know-how knowledge 1 splitting, 2 subtheorems



Cellular Automata

Firing squad synchronization problem

n machine,
t=0 all machine in quiescent state L
find a set of states and transition rules to fire in minimal time (2n-1)

A solution with 6 states has been found .

1- build the automata
2- prove that it is 6-state minima time solution
(needs to prove 11 difficult lemmas to verify the theorem)

Notations

If $7 \le m < n$, we let DD_m be the set $\{(m-i,m+i), (m-i,m+i+1) \mid i \in \{-1,0,...,m-1\}\}$

If $7 \le m < n$ and $1 \le k < j \le m+1$, we let $DD_{m,j,k}$ be the set

$DD_m \cap \{(l,t) \mid k \le l \le j\} = \{(m-i,m+i), (m-i,m+i+1) \mid i \in \{m-j,0,...,m-1\}\}\}$

Definitions

Suppose $j \ge k$ and $k \ge 1$; we say that $DD_{m,j,k}$ is *basic* in state x if

all sites (i, 2m-i),(i,2m-i+1) with $i \in \{m-j+1,...,m-k-1\}$ have state x

A first easy result to be proved (fact)

Every site(K,t) with K>t has state L

A simplification of the first lemma to be proved (among 11)

Let 1<m, 1<k, j≤k and j≤m+1. If $DD_{m,j,k}$ is basic in A and if site (k-1,2m-k+1) has state A,then the first diagonal of $DD_{m+1,j,k}$ is also basic in A

Difficulties

- two order relations (large and strict)
- $x \le y \Leftrightarrow x < y \vee x = y$ leads to many splittings
- too much time on them, instead of iportant properties (state and site)
- symbolic manipulation

To help Muscadet

- introduce the concept of successor suc(x) for x+1
- and the following properties

  $x < y \Rightarrow suc(x) \le y$

  $x < y \Rightarrow x \le y$
- introduce a special function f(m,i) for 2m-i
- and its following properties

  f(m,i+1)= f(m,i)-1

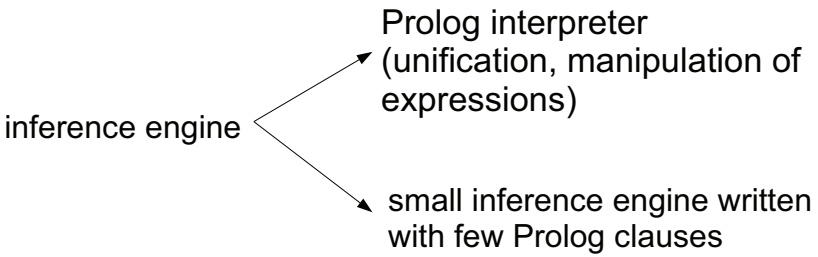
  f(m+1,i)= f(m,i)+1+1

Needs to combine theorem proving and symbolic manipulation

A Prolog prover using declarative as well as procedural knowledge (Muscadet2)

Muscadet has been rewritten in Prolog

- know-how knowkedge easier to write
- expression of procedural strategies
- more flexibility (declarative/procedural/mixing both)
- more readability



syntax for formulas

- Prolog conventions are used for variables and constants

- formulas are writen in infix and partially parenthesized notations, almost as matheticians do.
- Examples :

  A subset B <=> forall(X, X in A => X in B)
- and also

  A inter B = [X, X in A and X in B]

  forall(X in A, exists(Y in B, p(X, Y, A, B)))

  forall(X in real, forall(Y in real, forall(Z in real,

  X < Y and Y < Z => X <Z)))

rules and actions

```
rule(N,=>) :- concl(N , A => B), addhyp(N, A), newconcl(N,B)
rule(N, forall) :- concl(N,forall(X,C)), create_object(x,X1)
                   addobj(N, X1), replace(C, X, X1, C1),
                   newconcl(N, C1).
newconcl(N, C) :- retractall(concl(N, _)), assert(concl(N, C))
ajhyp(N, H) :-
       ( H = A and B -> addhyp(N, A), addhyp(N,B)
       ; H = (X = X) -> true
       ...
       ; H = forall(_, _) ->  buildrule(H,N)
       ...
       ; assert(hyp(N, H))
       )
```

these examples have been slightly simplified for readability

Functional symbols

cannot be variables in Prolog
pseudo-second-order as in Datte

..[R,X,Y] means R(X,Y)
the symbol ".." and the list notation were chosen by analogy with the Prolog operator "=.." since in Prolog we have
r(a,b) =.. [r,a,b]
and E =.. [r,a,b] gives E=r(a,b) :-)
r(a,b) and ..[r,a,b] cannot be unified but ..[r,X,Y] is always replaced by r(X,Y) if R is instanciated by r in ..[R,X,Y]

rules and actions (continued)

```
addhyp(N, H) :-
       (
       ...
       ; H = ..[R, X, Y]  -> H1 =..[R, X, Y], addhyp(N, H1)
       ...
        )
```


procedural rules

$\wedge$ [resp. $\vee$] cannot have an arity > 2
$A \wedge B \wedge C \wedge D \wedge E$ is in fact $A \wedge (B \wedge (C \wedge (D \wedge E)))$

Scanning all elements of a conjunction or a disjunction is now done by a specific Prolog predicate which returns successively all the elements of the conjunction [resp. disjunction].
This leads to a gain of time

Each time that a developper discovers that a task takes more time that it should do, he/she may define a Prolog predicate to speed up.

A prover knowing nothing about mathematics apart logics, application to TPTP problems (Muscadet3)

In particular, this new version does not know
- ∈
- keywords "definition" and "lemma"

Modifications
- statements which are definitions have to be recognized
- methods using ∈ has been rewritten in a more general manner (for example, methods about handling functional symbols)

Thousands of Problems for Thorem Provers (TPTP)

since 1993
FOF since 1997 : 217 FOF problems (5 SET) / 3330
nows ~8000 FOF (~1400 SET-SEU / ~20000

I proposed 120 problems in 1999

CADE ATP System Competition (CASC)

Since 1999, Muscadet participated and proved some theorems that no other prover could prove.
This shows its complementarity with other provers.

A TPTP prover (Muscadet3)

TPTP syntax
- input and output formulas
- internal representations
- results display and trace

Relevant proofs (Muscadet4)

Extraction of relevant steps
- difficult because of the calls of procedural actions in the rules (in particular handling subtheorems and return)

many modifications of the program
- step numbers as new parameters, in facts and rules
  - the number of the current step in the actions
  - the preceding step numbers of each fact in the conditions
- add an explanation for each step
- these parameters have to be passed in many predicates
- memorisation of all steps with these informations
- extraction of the path from the success step up to the first step

Utilities and tools (Muscadet4)

- The prover may be simply called by executable C program, with the address or the name of a thorem to be proved (+ options if needed).
- It may also be called under Prolog. In this case it is possible to display all the facts, and also all the memorized steps and examine them, with the help of Prolog commands.
- There are also some shell scripts to analyse the proofs.


```
cc tptp-en.c -o tptp
tptp SET027+4.p

cat res_SET027+4.p

*****************************************************************************
theorem to be proved
![A,B,C]: (subset(A,B)&subset(B,C)=>subset(A,C))
-----------------------------------------------------------------------------
theorem 0 proved (conjecture:thl03) in 0.009 seconds
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
in the following, N is the number of a (sub)theorem
      E is the current step
      or the step when a hypothesis or conclusion has been added or modified
hyp(N,H,E) means that H is an hypothesis of (sub)theorem N
concl(N,C,E) means that C is the conclusion of (sub)theorem N
obj_ct(N,C) means that C is a created object or a given constant
addhyp(N,H,E) means add H as a new hypothesis for N
newconcl(N,C,E) means that the new conclusion of N is C
      (C replaces the precedent conclusion)
a subtheorem N-i or N+i is a subtheorem of the (sub)theorem N
      N is proved if all N-i have been proved (&-node)
          or if one N+i have been proved (|-node)
the initial theorem is numbered 0
```



```
* * * proof

* * * theorem to be proved
![A,B,C]: (subset(A,B)&subset(B,C)=>subset(A,C))

* * * * * * theoreme 0 * * * * * *
*** newconcl(0,![A,B,C]: (subset(A,B)&subset(B,C)=>subset(A,C)),1)
*** explanation : initial theorem
------------------------------------------------------------ action ini
create object(s) z3 z2 z1
*** newconcl(0,subset(z1,z2)&subset(z2,z3)=>subset(z1,z3),2)
*** because concl((0,![A,B,C]: (subset(A,B)&subset(B,C)=>subset(A,C))),1)
*** explanation : the universal variable(s) of the conclusion is(are) instantiated
------------------------------------------------------------ rule !
*** addhyp(0,subset(z1,z2),3)
*** addhyp(0,subset(z2,z3),3)
*** newconcl(0,subset(z1,z3),3)
*** because concl(0,subset(z1,z2)&subset(z2,z3)=>subset(z1,z3),2)
*** explanation : to prove H=>C, assume H and prove C
------------------------------------------------------------ rule =>
*** newconcl(0,![A]: (member(A,z1)=>member(A,z3)),4)
*** because concl(0,subset(z1,z3),3)
*** explanation : the conclusion  subset(z1,z3) is replaced by its definition(fof axiom:subset )
------------------------------------------------------------ rule def_concl_pred
```



```
create object(s) z4
*** newconcl(0,member(z4,z1)=>member(z4,z3),5)
*** because cocncl((0,![A]: (member(A,z1)=>member(A,z3))),4)
*** explanation : the universal variable(s) of the conclusion is(are) instantiated
------------------------------------------------------------ rule !*** addhyp(0,member(z4,z1),6)
*** newconcl(0,member(z4,z3),6)
*** because concl(0,member(z4,z1)=>member(z4,z3),5)
*** explanation : to prove H=>C, assume H and prove C
------------------------------------------------------------ rule =>
*** addhyp(0,member(z4,z2),7)
*** because hyp(0,subset(z1,z2),3),hyp(0,member(z4,z1),6),obj_ct(0,z4)
*** explanation : rule if (hyp(A,subset(B,D),_),hyp(A,member(C,B),_),obj_ct(A,C))then
addhyp(A,member(C,D),_)
built from the definition of subset (fof axiom:subset )
------------------------------------------------------------ rule subset
*** addhyp(0,member(z4,z3),8)
*** because hyp(0,subset(z2,z3),3),hyp(0,member(z4,z2),7),obj_ct(0,z4)
*** explanation : rule if (hyp(A,subset(B,D),_),hyp(A,member(C,B),_),obj_ct(A,C))then
addhyp(A,member(C,D),_)
built from the definition of subset (fof axiom:subset )
------------------------------------------------------------ rule subset
*** newconcl(0,true,9)
*** because hyp(0,member(z4,z3),8),concl(0,member(z4,z3),6)
*** explanation : the conclusion member(z4,z3) to be proved is a hypothesis
------------------------------------------------------------ rule stop_hyp_concl
then the initial theorem is proved
* * * * * * * * * * * * * * * * * * * * * * * *
```

Why Muscadet is an efficient system in a number of circumstances

- the fact that the growth of the bases of facts in linear, not exponential ;
- the importance and efficiency of the chosen representations of the problem ;
- the splitting of a (sub)theorem in many subtheorems easier to prove, independent or not ;
- the treatment of functional symbols which flattens the handled expressions ;
- the replacement of definitions and universal hypotheses by natural and efficient rules ;
- the treatment of equalities and negations which removes them as far as possible.

Muscadet is

efficient for

- every day mathematical problems wich are expressed in a natural manne
- problems which involve many definitions, axioms or lemmas

not efficient for

- problems defined axiomatically, from a logician's point of view
- problems which involve only one large conjecture and no intermediary definitions

And now ?

Nowadays, Muscadet is no longer being actively improved, except the interface to let it more and more easy to use.

Nevertheless, the analyses of failures (obtained during CASC or by users) often lead to the correction of bugs or to some improvements of some rules ou metarules.

Conclusion

For all these reasons,

how theorem proving with Muscadet could progress ?

- to some extend by the improvements of its heuristics
- certainly not by the increase of computer power
- but surely by having it cooperate with other provers