Can and Must a Machine Prove Theorems as Humans Do ?
Analysis of Some Automated Proofs by the MUSCADET System

DOMINIQUE PASTRE
Université René Descartes
CRIP5
UFR de Mathématiques et Informatique,
45 rue des Saints Pères, 75270 PARIS CEDEX 06,
e-mail: pastre@math-info.univ-paris5.fr

Abstract. In this paper, we study the theorem proving activity by Humans and by automated provers. We discuss the necessity and the possibility for a machine to reason as human beings do. This may be useful in the case of computer-human interaction, either because the automated prover has to be incorporated into a tutorial system, or is intended to become an assistant, or needs to be helped by a human user.

The MUSCADET system is able to automatically find proofs and may be considered as a model of theorem proving activity. It may also help human discovery in mathematics by automatically proving intermediary results.

We analyse some automated proofs by the MUSCADET system and give some higher-level proofs of the same theorems. We discuss the use of declarative or procedural knowledge and compare the behaviour of MUSCADET and of human researchers.

1. Introduction

Automated Theorem Proving has been studied since the beginning of Artificial Intelligence and many automated provers have been written. Most of them reason in a manner quite different from human reasoning; others try to reason as Humans do and their proofs may be understood by Humans. Moreover, the manner in which the statement of the theorem to be proved is given to an automated prover or to a human being is often not the same. Thus, interaction with Humans is very difficult. But this human interaction is necessary in several cases, either because the automated prover has to be incorporated into a tutorial system, or is intended to function as an assistant, or needs to be helped by a human user.

In this paper, we analyse what is generally considered as a theorem-proving activity (section 2), and, according to the aims of Automated Theorem Proving (section 3), we examine in what circumstances a machine has to reason like Humans (section 4). Different levels of proofs are compared in sections 5 and 6. In section 6, we also give examples of automated proofs by the MUSCADET system which is a knowledge based theorem prover and may be considered as a good model of theorem proving activity although some mathematicians sometimes give finer proofs. It is described in detail in Pastre (1989), and more succinctly in Pastre (1993). In section 7, declarative and procedural knowledge are analysed, and we compare the behaviour of MUSCADET and the behaviour of researchers who were observed while trying to find some proofs.

2. What is meant by "to solve a problem" or "to prove a theorem" ?

In a problem-solving activity we can distinguish three stages. The first stage is to understand the problem and to express it in adequate language or representation, the second stage is to find a solution and the third stage is to give the solution to the problem. Theorem proving is a special case of problem solving. The first stage is to read the statement of the theorem and to express it in an adequate manner, the second one is to find the proof, the last one is to give the proof or perhaps just to answer if the statement is a theorem or not.

2.1. THE STATEMENT OF THE THEOREM TO BE PROVED

The statement of a theorem to be proved may be given in natural language, for example:

Prove that the union of two subsets of a set is also a subset of this set.

It may also be given in formalized mathematical language. Usually first-order or higher-order predicate calculus is used and the metamathematics are expressed in natural language, for example

$$Prove\ that\ A \subset C \wedge B \subset C \Rightarrow A \cup B \subset C$$

but metamathematics may also be formalized.

The definitions may be implicit or may also be given either in natural language or in formalized mathematical language:

A set is a subset of another set if every element of the first set is an element of the second one.

or $A \subset B \Leftrightarrow \forall x(x \in A \Rightarrow x \in B)$

The union of two sets is the set of elements which belongs to one or the other or both.

or $A \cup B = \{x \mid x \in A \vee x \in B\}$

For many automated theorem provers, the language used to express the theorem to be proved is the clause language, that is, for the above example

subset : $\neg$ *subset (A, B)* $\vee \neg$ *element (x, A)* $\vee$ *element (x, B)*

subset (A, B) $\vee$ *element (f(A, B), A)*

$\neg$ *element (f(A, B), B)* $\vee$ *subset (A, B)*

where *f* is a Skolem function.

union: $\neg$ *element(x,union(A,B))* $\vee$ *element(x,A)* $\vee$ *element(x,B)*

$\neg$ *element (x, A)* $\vee$ *element (x, union (A, B))*

$\neg$ *element (x, B)* $\vee$ *element (x, union (A, B))*

negation of the theorem to be proved: *subset* (A_O, C_O)

subset (B_O, C_O)

$\neg$ *subset (union* (A_O, B_O), C_O)

where A_O, B_O, C_O are Skolem constants.

It is implicit that the prover has to derive the empty clause.

We cannot say that the prover which receives this set of clauses solves the same problem as the student who is asked to prove the theorem stated in natural language. There is a difference between the problem stated in natural language and the problem stated in first-order predicate calculus, but usually problems for students are partially stated in this form, perhaps because natural language may be ambiguous, and only a formalized language is precise enough to avoid misunderstanding.

This phenomenon is more crucial in physics. As it is analysed in Tisseau (1992) the greatest difficulty for students solving problems or exercises of physics stated in natural language is to choose the physical systems and to find the right physical laws and equations to

express them. When the formalisation is over, the problem is almost solved. In the case of the MODÉLIS system (Tisseau (1992)) , which solves problems of thermodynamics stated in French, the most difficult task is to formalize the problem.

The difficulty of many problems in general-problem solving depends on their formalisation and on the chosen representations. So, I think that it is important, when describing the performances of a problem solver or of a prover, to describe exactly what problem it solved.

2.2. THE ANSWER OF THE PROVER

The prover, human or machine, may only give the answer "yes, the statement is a theorem", or "no, the statement is not a theorem", or "I did not succeed in proving that the statement is a theorem". The prover may also give the proof of the theorem. All intermediary answers are possible between a brief proof with only the main steps in natural and perhaps informal language (the "incomplete proof" in Polya (1957)), and a detailed proof giving all the steps in a more or less formalized language (the "Euclidian exposition" in Polya (1957)) and telling how these steps were found.

2.3. THE PROOF

An automated proof may ressemble a human proof or, on the contrary, be quite different. Provers based on the principle of resolution (Fujita & al. (1993), Kerber (1992), Mc Cune (1994), Präcklein (1992), Robinson (1965), Wos (1988)) give proofs which are difficult to understand for Humans, except for very easy and short ones. Provers based on Natural Deduction or other natural methods (Ammon (1988), Bledsoe (1971), Bledsoe (1977), Brown (1978), Pastre (1978), Pastre (1989), Zhang & al. (1995)) give proofs which are closer to human proofs. We shall examine to what extent we can say that they are humanlike. Some other provers (Boyer and Moore (1979), Brown (1986)) give proofs which are easily understandable by a user accustomed to their principles and which are understandable with a little effort by an unaccustomed observer. Both approaches may be combined, such as in Matsuyama & al. (1995) where algebraic reasoning (unreadable but efficient except for order relations between points or relations) and logical reasoning (manipulation of these order relations) cooperate.

If the human user wants to obtain a more or less detailed proof and not only to know if the statement is a theorem, it is better if the proof ressembles a human proof, and, better still, if it is a beautiful one.

3. Why is the point of proving theorems automatically ?

The necessity for an automated prover to work as Humans do depends on the aim of the prover. So, we shall first examine why automated provers are built. The first reason is because it is a challenge for Artificial Intelligence: mathematical activity is considered as an activity which needs intelligence. The second reason is that theorem provers may solve problems in other domains apart from mathematics. Thirdly, theorem provers may be written with a tutorial objective: it is now admitted that a tutorial system must be able to solve the problems it proposes to the students and not only to know the solutions given by a teacher. Lastly, theorem provers may be written to assist mathematicians (Huang & al. (1994)), Pastre (1994)). The machine may automatically find the solution of a problem or automatize the proofs of subproblems in order to help the discovery of the solution of a difficult problem. Some new theorems have been proved by or with the help of automated provers, in domains which are new to mathematicians or which require large combinatory research, such as combinatory logic, equivalential calculus, finite semigroups, and finite algebra, (Fujita & al. (1993), Wos & McCune (1992)). Proofs given by OTTER (McCune (1994)) in equational logic and algebraic geometry are the result of the cooperation of a mathematician and a computer scientist (Mc Cune & Padmanabhan (1996))

4. Must a machine prove theorems as Humans do ?

According to the different aims of automated theorem proving which have just been just mentioned, we shall examine when and where it is useful for machines to behave like Humans and in what circumstances.

4.1. WHAT ABOUT STATEMENTS ?

To successfully conduct an intelligent task, it has long been considered that the manner in which statements are given to a problem solver or a theorem prover is of no importance; the interest and the difficulty are in the proof. This was justified at the beginning of Artificial Intelligence because the first difficulty to overcome was that of automatizing the proofs. Now many problem solvers and theorem provers are capable of finding proofs, and we must not forget that translating a problem with an adequate representation in a suitable language is already a part of the solution (many examples are given in Laurière (1986) and Winston (1984)). Moreover, choosing a definition of a mathematical concept among several equivalent definitions points the search in a particular direction. The choice of predicates

may also be very important and modify the difficulty of a proof. The writing of the clauses is already a non-trivial problem because the prenex and skolemized forms are not unique; moreover, it is a long and tedious activity and it is desirable that it should be automatized.

In order to be used in a tutorial environment, a prover must also be able to understand statements in natural language (eventually restricted) or in a formalized language which has been chosen because it is useful for the student and not because it is imposed by the machine.

In order to function as an assistant for mathematicians, a prover must understand the statements as mathematicians accept to give them, that is, usually, in more or less formalized mathematical language. It must accept common misuse of language (for example using *f(x)* to denote the element image of the element *x*, and *f(A)* to denote the set image of the set *A*), ambiguity, abbreviations, implicit information (for instance saying *f is continuous* without indicating on what set if it is obvious).

4.2. WHAT ABOUT THE ANSWER ?

If it is claimed that the machine proved a difficult theorem, the community must be convinced that the machine effectively proved the theorem either by giving the main steps of a proof understandable by an observer, or by proving that the prover is correct. Proving that the prover is correct may convince computer scientists, but it may not convince all mathematicians. For example, Fujita et al. (1993) automatically proved a new theorem in finite algebra, but the proof, based on the study of very many cases, is not humanly verified. The user must be confident in the prover and the machine. The same goes for the proofs obtained by Wu's method (Chou (1988)) in analytic geometry which requires intermediary polynoms of more than 14000 terms. The same was also true for the Four colour problem, the proof of which was obtained with the aid of computers and which required the study of many cases; about 10 000 diagrams were used as illustrations ! And are all mathematicians convinced that the theorem was really proved ?

If a mathematician uses an automated theorem prover as an assistant and only wants to verify a minor point, a black box which answers yes or no is sufficient, but if he wants to study a delicate point, it is necessary that the prover explain itself by giving important reasons and not tedious argumentation.

The same goes if the prover is used in a tutorial environment, except that the explanations are usually needed in more detail; moreover, if it is necessary to introduce an intermediary object (for example a point, an element, a set, a mapping, a lemma), it is not sufficient to say "let x be …"; the machine must also say "introduce x … because …"

Moreover, in an assistance or tutorial environment, if a certain succession of steps often arises, it would be better for the machine to say "this step is performed like the preceding one" instead of repeating the same sequence of steps.

4.3. WHAT ABOUT THE PROOF ?

If the aim is to prove a theorem, the machine does not have to prove it as Humans do, except for two reasons. First, good heuristics may be found by observing human behaviour and imitating human heuristics if possible. Secondly, if the prover has to convince the observer, it must be able to extract the important points of the proofs and to translate their succession in a form understandable by him. It is easier if the proof already ressembles a human one. It is much more difficult to extract these points from a resolution-based proof for example.

The preceding remark only concerns the main steps of the proof. Intermediary results which are obvious for the observer may be proved by any method. An extreme example is a numerical computation: the machine does not have to add 356786 and 89345 as Humans do, but it computes with its central unit and gives the result. The same may be true for minor points of the proof. All manipulations which do not need to be explained may be done by methods which are specific to the machine. Moreover, all intermediary results which are difficult to prove but easy to verify when they are given may also be found by methods specific to the machine. This is the case in the factorisation of polynoms or in the calculus of integrals. When the result is given, it is easy to develop the factors or to derive the function, and so congratulate the machine which was able to find it. Many systems are now able to do these symbolic manipulations and may be used as a black box.

If the prover is an assistant, the principle is the same, but if the mathematician asks for more details, the machine must also be able to explain the intermediary points that it first chose not to explain. So, if it is certainly never necessary to explain how the sum of 356786 and 89345 was obtained, it may or may not be necessary to explain that a=c was deduced because it is known that a=b and b=c and the relation = is transitive.

The same goes for a tutorial environment, and, here, much more flexibility and adaptation to the student level is required. In certain cases, for some students, all detailed steps will have to be explained.

5. Use of lemmas and of high-level knowledge

Very often, experts use lemmas when proving theorems. This produces simpler and finer proofs. These lemmas may be well-known results, for example the transitivity of the subset

relation. They may also be intermediary results that the mathematician prefers to prove (separately) in a general manner and then apply in the particular case encountered during the proof. The advantage is that the difficulties are cleary separated, as are the properties which are useful either for the theorem to be proved or for the lemma. No automated prover is yet capable of explicitly doing such difficult splitting.

The possible use as lemmas of known theorems frequently occurs in theorem proving. Provers which are rather proof checkers such as Boyer and Moore (1979) receive many such lemmas and prove them one after the other right up to the final theorem. Most theorem provers may receive some lemmas. Sometimes it is necessary because, although they are able to prove the lemmas and are able, knowing the lemmas, to prove the theorem, they do not actually think of using the lemmas. On other occasions, the use of lemmas is not indispensable but they enable us to have simpler and finer proofs. If the user gives a prover precisely those lemmas which are useful, the proof is simpler and it is also obtained faster. But if the prover receives all known theorems of a theory, or keeps as possible lemmas all the theorems it proved, the task is more difficult. Laublet (1993) carried out such experiments with his system. The proof may not be as simple as it could be, but it may be simpler than the proof without the lemmas. The required time may be as great as in a the proof without the lemmas. So, it is useful for the aesthetic quality of proofs, but not for time efficiency.

Another frequent use of high-level reasoning consists in noticing that a part of a proof ressembles a preceding part which was obtained. This is the case, for example, if the conclusion to be proved is $A \cup B \subset C$, as in the example mentioned in section 2.1 (but we could have another situation and the proof could be more difficult). Let x be an element of $A \cup B$, x belongs either to A or B. The first case consists in assuming x in A and proving that x belongs to C. MUSCADET, for example, proves this without difficulty. Then, a human being will say that the other case, x in B, is treated in the same manner. For a prover, it is much easier to repeat the subproof than to notice the similarity. This is the case also in the example analysed in the next section.

6. Examples of proofs

We shall illustrate the preceding points with two theorems which were proved by MUSCADET

6.1. A PROOF BY MUSCADET

The English statement of the theorem is the following:

The union of the images of two sets is included in the image of the union of the two sets

The formalized statement is the following:

$$f(A) \cup f(B) \subset f(A \cup B)$$

It may be implicit or specified that f is a mapping and that A and B are subsets of the domain of f.

The latest version of MUSCADET receives it in the following manner:

```
mapping (f,x,y)    and    a subset_of x    and    b subset_of x
                 implies
image(f,a)  union  image(f,b)     subset_of     image(f, a union b)
```

The precedences of operators enable us to use infix notations and to avoid some parentheses.

We must specify that f is a mapping and use the function *image* because MUSCADET is not (yet) able to understand *f(a)* where *a* is a set, because this usual human notation is ambiguous. Except for these cases and the fact that the machine does not accept symbols like $\approx$ or $\wp$, it is the same statement as the preceding mathematical one.

To prove this theorem, MUSCADET first creates and names the sets

```
image(f,a)
image (f,b)
image(f,a) union image(f,b)
a union b
image(f, a union b).
```

It replaces the conclusion

```
image(f,a)  union  image(f,b)     subset_of     image(f, a union b)
```

by its definition

```
forall(X,   X   element   image(f,a)  union  image(f,b))
            implies      X   element   image(f, a union b)
```

then it creates `x1` in `image(f,a) union image(f,b)` and the new conclusion is

```
x1 element   image(f, a union b)
```

From the hypothesis

```
x1 element   image(f,a) union image(f,b)
```

and rules that were built by metarules from definitions (see Pastre (1989) for the description of this mechanism, applied here to the `union`), it adds the disjunctive hypothesis

```
x1 element  image(f,a)   or   x1 element  image(f,b)
```

Because of this disjunctive hypothesis, the theorem to be proved is split into two subtheorems. Note that disjunctive hypotheses are not systematically treated, because if the splitting is done too early, the system could have to prove the same subtheorem several times (see Pastre (1989)).

The first subtheorem has the hypothesis

```
x1 element  image(f,a) .
```

From rules built from the definition of `image`, MUSCADET adds the existential hypothesis

```
exists (X element a ,  x1 = f(X))
```

then decides to treat it, i.e. to create a new object

```
x2 in a   such that   x1 = f(x2).
```

Note that , as disjunctive hypotheses, existential hypotheses are not systematically treated, because the creation of new objects may be expansive (see Pastre (1989)). (A method is said to be expansive if, each time it is applied, new conditions arise to apply it again.)
From the hypothesis

```
x2 element a
```

and

```
x1 = f(x2).
```

rules built from the definitions of `union` and `image`, add the new hypotheses

```
x2  element  a union b
```

then

```
x1 element  image(f, a union b)
```

which is the conclusion to be proved.

The second subtheorem has the hypothesis `x1 element image(f,b)` MUSCADET then repeats the same sequence of steps with `b` in the place of `a`. It is not (yet) able to notice the symmetry between `a` and `b`. No mathematician would repeat it, and even a student who is not very advanced would say that it is the same if `x1 element image(f,b).`

So, the proof by MUSCADET and the trace are very similar to a human proof, albeit a low-level one, and a human being would produce a higher-level proof by noticing the symmetry.

6.2. A HIGHER LEVEL AND FINER PROOF

Some mathematicians think that the preceding proof, even with the remark on symmetry is a beginner's proof. An expert produces a higher-level proof by using the following lemmas:
Lemma 1 : $X \subset X \cup Y$
Lemma 2 : *if* $X \subset Z$ *and* $Y \subset Z$ *then* $X \cup Y \subset Z$
Lemma 3 : *if* $X \subset Y$ *then* $f(X) \subset f(Y)$

The proof is the following:

$A \subset f(A \cup B)$ by lemma 1
$f(A) \subset f(A \cup B)$ by lemma 3
$f(B) \subset f(A \cup B)$ by a symmetry argumentation
$f(A) \cup f(B) \subset f(A \cup B)$ by lemma 2

It is simpler and finer that the preceding proof.

MUSCADET is able to produce such a proof if it receives these three lemmas (which it is also able to prove). But, if it also receives as lemmas, all well-known theorems about set theory, it will produce useless steps. Moreover, a mathematician may not be conscious of some lemmas when he begins to search for the proof, but he will remember or rediscover them wherever necessary.

6.3. ANOTHER PROOF BY MUSCADET

Consider three mappings $f: A \to B$, $g: B \to C$, $h: C \to A$. If any two of the three mappings $h \circ g \circ f$, $f \circ h \circ g$, $g \circ f \circ h$ are injective (resp. surjective) and the third is surjective (resp. injective), then f, g and h are one-to-one.

This theorem is not conceptually difficult but rather complicated. MUSCADET succeeds in proving this theorem because of its ability to treat existential properties. It does not create an element each time an existential property appears because these creations may be expansive, but only when it decides to treat an existential hypothesis, and only one at a time, then other rules are tried again before another existential hypothesis is treated.

MUSCADET considers many elements in *A, B, C* which are images or pre-images of preceding ones. It has to avoid creating too many elements, and it also has to consider at the same time the 3 sets and the 6 mappings. The theorem is split into 6 subtheorems all of which have to be proved because there is no symmetry. Each of them is split into 2 subtheorems (injectivity and surjectivity). The easiest of these 12 subtheorems requires the creation of 4 elements. The most difficult subtheorem requires 9 elements and MUSCADET also creates 14 other useless elements.

6.4. A HIGHER LEVEL PROOF

These proofs are easy to read for an observer and could be produced by a human being. A mathematician remarked to me that this theorem was not so complicated because we could apply the following lemmas:
Lemma 1 : *If $g \circ f$ is injective then f is injective.*
Lemma 2 : *If $g \circ f$ is surjective then g is surjective.*

If $h \circ g \circ f$ and $f \circ h \circ g$ are injective and $g \circ f \circ h$ is surjective, we can immediately conclude that f, g, $g \circ f$ and $h \circ g$ are injective and that g and $g \circ f$ are surjective..

Then we must also use the following lemmas:
Lemma 1bis :

If $g_{\circ}f$ is injective and f is surjective then g is injective.

Lemma 2bis :

If $g_{\circ}f$ is surjective and g is injective then f is surjective.

By Lemma 1bis, $h_{\circ}g$ injective and g surjective, we deduce that h is injective.

By Lemma 2bis, $g_{\circ}f$ surjective and g injective, we deduce that f is surjective.

Then, by Lemma 2bis, $g_{\circ}f_{\circ}h$ surjective and $g_{\circ}f$ is injective, we deduce that h is surjective.

This proof is indeed shorter and of a higher level than the preceding one, but I am not sure that it is simpler ! It would certainly not be produced by a beginner. Moreover, I think that, if Lemmas 1 and 2 are rather well-known theorems, Lemmas 1bis and 2bis are certainly not known by all mathematicians and notably those who do not work with them every day. They probably would rediscover them by searching for conditions necessary to deduce injectivity or surjectivity, as I did myself when I decided to use such lemmas.

This discovery could be a challenge for automated provers, and it would be a very difficult task to perform.

7. Declarative and procedural knowledge

7.1. RULES AND METARULES

One of the main characteristics of the MUSCADET system is that it is a knowledge-based system. Knowledge is expressed by rules and metarules that we tried to write in as declarative a manner as possible. In fact, very often rules express, at the same time, the declarative knowledge, and the condition and manner of applying this knowledge. So, it is not really declarative, but it is closer to human behaviour. Generally, such rules were written for difficult mathematical fields (see Pastre (1989)), after having observed mathematicians at work. Mathematicians do have such rules which are partially compiled in their mind. In particular, many rules come to their minds, not in their most general form, but adapted to a given situation. These rules are much more efficient than the more general ones, and only experts possess them.

There are other circumstances where Humans seem to apply compiled knowledge. Each time they have to solve a trivial subproblem, they do not explicitly explain the reasoning. The reasoning is rather unconscious and done by compiled procedures they have in mind. While observing them, I sometimes had to ask them to prove an intermediary result they claimed to be true.

7.2. PROCEDURAL METHODS

Procedural methods, especially if they are associated with specific representations, may be more efficient than the use of rules and the use of a unique formalism for everything as is the case in MUSCADET.

We shall give two examples of such methods.

7.2.1. Representation of Binary Relations

In DATTE (Pastre (1978)), a theorem prover which can be considered as a predecessor of MUSCADET, many methods were programmed and several specific representations could be used. In particular, binary relations were represented by a graph implemented as an array, and specific procedures were written to manipulate them efficiently. Theoretically, this added nothing to the capabilities of the prover, but there was a gain in CPU time.

This graph was first simulated in MUSCADET: all binary relations were separated from others hypotheses, and rules manipulated them. Then as the knowledge base increased, much more time was wasted in the interpretation of these rules than was gained by the method. So it was removed.

This shows that the interest of a method may reside as much in the representations as in the method itself.

7.2.2. The System of Merialdo

Mérialdo (1979) wrote a system which was capable of proving theorems in Topology. One of its characteristics was its capacity to assure very efficient representations and manipulations of sets. This was very useful because many sets occur in general Topology, and the inclusion relation is very important because of the definitions and properties of closures and interiors. The closure (resp. interior) of a set X is the smallest (resp. largest) set which is closed (resp. open) and includes (resp. is included in) X. These manipulations were so easily and so quickly performed that the system knew all inclusion relations between all sets it created. So it was able to prove the following theorem

The closure of the union of two sets is equal to the union of the closures

$$\overline{A \cup B} = \overline{A} \cup \overline{B}$$

without receiving any lemma.

To prove this theorem, MUSCADET has to receive the following lemmas :

$A \subset B \wedge B \subset C \Rightarrow A \subset C$

$A \subset A \cup B$

$A \subset C \wedge B \subset C \Rightarrow A \cup B \subset C$

MUSCADET is able to use these lemmas and also to prove them, but not to discover them. Merialdo's system is able to discover, not the general lemmas, but the right inclusions.

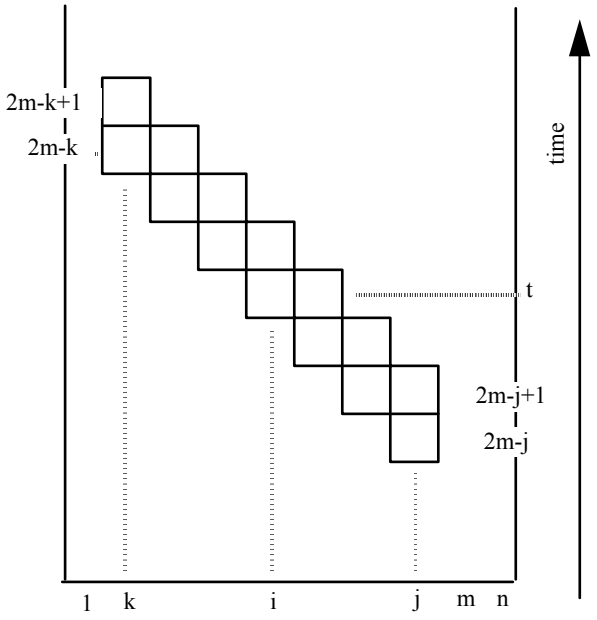
Humans also use these lemmas, often without indicating that they apply a lemma, because these results are quite well known and immediately applied where it is necessary.

The methods used in Merialdo's system, whose use depends on associated representation, are not only useful in order to gain time, but also to avoid giving lemmas.

7.3. USING BOTH DECLARATIVE KNOWLEDGE AND PROCEDURAL METHODS

MUSCADET is now used to prove theorems in discrete geometry and it is hoped that, in the future, it will become an assistant for researchers working on cellular automata. Some theorems are analysed, and proofs by human beings and by MUSCADET are described and commented on in Pastre (1993).

We have to handle surfaces which represent sets of sites (i,t), where *i* is the number of a machine and *t* is an integer for time, such as in the following example



which is formally defined by

$DD_{m,j,k} = \{(m-i, m+i), (m-i, m+i+1) \mid i \in \{m-j, 0, \ldots, m-k\}\}$.

Each site has a state which depends on the states of its antecedents (the same machine and its neighbours at the preceding time) in a way laid out in transition tables. We have to study the states of such kinds of surfaces.

MUSCADET only proved very simple theorems. Its behaviour is different from the behaviour of some researchers who were observed while working on these theorems. For Humans, visual perception is very important but they have difficulty formalizing and reasoning in a rigorous manner. In particular, induction is conducted only by informal reasoning. On the contrary, MUSCADET has no difficulty handling formal expressions and in particular determining if a site belongs to a $DD_{m,j,k}$, but it cannot easily perceive if a site is at the frontier of the surface, that is if one of its neighbours does not belong to this $DD_{m,j,k}$, a property that human beings immediately know without particularly thinking about it.

MUSCADET wastes a lot of time in manipulations of the integer numbers and their comparisons. All knowledge was first expressed in a rather declarative manner. Properties of the $DD_{m,j,k}$ (rules built from the definition) and properties of the order relations on N (such as transitivity and antisymmetry) were on the same level and treated in the same manner. Moreover, the following three relations $x \leq y$ (weak order), $x < y$ (strict order), $y = x+1$ (x is the immediate predecessor of y) are all important and are all kept although they are redundant. If $x<y$ and $y<z$, the transitivity of $<$ implies $x<z$, and it is also true that $x \leq z$ because the strict order implies the weak order. As we also have $x \leq y$ and $y \leq z$, the transitivity of $\leq$ also implies $x \leq z$. I tried to give rules for avoiding this last application, but applying these rules was as time consuming as applying the useless transitivity !

Another time-wasting factor is the following operation. Each time a new hypothesis is added, all rules are tried again. It is important that rules built from the definition of $DD_{m,j,k}$ be applied each time a new element is known to belong to such a $DD_{m,j,k}$, or each time a integer is known to be between k and j. But it is stupid to apply the transitivity of $\leq$ or $<$ if the hypothesis which has just been added has nothing to do with the order relation. Moreover, although a rule is never appplied with the instantiations for which it was already applied, the search is still carried out for all possible ordered pairs.

Humans obviously do not reason like this. They consciously think of the properties of the $DD_{m,j,k}$ but unconsciously apply the properties of the order relation only where it is necessary.

So, the treatment of the properties of the order relation was modified and made more procedural. The general rule expressing the transitivity

if $x \leq y$ *and* $y \leq z$ *then add* $x \leq z$

is removed. Instead, each time the system has to add a hypothesis $x \leq y$, it also applies the following rules:

$$\textit{if } y \leq z \textit{ then also add } x \leq z$$
$$\textit{if } y < z \textit{ then also add } x < z$$
$$\textit{if } t \leq x \textit{ then also add } t \leq y$$
$$\textit{if } t < x \textit{ then also add } t < y$$

A similar operation is used for adding hypotheses $x<y$ and $y = x+1$, and for the treatment of antisymmetry.

Much time is then gained, and this behaviour is closer to the behaviour of a human being who does not think of the transitivity, except in these circumstances.

Another difficulty concerns the splitting of $x \leq y$ into $x<y \vee x=y$. Some heuristics have been written to indicate where it has to be done. But, as it has often to be done at the frontier of the surface, it will be useful to find a representation of the surfaces which will enable MUSCADET to easily perceive these frontiers, as Humans do visually. The aim will not be to imitate Humans (visual perception is very difficult to automatize), but to find a representation which could have the same effect in this special case.

8. Conclusion

The necessity for a machine to prove theorems as Humans do depends on the aim set for the prover and the use laid out for it. It is necessary if it has to interact with Humans. This is the case if it has to explain its proof, especially in a tutorial or an assistance activity. This is also the case if it has to convince an observer who is not confident in the machine. And, lastly, this is the case if it has to receive help from a human user in an interactive manner or by receiving a specific mathematical-knowledge base.

The MUSCADET system is a knowledge based theorem prover and a good model of theorem proving activity. It proves easy theorems as Humans do. For more complicated theorems, it produces proofs which are understandable by human readers, but different from the proofs that an expert would give. One reason is that experts produce higher-level proofs by noticing properties such as symmetry and by using as lemmas some known theorems (and by often unconsciously selecting only the right ones). Moreover, even if they usually consciously explain the main points of a proof with the help of explicit and rather declarative knowledge, they solve many subproblems by procedural and compiled knowledge and adequate representations. It is perhaps necessary that automated provers should do the same in order to improve their performance and capabilities.

References

Ammon, K. (1988). Discovering a proof for the fixed point theorem: a case study, Proceedings of the 8th European Conference on Artificial Intelligence, 613-618

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

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

Boyer, R.S, Moore, J.S. (1979). A computational Logic, Academic Press, New York

Brown, F.M. (1978). Towards the automation of set theory and its logic, *Journal of Artificial Intelligence* 10, 281-316

Brown, F.M. (1986). An experimental logic based on the fundamental deduction principle, *Journal of Artificial Intelligence* 30 , 117-263

Chou, S.C.(1988). An introduction to Wu's method for mechanical theorem proving in geometry, *Journal of Automated Reasoning* 4 , 237-267

Fujita, M., Slaney, J., Bennet, F. (1993). Automated generation of some results in finite algebra, 13th IJCAI, 52-57

Huang, X., Kerber, M., Kohlhase, M., Melis,E., Nesmith, D., Richts, J., Siekmann, J. (1994). The ?-MKRP proof development environnement, ECAI 94 Wokshop "From theorem provers to mathematical assistants: issues and possible solutions

Kerber, M. (1992). On the representation of mathematical concepts and their translation in first-order logic, SEKI report SR-92-08, Universität des Saarlandes

Laublet, P. (1993). FORREnMat: un système à base de connaissances pour l'étude expérimentale du raisonnement mathématique, thèse de l'Université Paris 6

Laurière, J.L. (1986). Intelligence Artificielle, Résolution de problèmes par l'Homme et la machine, Eyrolles, Paris

Mc Cune, W.W. (1994). OTTER 3.0 Reference Manuel and Guide, Tech. Report ANL-94/6, Argonne National Laboratory,

Mc Cune, W.W. , Padmanabhan, R. (1996), Automated deduction in equational logic and cubic curves, Springer Verlag

Matsuyama, T. , Nitta, T. (1995). Geometric theorem proving by integrated logical and algebraic reasoning, *Journal of Artificial Intelligence* 75 , 93-113

Mérialdo, B. (1979). Représentation des ensembles en Démonstration Automatique, thèse de 3ème cycle, Paris 6

Pastre, D. (1978). Automatic Theorem Proving in Set Theory, *Journal of Artificial Intelligence* 10, 1-27

Pastre, D. (1989). MUSCADET: an automatic theorem proving system using knowledge and metaknowledge in mathematics, *Journal of Artificial Intelligence* 38, 257-318

Pastre, D. (1993). Automated Theorem Proving in Mathematics, *Annals of Mathematics and Artificial Intelligence,* Volume 8, No. 3-4 , 425-447

Pastre, D. (1994). Towards a man-machine cooperation in mathematics with the MUSCADET system, ECAI 94 Wokshop "From theorem provers to mathematical assistants: issues and possible solutions, 54-65

Polya, G. (1957). How to solve it ? Princeton University Press

Präcklein, A. (1992). The Markgraf Karl Refutation Procedure User Manuel, SEKI report SWP-92-03, Universität des Saarlandes

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

Tisseau, G. (1992). Modelis : an artificial intelligence system which models thermodynamics textbook problems, in *Intelligent Learning Environments and Knowledge Acquisition in Physics*, A.Tiberghien, H.Mandl (eds), Nato ASI series, series F : Computer and system sciences, vol. 86, Springer-Verlag

Winston, P.H. (1984). Artificial Intelligence, Addison-Wesley

Wos, L. (1988). Automated Reasoning: 33 Basic Research Problems, Prentice Hall

Wos, L., Mc Cune, W. (1992). The application of automated reasoning to questions in mathematics and logic, *Annals of Mathematics and Artificial Intelligence,* Volume 5 , 321-370

Zhang, J.Z., Chou, S.C., Gao, X.S. (1995). Automated production of traditional proofs for theorems in Euclidean geometry, *Annals of Mathematics and Artificial Intelligence,* Volume 13 , 109-137