The conjectures, found by SECTE, seem to capture the meaning of the studied concepts. If we want to have more semantic discovered by the system, we only need to give it more predicates and operations. However, as we will see in the next section, the selection of interesting conjectures is not perfect yet.

6. Perspectives

An interesting direction to further investigate is to enhance the selection of conjectures. For instance, actually, for the operation SetUnion (union two sets S1 and S2 in a set S3), the system has found that it is rarely true that S1 and S2 are equal. This conjecture is valid but not interesting since a more general one exists: it is rarely true that the result of SetEqual is true. A solution here is to find a representation that enables the conjectures to be ordered by generalization. This way, SECTE will be able to eliminate more conjectures.

A new direction is the study of the combinatorial search. The idea is that SECTE could have the same results with less search. To deeply understand this process, some measures should be defined, some heuristics be gathered and new methods be developed. A first step on this way, already started with SECTE, is the shifting of constraints from the conjecture level to the relation level. At the end, it should be very interesting because it could reveal us examples of an other type of meta-knowledge: knowledge about strategies [Pitrat 96].

These perspectives ensure that a lot of works is still available in that direction. The best being perhaps the possibility to have one day a system that will be an expert in the domain of generate-and-test methods.

7. Conclusion

In this paper, we have described a system, called SECTE, that can automatically generate interesting conjectures. First, we have seen that it was possible to define conjectures like a space composed of a frequency and two relations. Second, we have described the meta-knowledges used by the system to select interesting conjectures from this broad space. Experiments with SECTE have brought three results. The first result is that if you have operations and predicates concerning a domain then it can be studied by SECTE. the second result is that each conjecture represents a part of the semantic of the studied operation. The last result says that more you have predicates and operations about a domain and more the conjectures will be interesting . Finally, some perspectives have been suggested to better select interesting conjectures and to reduce the combinatorial search. One could conclude that any problem can be viewed as a search in a space. And in that case, a system with knowledge on generate-and-test methods will be a very general and interesting one.

8. References

[Lenat 82], Lenat D.B., AM: Discovery in mathematics as heuristic search, in Davis R. and Lenat D.B., Knowledge-based systems in Artificial Intelligence, McGraw-Hill, New York, 1982.
[Minsky 75], Minsky M., Frames, in Winston P., The Psychology of computer Vision, McGraw-Hill, New York, 1975.
[Pitrat 90], J.Pitrat, Métaconnaissance, futur de l'intelligence artificielle, Hermès, Paris, 1990.
[Pitrat 96], J.Pitrat, Méta-expliquer pour apprendre une stratégie, internal report 21, LAFORIA, 1996.

A second method to reduce the space of conjectures is to generate R1 and R2 and then to test if the corresponding conjecture could be interesting. To do this, SECTE uses meta-knowledges [Pitrat 90] which inform the system when a conjecture is not interesting. For instance, if SECTE has already studied the conjecture C3, which is represented like *(Always (SetMember %Dom1 %Dom2 True) (SetEqual %Dom1 %Range True)*), a meta-knowledge says that it is not interesting to study a conjecture like (- *(SetMember %Dom1 %Dom2 True) (SetEqual %Dom1 %Range False)*). The reason is that this conjecture can be deduced from the previous one. The general meta-knowledge used there was that '(- *R1 R2*) is not interesting if (*Always R1 ¬R2*) already exists'. Indeed, some of these meta-knowledges are used directly after the determination of R1 or R2. This way, we can stop the search more quickly. An example of this kind of meta-knowledge is '(*AlwaysTrue False*) is not an interesting relation'.

The third method is to use a canonical form to save the conjectures. This is used with the previous method to check if a similar conjecture doesn't already exist. The canonical form is determined by criteria of esthetics implicitly given initially. 'If we can have *Always* as the value of Q, get it' is one of these criteria. It can, for instance, change a conjecture like (*Never R1 R2*) in the conjecture (*Always R1 ¬R2*).

The last method is a global check each time a new conjecture is found. This check means that each already existing conjecture needs to be tested again. This is useful when a conjecture is interesting until an other has been found. An example of this is when SECTE has kept the conjecture *(Always (SetEqual %Dom2 %Range True) (SetMember %Dom1 %Range True))* and later found the conjecture C1: *(Always (AlwaysTrue True) (SetMember %Dom1 %Range True))*. When the first conjecture had been checked, SECTE realized that it was a specialization of the newly found conjecture and then decided to forget it.

In this section, we have described the different methods SECTE uses to choose the final conjectures. We have also stressed that knowledge on conjectures, a kind of meta-knowledge, were the basis of this process. The next section will show the results of the system.

5. Results

Four different domains have been tested : Set, Bag, Liste and Number. In fact, any domain can be used by the system. The only thing to do is to find operations and predicates on the new domain. Below are listed the exhaustive operations and predicates of Set and Number given to the system. The other concepts are too similar with Set to be interesting here.

Operations on Set : union, intersection, difference of two sets, delete something from a set, insert something into a set, get a random element from a set and create a set from anything.
Predicates on Set : membership and equality.
Operations on Number : incrementation, decrementation.
Predicates on Number : equality, inferior.

Here are some good conjectures generated by the system:

When you decrement N1 into N2, N2 is always inferior to N1.
When you increment N1 into N2, N1 and N2 are always not equal.
When you intersect or union two sets, which are equal, the result is always the same set. (four conjectures)
When you remove an element from a set, it is no more a member of this set.

Relations have at most 4 dimensions P, D1, D2 and R. P is a predicate for which the domain is included in the domain of the studied operation. D1 and D2, which are optional, are respectively the values of the Dom1 and Dom2 (if any) of the predicate. Their value can be %Dom1, %Dom2 or %Range. They represent respectively the Dom1, Dom2 or Range of the concerned operation. R is the value of the range of the predicate and can take the values True or False. The meaning is that the result of applying P on (D1 D2) is Range. The two relations of the conjecture C1 are:

C1-R1:	(**P** *AlwaysTrue*	**D1** -	**D2** -	**R** *True*)
C1-R2:	(**P** *SetMember*	**D1** *%Dom1* (x)	**D2** *%Range* (s2)	**R** *True*)

Finally, the conjecture C1 may be read like:

C1: (**Q**: *Always* **R1**: *(AlwaysTrue True)* **R2** *(SetMember %Dom1 %Range True)*)

3.3. Generation of conjectures

We have defined the space of conjectures as a frame-like structure. The method for generating one conjecture is trivially to give a value to each dimension of this structure. Let's see for instance what is the number of potential conjectures, NbPotConj, for the operation SetInsert.

NbPotConj is by definition NbQ * NbR1 * NbR2, where NbQ is the number of values for Q, NbR1 the number of value for R1 and NbR2 the number of values for R2. NbQ is 6 as defined in the previous subsection. NbR1 and NbR2 are equal by definition. We'll now determine their value. Actually, there are three predicates that have their domain included in the domain of SetInsert: SetMember, SetEqual and AlwaysTrue (for which it is always the case). If P is SetMember, %Dom1 is the unique value D1 can take because the only Anything of SetInsert is Dom1. D2 can have %Dom2 and %Range as value since they are Sets. True and False are the possible values for R (as for any other predicate). So, NbR1 for P = SetMember is 1*2*2 = 4. Similarly, NbR1 for P = SetEqual is 2*2*2 = 8 and for AlwaysTrue is 2. So NbR1 and NbR2 are 4+8+2 = 14 and NbPotConj is finally 6*14*14 = 1176.

In this section, we have described the representation of conjectures used by SECTE. We have also noticed that the corresponding space was very broad. Conjectures from this space are surely not all interesting. The next section will reveal us how the system reduces it.

4. Selection of Conjectures

This section describes the different methods used by SECTE to reduce the space of valid conjectures.

A first method is the use of functions that determine the value of dimensions. Actually, there is only one function of this kind in SECTE. This function determines the value of the dimension Q. This is done by computing the number of examples and non-examples that satisfy the 'if R1 then R2' rule of the concerned conjecture. After that, the function returns a value according to the frequency of the positive examples. For instance, for the conjecture C1: *(Always (AlwaysTrue True) (SetMember %Dom1 %Range True))*, SECTE tests for each example of SetInsert, created by the system as in AM [Lenat 82], if *(SetMember %Dom1 %Range)* is true. In this case, the value Often is returned because the computed frequency of examples is between 80% and 99%.

E1: sometimes a number is even
E2: you obtain an even number when you add two even numbers
E3: if you increment a number and if this number is an odd one then the result is even
E4: two sets are rarely equal

We can notice that all these examples can be expressed as a frequency followed by an implication. Below are their new form.

E1’: sometimes, if X is a number then X is an even number
E2’: always, if X is the result of an addition between two even numbers then X is even
E3’: always, if X is the result of an incrementation of Y and Y is odd then X is not odd
(we assume there that the concept IncrementOddNumber is not defined)
E4’: rarely, if X is the result of an equality between two sets then X is true

Finally, we can see the correspondences between these examples and the following frame-like sentence « For the operation **OP**, it is **Q** true that if **R1** then **R2** ».

E1’’: **OP**=IsaNumber,	**Q**=Sometimes,	**R1**=IsaTrue(Result(Op)) is true	**R2**=IsaEven(Dom(OP)) is true
E2’’: **OP**=Add2Even,	**Q**=Always,	**R1**=ItIsTrue is true,	**R2**=IsaEven(Result(OP)) is true
E3’’: **OP**=Increment,	**Q**=Always,	**R1**=IsaOdd(Dom(OP)) is true	**R2**=IsaOdd(Result(OP)) is false
E4’’: **OP**=SetEqual,	**Q**=Rarely,	**R1**=ItIsTrue is true,	**R2**=IsaTrue(Result(OP)) is true

The proposed **(OP Q R1 R2)** structure seems sufficient to generate a lot of interesting conjectures. We can now describe a computational representation for the conjectures.

3.2. Representation of conjectures

The chosen type of representation for concepts is the frame-like one [Minsky 75]. Concepts are defined similarly as in AM [Lenat 82]. Only concepts and slots that interest us will be described in this paper.

Operations and predicates are the concepts on which the system creates conjectures. They have at most 3 dimensions Dom1, Dom2 and Range. Dom1 and Dom2, which are optional, are the type of domain on which apply the operation. Range is the type of result of this application. Below is the partial description of concepts needed to explain the conjecture C1 (see Section 2).

SetInsert	(Dom1 *Anything*	Dom2 *Set*	Range *Set*)
SetMember	(Dom1 *Anything*	Dom2 *Set*	Range *TruthValue*)
SetEqual	(Dom1 *Set*	Dom2 *Set*	Range *TruthValue*)
AlwaysTrue	(Dom1 -	Dom2 -	Range *TruthValue*)
By definition, this last concept always returns true.			

Conjectures have 3 dimensions Q, R1 and R2. Q is the qualification of the conjecture and R1 and R2 are the relations of the conjecture. Qualifications can take one of the six following values : Always, Often, Sometimes, Rarely, Never and Undefined. The meaning of conjectures is ‘it is Q true that if R1 then R2’. The conjecture C1 has the form:

C1: (**Q** *Always* **R1** *it is True* **R2** *X is a member of S2*)

1. Introduction

« Recode just the conjecture-finding heuristics as constraints on a new, purely syntactic 'legal conjecture generator.' A simple Generate-and-Test paradigm would be used to synthesize and examine large numbers of conjectures. ». This is one of the open problems Lenat suggested after his realization of AM [Lenat 82]. AM is a discovery system in the mathematical field. It starts with concepts from elementary set theory, creates new concepts and then proposes new conjectures about them. Unfortunately, the space of conjectures is not clearly defined by Lenat. We have no way to know its range neither can we estimate the interest of the found conjectures. For these reasons, it seems important to focus on the explicitation of the space of conjectures.

In this paper, we describe a system, called SECTE, which automatically generate interesting conjectures. To begin with, we'll see the conjectures discovered by the system for one concept. Then, we'll describe the definition of conjectures that is implemented in SECTE. Next, we'll enumerate the different kinds of meta-knowledge used by the system for the selection of the interesting conjectures. After that, we'll present the studied domains as well as the obtained results. And finally, we'll give some perspectives before we conclude.

2. An Example of Discovered Conjectures

In this section, we'll see the results of the system for the task « generate all interesting conjectures of SetInsert ».

First, we have to define the concept SetInsert. SetInsert is an operation that inserts an Anything, say X, into a set S1 to give a set S2. For instance, if the arguments of SetInsert are A, an Anything, and (B), the Set containing B, the result will be (A B), which is the Set containing A and B.

Below, are the 5 conjectures, created by SECTE, out of the 1176 potentially valid ones.

C1: It is always true that X is a member of S2.
C2: It is rarely true that X is a member of S1.
C3: It is always true that if X is a member of S1 then S1 is equal to S2.
C4: It is always true that if X is not a member of S1 then S1 is not equal to S2.
C5: It is rarely true that S1 is equal to S2.

This example surely raised certain questions with the reader. What is a potentially valid conjecture ? How the system finally chooses only five of them ? The answers to these questions are the topics of the next two sections.

3. The Nature of Conjectures

In this section, we propose to answer the following questions : what is a conjecture, how we represent them, and how SECTE can generate them.

3.1. Definition of conjectures

What we need to define conjectures is a common description to express them. To obtain this description, we'll see how to merge the following examples of conjectures.

Christophe TURLE
University Pierre et Marie Curie
LAFORIA-IBP Laboratory
4, place Jussieu
75252 Paris cedex 05, France
turle@laforia.ibp.fr

Abstract: Since we can't give a system all the knowledge of the world, A.I. systems will have to discover it. They will also have to organize this knowledge in order to make it useful. This task is not easy because these systems may produce too many useless concepts which at last overwhelm them. One solution to the problem of creating only good concepts is to guide the system with interesting conjectures. Unfortunately, the problem is just translated since the new concepts will also need conjectures. A better solution is then to enable the system to discover these conjectures. In this paper we present a general method that generates interesting conjectures. This method is based on the notions of space representation and generate-and-test procedure using knowledge on conjectures to constrain the search. We present results in two domains: the elementary set theory and a subset of the number theory. These results come from experiments realized with SECTE which is the successor of a new implementation of Lenat's AM system [Lenat 82].

Key-words: discovery, knowledge representation, meta-knowledge, AM.

Résumé : Puisque nous ne pouvons pas donner à un système toute la connaissance du monde, les systèmes d' I.A. devront la découvrir. Ils devront aussi organiser cette connaissance afin de la rendre utile. Cette tâche n'est pas simple car ces systèmes peuvent produire trop de concepts inutiles qui à la fin les submergent. Une solution au problème de créer uniquement de bons concepts est de guider le système à l'aide de conjectures intéressantes. Malheureusement, le problème est juste déplacé puisque les nouveaux concepts auront aussi besoin de conjectures. Une meilleure solution est alors de donner au système la capacité de découvrir ces conjectures. Dans cette article, nous présentons une méthode générale qui génère des conjectures intéressantes. Cette méthode est basée sur les notions de représentation espace et de procédures générer-et-tester utilisant des connaissances sur les conjectures afin de contraindre la recherche. Nous présentons des résultats dans deux domaines : la théorie élémentaire des ensembles et une partie de la théorie des nombres. Ces résultats proviennent d'expériences réalisées avec SECTE qui est le successeur d'une nouvelle implémentation du système AM de Lenat [Lenat 82].

Mots-clés : découverte, représentation des connaissances, métaconnaissances, AM.

Cet article sera publié dans les actes de EXPERSYS-96