DISCOURSE CONSISTENCY AND MANY-SORTED LOGIC

Jean VERONIS
Groupe Représentation et Traitement des Connaissances
Centre National de la Recherche Scientifique
30, Chemin Joseph Aiguier
13402 MARSEILLE CEDEX 9 - FRANCE

ABSTRACT

We propose the use of a many-sorted logic based on a boolean lattice of sorts, with polymorphic functions and predicates, for natural language understanding. This type of logic provides a unified framework for various problems such as discourse consistency verification, polysemy and "abuses" of terms, syntactic ambiguity solving and anaphora resolution. In addition, this logic enables intelligent diagnosis of categorial constraint violations between predicates and arguments.

I. INTRODUCTION

In most fields for which natural language interfacing is relevant (expert systems, data bases, etc.), the universe is divided into categories, and inter-object relations are defined only between given categories. Thus, speaking about the hypotenuse of a circle or about the radius of a triangle does not make any sense*. Natural language interfaces must by all means prevent such inconsistencies.

Herein, we propose the use of a many-sorted logic for inconsistency checking. In this approach, ill-sorted formulae do not belong to the logic language, and the problem of their truth value does not even arise, no more so than if they were syntactically ill-formed. The many-sorted logic used is based on a boolean lattice of sorts which reflects the categorial organisation of the universe. It includes polymorphic functions and predicates which enable us to handle polysemy problems (metonymy, "abuses" of terms, etc.). The sort computation mechanisms of this logic allow for consistency checking even if the sort of an objet is not expressed, thus allowing a rather elliptic style in the dialog, and enable us to provide intelligent diagnosis of the various possible situations of inconsistency. In addition, this logic automatically solves many cases of syntactic ambiguity and makes anaphora resolution easier. These various features provide for greater flexibility in natural language man-machine dialogue.

While it is obvious that the above-mentioned natural language understanding (N.L.U.) problems can be solved by other means, we show here that many-sorted logics provide a *unified* framework and efficient algorithms.

II. SORT SYSTEM

The interest of many-sorted logics in computer science in general has been stressed time and again, since they often make theorem proving easier by cutting down the search space (Walther, 1984, Cohn, 1985). In addition, they also constitute an efficient knowledge representation tool (Hayes, 1971), hence our choice to use this feature in N.L.U.

*Examples throughout this paper are taken from a natural language interface for a C.A.I. system for plane geometry under development at G.R.T.C.

We use a particular many-sorted logic which has some similarities to that proposed by Cohn (1983), but also some differences, particularly in the way we define quantifiers and sorting functions (unfortunately, we have no room here to go into technical matters : see Véronis, 1987). In this logic, the set of sorts constitutes a boolean lattice for the partial order relation ≤ ("a-sort-of") between sorts. This hypothesis is not at all a restrictive one, since the set of subsets 2^U of the universe U of interpretation is really a boolean lattice for set inclusion, and since there exists a rather natural boolean morphism φ which maps the sort lattice into 2^U (Figure 1). Given a sort s, $\varphi(s)$ is the subset of individuals "which are" of sort s. Therefore, the relation x "is-an" s can be translated by $x \in \varphi(s)$. Hence, the boolean sort lattice is a perfect mirror of the organization of the universe.

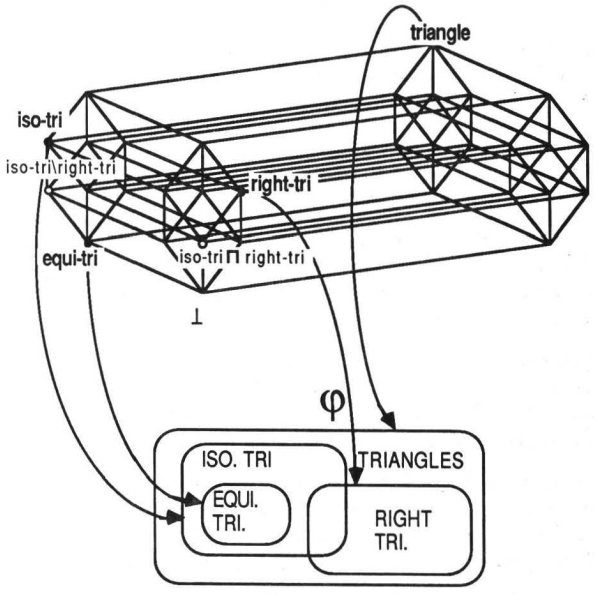

Figure 1 : boolean lattice of sorts

There is no need, of course, to describe *in extenso* the sort lattice, by giving a name to each sort. We will have *eponymous* and *anonymous* sorts, as in Cohn (1983). Eponymous sorts are those which are actually named, and correspond to sort predicates (e.g. *triangle, iso-tri, right-tri, equi-tri*). They are related to a family of subsets of the universe which constitute pertinent categories from a cognitive point of view. Anonymous sorts correspond to all the other subsets obtained by boolean closure from this family. They are useful in computation, but do not correspond to cognitively pertinent categories. They can be automatically expressed (only when the inference engine uses them) in

terms of the eponymous sorts by means of the lattice operators $\sqcap, \sqcup$, and $\setminus$ (e.g. *iso-tri* $\sqcap$ *right-tri* will be associated with triangles which are both isosceles and right-angled, *iso-tri* $\setminus$ *equi-tri* with those which are isosceles but not equilateral, etc.).

Most many-sorted logics do not impose a boolean lattice structure on the sort system, and the relation $\leq$ is seen simply as a common partial order. This can be debated on resolution efficiency and completeness grounds (Schmidt-Schauss, 1985). Nevertheless, as we previously said, in N.L.U. we are concerned mainly with knowledge representation, and it is easy to see that if the sort system is not a boolean lattice, some sorts will be "lacking". For example, we need complements since if a triangle is known as *not* being right-angled, we have no right to speak about its hypotenuse. Similar arguments can be advanced concerning meets and joins. Moreover, given the mechanism of eponymous/anonymous sorts described above, it is just as easy for a user to give a boolean system as to give a common partial order. One has simply to describe overlapping or disjointness of sorts, for example in terms of atomic sorts.

III. POLYMORPHISM

Polymorphism (Stratchey, 1967) is a very interesting feature as regards N.L.U., since it provides a good framework for polysemy. We will first distinguish between two types of polymorphism : *true* (or *universal*) and *apparent* (or *ad hoc*) polymorphism (Cardelli and Wegner, 1985).

In universal polymorphism, the same function or predicate works uniformly on a range of sorts. It can be subdivided into *inclusion polymorphism,* which is due to the inclusion of categories (for example, isoceles triangles inherit all functions and predicates that are allowed for triangles), and *parametric polymorphism,* in which an explicit or implicit parameter determines the coherence of functions or predicates relative to their arguments. In our system, set membership can be seen as a case of parametric polymorphism : points belong to segments, lines and circles; lines belong to line directions, but segments do not belong to lines, etc.

In *ad hoc* polymorphism, while the same function or predicate (and the same word in natural language, assuming that we try to maintain, inasmuch as possible, a one-to-one correspondance between words and functions or predicates) works on different sorts, it corresponds in the interpretation to different, and even unrelated, functions or relations. It also subdivides into two major types, corresponding to different forms of polysemy in natural language.

We have first a certain number of "abuses" of terms. Thus, sides and heights of a triangle are seen sometimes as lines, sometimes as segments; one can speak of perpendicular lines as well as perpendicular segments, and even about lines perpendicular to segments. This phenomenon can be seen as a form of *metonymy* : when we say that two *segments* are perpendicular, we want to say that the *lines* which contain them are perpendicular. Purist may consider that one should use distincts words, and distinct predicates, since one actually has different mathematical relations. Nevertheless, this is an unnecessary imposition on the user and generally leads to larger axiomatisations due to the duplication of information (e.g. in many cases, the same theorem can apply to the "perpendicularity" of both lines and segments). Through the use of polymorphic predicates and functions one can considerably simplify the system, and maintain one-to-one correspondences between words and predicates. Appropriate treatment precludes contradictions, and hence provides for a better modelling both from cognitive and linguistic view points. This first type of polysemy can be related to *coercion polymorphism* : the relations (e.g. perpendicularity) are actually defined for a given category (lines) in the universe, and their application to other categories (segments) through "abuse" of term can be understood only through a *coercion* schema (the *lines* containing those segments are perpendicular).

The second type of polysemy is quite different. Although we can speak of the *base* of an isosceles triangle, as well as the *bases* of a trapezoïd, the sub-jacent mathematical phenomena are completely unrelated. The base of an isosceles triangle is defined as a side adjacent to two equal sides, whereas the bases of a trapezoïd are two parallel opposite sides. This is no longer an "abuse" of a term based on a metonymic relationship (and, in fact, experts do *not* speak of "abuse" of term in such a case). Instead, the same word is accidentally "overloaded" by two unrelated meanings. Here again, there is no need to impose a rigid terminology, and this kind of polysemy can be handled in the logic system as *overloading polymorphism.*

IV. CONSISTENCY VERIFICATION

The natural language interface must verify the consistency of the dialog. This task is somewhat complicated by the fact that sorts may be expressed or left implicit. Phrases such as "*circle C*" or "*A is a point* " explicitly assign sorts to objects, since the words "*circle*" or "*point*" correspond to sort predicates. In this case, the rest of the statement must be checked in order to verify consistency with these sorts. Quite often however, the sort of objects is not expressed. For example : "*A and A' belong to D and D' respectively, and are distinct from O.* ". Entire statements can be given in this fashion. In this case, the analysis system (and the human reader) must be able to reconstruct the sort of each object.

Due to polymorphism, this sort computation is not purely trivial. From a sentence such as "*S is the base of T*", we cannot directly infer unique sorts for objects *S* and *T*. Following the analysis of each sentence (or phrase) corresponding to an atomic formula, we create a *sort array* in which we keep track of the well-sorted domain of this formula, that is to say the greatest sorts which can be given to each variable while maintaining coherence. Roughly speaking, sentence or phrase combination corresponds to logical operations on formulae : conjunction, disjunction, negation, implication. We assume that a compound formula makes sense only if each of its constituents does, and this leads us to define a meet operation between sort arrays. Basically, well-sorted domains corresponding to constituent formulae are intersected to give the well-sorted domain of the compound formula, as shown in Figure 2.

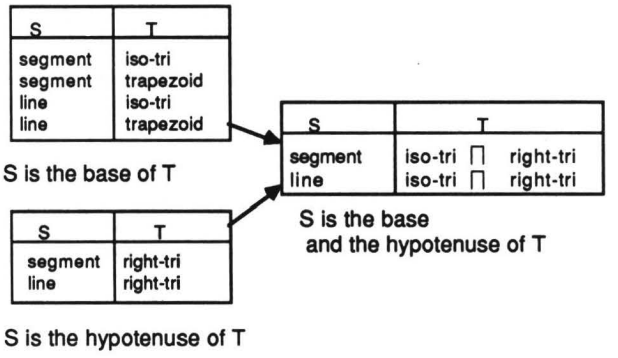

Figure 2 : sort computation

If the resulting array is empty, no sort can be attributed to variables in order to render the formula coherent. The formula is therefore ill-sorted, and discourse inconsistency is detected (Figure 3). We will describe below the diagnostic process corresponding to this case. When the analysis of the entire statement (in theorem and problem acquisition), or of a single sentence (in demonstration dialogs), is complete, and inconstency is not detected, two cases can occur.

1) If the final array enables us to express the sort of each variable as an eponymous sort, or as a meet of eponymous sorts, the discourse is quite satisfactory.

2) If not, that is to say if the domain of some variable can only be expressed by means of joins or complements (e.g. *x* is a segment *or* a line), the discourse is ambiguous, and this situation must be reported to the user.

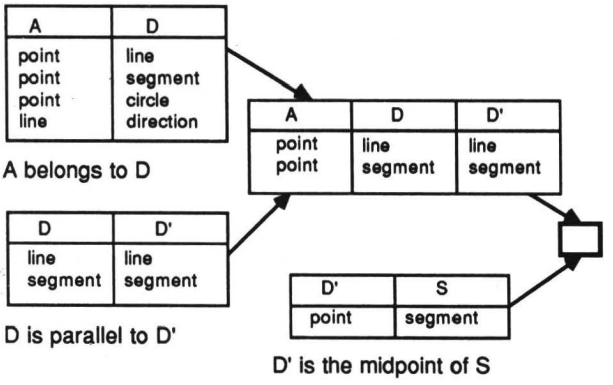

Figure 3 : Inconsistency

Various cases of inconsistency can be distinguished, hence intelligent diagnoses can be provided instead of some standard laconic message such as "sentence rejected", which hardly helps the user to repair the mistake. Figure 4 shows various occurring situations.

The sort computation also enables us to remove most syntactic ambiguities. Let us take for example the ambiguous sentence : *"[AB] is a chord of circle C of which D is the perpendicular bisector"* . Two syntactic parsings can be performed depending on whether the relative clause is attached to *chord* or to *circle*. The sort computation automatically gives the second analysis as inconsistent. Finally, this technique can also facilitate the resolution of anaphora, by cutting down the space of candidate objects.

V. CONCLUSION

A many-sorted logic based on a boolean lattice of sorts, with polymorphic functions and predicates seems well-suited to natural language understanding. It provides a unified framework for various problems such as discourse consistency verification, polysemy and "abuses" of terms, and facilitates syntactic ambiguity solving and anaphora resolution. In addition, this logic enables intelligent diagnosis of categorial constraint violations. We believe that our approach is not specific to the particular domain of geometry, from which we drew our examples, but, to the contrary, can be extended to many fields in which the universe of interpretation is divided into various categories of objects.

1) If the sort *s* of an object is expressed :

a) the sort is re-expressed farther on :

Let C be a circle.....
Point C belongs to line D.
error message :
| C was previously defined as a *circle.*

b) the sort computation gives a sort *s'* incompatible with *s* :

Let C be a circle.....
C is perpendicular to D.
error message :
| C is a *circle* and cannot be *perpendicular* to a
| line. Can only be perpendular to a line :
| - another line
| - a segment

c) the sort computation gives a sort *s'* < *s* :

Let ABC be a triangle.....
.....the hypotenuse of ABC.....
error message :
| You did not specify that *triangle* ABC was a
| *right-angled triangle.* Do you confirm this
| specification (y/n)?

2) the sort of an object is not expressed :

A belongs to D.....
D is the mid-point of S.
error message :
| Taking into account what has been previously
| said, D cannot be a *point.* It can only be :
| - a line
| - a segment
| - a circle
| - a direction

Figure 4 : intelligent error diagnosis

REFERENCES

1. CARDELLI, L., WEGNER, P., On understanding types, data abstraction and polymorphism, *A.C.M. Computing Surveys,* 1985, 17,4, 471-522
2. COHN, A.G., *Mechanising a particularly expressive many-sorted logic,* Ph. D. Thesis, University of Essex, 1983
3. COHN, A.G., On the solution of *Schubert's Steamroller* in many-sorted logic, *IJCAI,* 1985, 1169-1174
4. HAYES, P. J., A logic of actions, in *Machine Intelligence 6,* Metamathematics Unit, University of Edinburgh, 1971, 495-520
5. SCHMIDT-SCHAUSS, M., A many-sorted calculus with polymorphic functions based on resolution and paramodulation, *IJCAI,* 1985, 1162-1168
6. STRACHEY, Fundamental concepts in programming languages, *Lecture Notes for International Summer School in Computer Programming,* Copenhagen, 1967
7. VERONIS, J., Verifications de cohérence dans le dialogue homme-machine en langage naturel, *Note Interne GRTC N°186,* 1987
8. WALTHER, C., A mechanical solution of Schubert's Steamroller by many-sorted resolution, in *Proc. NCAI 4,* Austin, 1984, 330-334