SOLVING AMBIGUITIES IN THE SEMANTIC REPRESENTATION OF TEXTS

Marie-Claude Landau

IBM France Paris Scientific Center
3-5 Place Vendôme 75021 PARIS cedex 01 FRANCE

Abstract

One of the issues of Artificial Intelligence is the transfer of the knowledge conveyed by Natural Language into formalisms that a computer can interpret. In the Natural Language Processing department of the IBM France Paris Scientific Center, we are developing and evaluating a system prototype whose purpose is to build a semantic representation of written French texts in a rigorous formal model (the Conceptual Graph model, introduced by J.F Sowa [10]).

The semantic representation of texts may then be used in various applications, such as intelligent information retrieval. The accuracy of the semantic representation is therefore crucial in order to obtain valid results in any subsequent applications. In this article we explain how ambiguities related to Natural Language may be solved by semantic analysis using the Conceptual Graph model.

Key words

Natural Language Understanding, Computational Linguistics, Conceptual Graph Model.

Introduction

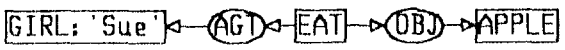
In the system prototype we have been developing, the analysis of a text is carried out in two steps: first syntactic and then semantic [1].
We assume that the syntax of a text conveys some meaning, but since our syntactic analyzer does not take semantics into account, a lot of ambiguities remain:

- Lexical ambiguities, coming from the fact that the same word may correspond to several lemmas in the syntactic dictionary. This kind of ambiguity can be almost completely solved by the syntactic analyzer.
- Structural ambiguities, a consequence of the multiple possible attachments of the syntactic components in a sentence. This kind of ambiguity may be solved to a large extent by the semantic analyzer.
- Anaphoric ambiguities, that could be solved in part by syntactic analysis *within* a sentence [3], but cannot be solved *across different sentences* because a syntactic analyzer processes each sentence independently. In our system, the resolution of anaphoric ambiguities is done uniquely by the semantic analyzer.
- Ellipses, that could also be solved in part by syntactic analysis. But an incomplete syntactic analysis may in some cases be complemented by the semantic analysis.
- Semantic ambiguities coming from polysemous lemmas, that can only be solved at the semantic level (unless a polysemy leads to different syntactic constructions).

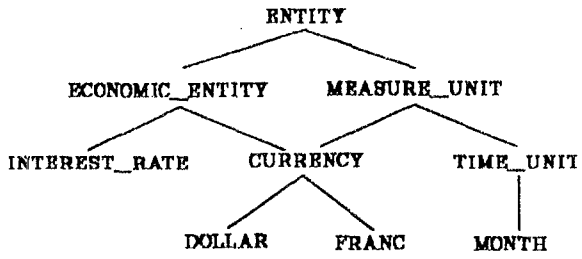
In this article, we concentrate especially on the **practical solving** of the different kinds of ambiguities, showing that these problems are inter-related and may be solved by unified methods.

The Conceptual Graph model

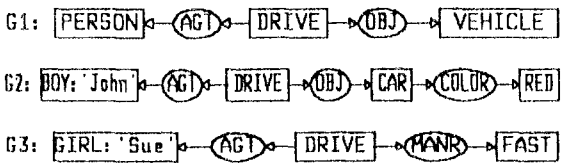
The Conceptual Graph model is a very promising unified model, because it generalizes many ideas contained in preceding works on natural language semantics, such as Fillmore [7], Schank [9], Montague [5], Wilks [12], and Kamp [8], for example.
For the sake of clarity, we briefly recall here the Conceptual Graph model introduced by J.F. Sowa [10]. A Conceptual Graph is an oriented graph made up of **concept nodes** related by **conceptual relation** edges. The concepts are represented by boxes, the relations by circles. Example:



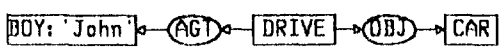
The concepts may have referents which specialize them. A referent can be a constant ('Sue') to denote individuals, a variable to denote cross-references, or more complex expressions. Most of the relations are binary relations (OBJ), some are unary. The concepts are organized in a concept type lattice with a partial ordering relation. The top concept type is ENTITY. Example:



Conceptual Graphs may be combined together using various algorithms, the most important of which are the **projection** and the **join** algorithms. They are pattern matching algorithms which take the concept types hierarchy into account.
The **projection** of one Conceptual Graph into another one is a restriction of the first graph to a sub-graph of the second one. The projection also gives the pending edges of the second Conceptual Graph in relation to the result.
The **join** of two Conceptual Graphs forms a common overlap, while keeping the most specialized concept types in the result, and attaches to the common overlap the pending edges remaining in the two graphs.



Result of the **projection** of G1 into G2:



Pending edge:

Result of the **join** of G1 and G3:

The semantic analyzer: general method

The semantic analyzer produces one or more Conceptual Graphs for each sentence, including cross-references within a sentence or between different sentences.
Our semantic analyzer is written in the VM/Programming in Logic (VM/PROLOG) programming language [11]. The semantic analyzer takes as input the annotated syntactic tree(s) resulting from the syntactic analysis. Applying compositionality rules, it links together the Conceptual Graphs corresponding to each word or locution of the sentence, according to the indications given by the syntactic tree(s).
The Conceptual Graphs for each word or locution are retrieved from a **semantic lexicon**. The words of the Natural Language may be coded in a semantic lexicon general to Natural Language and/or in a semantic lexicon specific to an application. In our project, we have concentrated on developing specialized semantic lexicons, in order to get fast results on texts dealing with a specific subject (economics, pharmacology).
In cases of polysemy there may be several entries (hence several Conceptual Graphs) for one word in the semantic lexicon. If, however, a word is missing in the semantic lexicon, default options are taken.

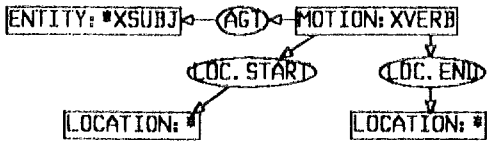
The directed join algorithm as a disambiguation tool

The Conceptual Graphs for words are linked by an algorithm that we call the **directed join**. In fact, the directed join is a deterministic version of the join algorithm described by J.F. Sowa: we force such and such concept box in the first graph to be mapped onto such and such concept box in the second graph, by use of attachment point labels which lie inside the concept boxes. The join may then be propagated along the edges related to those initial concept boxes. Semantic constraints on the concept types, contained in the concept type lattice, make it possible to rule out invalid polysemous combinations, and in some cases to discard non-pertinent syntactic analyses.

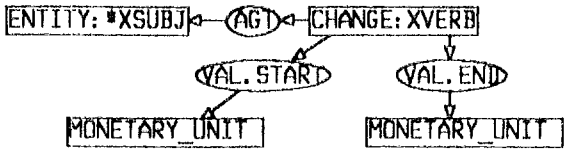
In addition, we have implemented a **directed join management algorithm** which allows the "best" possible solution to be chosen. Indeed, when two semantic structures must be linked together, all the conceptual choices (corresponding to the different entries for each word in the semantic lexicon)

are simultaneously taken into account by the directed join management algorithm, which only keeps the solutions leading to a maximum overlap between the two sets of Conceptual Graphs (according to the link constraints).
For example, suppose we have the following coding for the verb "passer" (*"to go from ... to"*) in the semantic lexicon:

VERB('passer',1) is

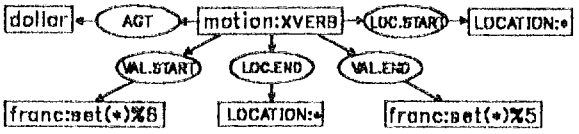


VERB('passer',2) is

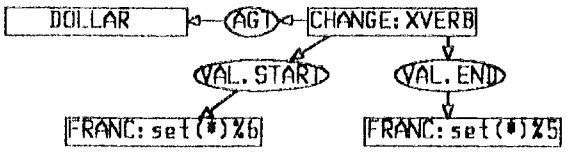


For the sentence "le dollar est passé de 6 francs à 5 francs", (*"The dollar went down from 6 francs to 5 francs"*) the directed join algorithm will only give solution 2, automatically discarding solution 1.

SOLUTION 1 is



SOLUTION 2 is



Therefore, the final result is usually not the combinatorial product of all the entries of polysemous words in the semantic lexicon.

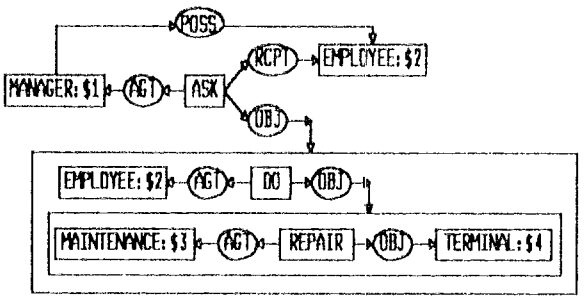
We thus see that the directed join algorithm is a powerful tool which can help disambiguate polysemy. It also helps fill in the gaps of incomplete syntactic information, as well as solve anaphors, as we shall explain below.

Processing of incomplete syntactic information

We prefer to speak here of incomplete syntactic information rather than of ellipses, in that the solving of true ellipses has not yet been done in our system.
In our system, the solving of incomplete syntactic information deals with missing subjects of complement clauses (infinitive verbs, verbal prepositional groups). The choice of the missing subject is made according to:

- the preposition introducing the complement clause (if applicable),
- the subject, object and dative of the main verb (i.e. the verb to which the complement clause is syntactically related),
- in some cases, the adverbial phrases of the complement clause.

For this processing it is necessary to have a knowledge base about the verbs of the Natural Language, along with their possible prepositional syntactic constructions. This knowledge base is organized into classes of verbs for which similar syntactic constructions lead to the same choice for the missing subject. Surprisingly enough, we have found that these classes also correspond in French to semantic classes (necessity, motion, perception, accompaniment, intention, delegation of power, etc.). Our algorithm has been written for the French language and should be partially or totally rewritten for other Natural Languages.
Here is an example of the kind of results we get:
"Le directeur demande à son employé de faire réparer le terminal par le service d'entretien" (*"The manager asks his employee to have the terminal repaired by the maintenance people"*)



Sometimes, the solution is not so straightforward. For example, let us consider the sentences:
"J'ai entendu jouer les enfants" (*"I heard the children playing"*)
"J'ai entendu jouer la musique" (*"I heard the music playing"*)
In one of these sentences (both in French and in English), the noun phrase following the infinitive is its subject, in the other it is

its object. Yet the structure of the sentences appears the same. Only by checking the semantic constraints with the directed join algorithm will the right interpretation be given. This is why, in our system, the processing of incomplete syntactic information is done at the level of semantic analysis rather than at the level of syntactic analysis.

Processing of anaphors

In this paragraph we group together the solving of the following co-reference problems, since the same resolution method is used:

Personal pronouns (*"he"*, *"them"*, ...)
Demonstrative pronouns (*"this one"*, *"those ones"*, ...)
Demonstrative determiners (*"this person"*, ...)
Noun ellipses (*"another one"*, *"that of"*, ...)
Possessive pronouns (*"theirs"*, ...)
Possessive determiners (*"her coat"*, ...)

The solving of a co-reference problem consists in instantiating the anaphoric element by assigning to it a concept type and possibly a referent which have already been used in the text. In some cases, it is also necessary to have a look-ahead procedure which scans the text forwards.

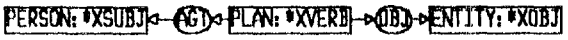
Backward search algorithm

In our system, the backward search is done by scanning a LIFO stack of concepts and referents.
Before starting to build a Conceptual Graph for the sentence, all the nouns (proper or common nouns, not preceded by a demonstrative determiner) and anaphors are processed in the order in which they appear in the sentence.
We assign to each of the nouns a new referent number (or new set of referents in the case of polysemy) and we store in a LIFO stack the sentence sequence number, the lemma, the noun Conceptual Graph(s), its referent(s), its gender and number. This processing of nouns is done once and for all, several syntactic analyses giving rise to the same referent number for the same noun at the same place in the sentence.
As for the anaphors, the stack is scanned LIFO and gender and number are checked. The result of this search is a set of possible solutions. In fact, the set of possible solutions for an anaphor may be viewed as an **"extended polysemy"**. For reasons of pragmatism and performance, the search is limited to a definite number of sentences upward in the text. This number is parameterized and may be specified by the user.
When the set of graphs corresponding to an anaphor is linked to its context (e.g. a pronoun subject to a verb), the "best" solutions are chosen by the directed join management algorithm, as explained above in the example of polysemy (*"to go from... to..."*).

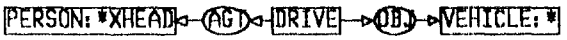
Then the solution corresponding to the most recent entry in the concept stack is selected, to avoid having too many solutions. This is done by way of a projection of the Conceptual Graph contained in the stack into the result of the directed join. However this selection of the most recent solution may backtrack: this is useful if the set of graphs for the anaphor has to be linked several times. (This is the case for coordinated verbs with the same subject, or for infinitives with the same subject as the main verb, for example). In this case, thanks to the directed join management algorithm, the best solution of the whole process is chosen.

Example:
"Le pilote et le garçon sont arrivés hier. Il projette de piloter l'avion" (*"The pilot and the boy came yesterday. He plans to pilot the plane"*)
Suppose we have the following entries in the semantic lexicon:

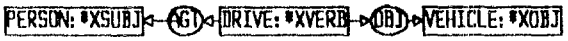
garçon (*boy*) < PERSON in the lattice
avion (*plane*) < VEHICLE in the lattice
VERB('projeter',1) is (*to plan*)



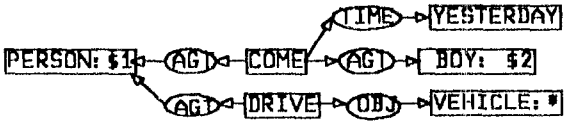
SUBS('pilote',1) is (*pilot*)



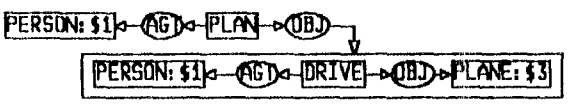
VERB('piloter',1) is (*to pilot*)



The result for the first sentence is:



The result for the second sentence is:



Forward search algorithm

If no solution has been found in the stack with the backward search algorithm, or if the

solutions found have led to a failure in the linkage to the context, then the forward search algorithm is activated. This is easy since we already have in the stack the information concerning all the nouns of the sentence. If the forward search also leads to a failure, our system simply prompts the user. If no answer is given (or if we are in batch mode), the system instantiates the anaphor to the most general concept in the lattice, which is ENTITY.

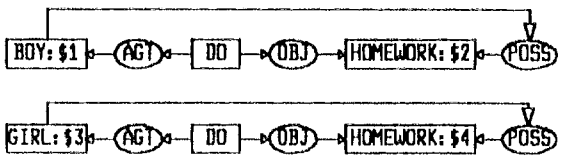
However, it is not always sufficient to activate the forward search algorithm only in cases of total failure of the backward search algorithm. In fact, some syntactic constructions (corresponding to **cataphoric** relations) should automatically start the forward search algorithm, even though there might be some solutions given by the backward search algorithm. Such cataphoric relations may correspond to set expressions that emphasize a word which appears later in the sentence (at least, in French): "Il marche bien, ce programme" (Literally, *"It works well, this program"*). "Il" (*"it"*) refers to "programme" (*"program"*).

Miscellaneous problems related to anaphors

In the case of demonstrative determiners, the information corresponding to the concept type is already given by the noun. But there may be set expressions for which the noun following the demonstrative does not correspond exactly to a previous word in the text. Example: "La hausse du dollar s'est intensifiée hier à Paris. Cette évolution a provoqué ..." (*"The rise of the dollar sharpened yesterday in Paris. This change caused ..."*) In this case, the search is the stack must not be made according to words: instead, a **projection** of the Conceptual Graph(s) of the noun (*"change"*) must be made into the Conceptual Graphs of the stack.

For noun ellipses (*"another one"*, *"that of"*), the thing to do is to search only for a concept type in the stack, and to assign a new referent to it. For example, the sentence: "Le déficit de 1988 est équivalent à celui de 1987" (*"The deficit of 1988 is equivalent to that of 1987"*) gives the following solution:

In order to solve possessive pronouns (*"theirs"*), concept types have to be found both for the possessed entity and for the owner, and the two have to be linked together with an appropriate conceptual relation.
Example: "Le garçon a fait ses devoirs et la fille a fait les siens" (*"The boy did his homework and the girl did hers"*)



A difficult problem is plural anaphors, since they may correspond to several entries in the stack (implicit coordination).
Example: "L'homme est arrivé avec la femme. Ils sont allés déjeuner" (*"The man arrived with the woman. They went to lunch"*).
In this case, we either search for a non-syntactically coordinated plural antecedent, or for a set of antecedents which have a common ancestor in the lattice, favoring elements which are already syntactically coordinated. This requires storing information concerning syntactic coordination of nouns in the stack.

Further to the problem of plural anaphors, it may happen that an anaphoric element is quantified ("those three persons", "the three of them", etc.). In such a case, and wherever applicable, the referents must be posted upwards until the target sum is reached.

In addition, in order to prevent the generation of absurd Conceptual Graphs, pragmatic rules based on syntax are applied. For the resolution of a given anaphor, this processing mainly consists in forbidding the stack entries whose syntactic structures in the sentence are incompatible with the syntactic structure of the anaphor [4]. (For example, a possessive determiner cannot refer to the possessed entity).

The semantic coherence checking algorithm

We have seen that the directed join and directed join management algorithms are useful in solving polysemy, incomplete syntactic information and anaphors. But this is not sufficient, because these problems may be inter-related. For example, we may have coordinated verbs with the same subject, this subject being polysemous, or even worse, a pronoun. We may also want to carry the

polysemous or pronoun subject of a main verb over to its infinitive complement.

In such cases, we have to check that the same solution for the subject has been taken everywhere in the resulting Conceptual Graph. This is the purpose of the semantic coherence checking algorithm. First, it ensures that different polysemous entries of one occurrence of a word in the sentence do not appear in the final result for the sentence. Secondly, it checks that the same solution for a pronoun has been selected throughout the processing. In cases of failure, the backtrack is activated. The backtrack on a pronoun is cut as soon as a satisfactory solution is found. This semantic coherence checking algorithm uses the projection algorithm.

Conclusion

Our prototype is still under development, and we do not claim to have solved all the ambiguities which can be found in Natural Language. However the Conceptual Graph model, along with the appropriate algorithms, has proven to be useful for the resolution of ambiguities which occur most often in real texts.

As far as the treatment of anaphors is concerned, we plan to extend it, as follows:

- The search for a referent will be applied to every proper noun and to every common noun preceded by a definite article, in order to introduce more cohesion in the representation of the text. (*"Mr John Akers, manager of IBM ... Mr Akers ... John ... the manager"*).
- But, in order to avoid wrong interpretations, the **local context** of a noun (i.e. its qualifiers) will then be stored in the stack of concepts and referents. This should also allow the solving of qualified noun ellipses (*"the red one"*), but the problem of the **scope** of a local context then arises.
- The solving of anaphors referring to **statements** is theoretically feasible with the Conceptual Graph model, by the use of **conceptual pointers** between PROPOSITIONS.
- The resolution of anaphors within long quotations, which introduce a context change, should take the context change into account.

Finally, some ambiguities may only be solved by the application of rules of common sense and/or deduction. A deductive component has been implemented in our system [6] [2]. This deductive component, applying appropriate production rules, should be invoked either during the text processing, or as post-processing on the set of Conceptual Graphs for a text.

References

[1] A. Berard-Dugourd, J.Fargues, M.C. Landau, **Natural Language Analysis Using Conceptual Graphs**, International Computer Science' 88, Hong-Kong, December 19-21, 1988.

[2] A. Berard-Dugourd, J. Fargues, M.C. Landau, J.P. Rogala, **Natural Language Information Retrieval from French Texts**, 3rd Workshop on Conceptual Graphs sponsored by AAAI, St-Paul, Minnesota, August 27, 1988

[3] P. Bosch, **Some Good Reasons for Shallow Pronoun Processing**, IBM Conference on Natural Language Processing, Thornwood, NY, October 24-26, 1988

[4] L. Danlos, **Génération automatique de textes en langues naturelles**, pp 191-208, Masson, Paris, 1985

[5] D.R. Dowty, R.E. Wall and S. Peters, **Introduction to Montague Semantics**, D. Reidel Publishing Company, Dordrecht (Holland), 1981.

[6] J. Fargues & al., **Conceptual Graphs for semantics and knowledge processing**, IBM Journal of Research and Development, Vol 30, No. 1, January 1986, pp 70-79.

[7] C.J. Fillmore, **The Case for Case**, Universals in Linguistic Theory, E. Bach and R.T Harms Eds, Holt, Rinehart & Winston, New York, 1968, pp 1-88.

[8] H. Kamp, **Events, Discourse Representations and Temporal References**, Langages 64, pp 39-64, Larousse Publishing Company, Paris, France, December 1981.

[9] R.C. Schank Ed: **Conceptual Information Processing**, North-Holland, Amsterdam, 1975.

[10] J.F. Sowa, **Conceptual Structures. Information Processing in Mind and Machine**, Addison Wesley Publishing Company, Reading, MA. 1984.

[11] **VM/Programming in Logic (VM/Prolog)**, IBM PO 5785-ABH, available through IBM branch offices.

[12] Y.A. Wilks, **Making preferences more active**, Artificial Intelligence, Vol 11, 3, 1978, pp 197-224.