# FRAGMENTATION AND PART OF SPEECH DISAMBIGUATION[1]

Jean-Louis Binot
B.I.M.
Kwikstraat, 4
B3078 Everberg, Belgium

***ABSTRACT***

That at least some syntax is necessary to support semantic processing is fairly obvious. To know exactly how much syntax is needed, however, and how and when to apply it, is still an open and crucial, albeit old, question. This paper discusses the solutions used in a semantic analyser of French called SABA, developed at the University of Liege, Belgium. Specifically, we shall argue in favor of the usefulness of two syntactic processes: fragmentation, which can be interleaved with semantic processing, and part-of-speech disambiguation, which can be performed as a preprocessing step.

## *1. Introduction*

The role of syntax is one of these issues in natural language processing which, albeit old and often (hotly) debated, have yet to receive a definitive answer. (Lytinen 86) distinguishes two approaches to NL processing. Followers of the "modular" approach believe usually in the autonomy of syntax and in the usefulness and cost-effectiveness of a purely syntactic stage of processing. Results of this approach include the development of new grammatical formalisms (Weir et al. 86) (Ristad 86), and of large syntactic grammars (Jensen et al. 86).

Followers of the "integrated" approach, on the contrary, believe that semantics should be used as soon as possible in the parsing process. An "integrated" system would have no discernable stages of parsing, and would build directly a meaning representation without building an intermediate syntactic structure. How much syntax is needed to support this semantic processing, however, and how should the integration between syntax and semantics be done are still open and crucial questions. Some integrated systems, such as IPP (Schank et al. 80) and Wilks' Preference Semantics system (Wilks 75), were trying to reduce the role of syntax as much as possible. Lytinen proposes a more moderate option in which separate syntactic and semantic rules are dynamically combined at parsing time. Another kind of integration is used in (Boguraev 79), where an ATN is combined with Wilks' style semantic procedures. And, lastly, one might consider that unification-based grammars (Shieber 86) offer yet another approach where syntactic and semantic constraints can be specified simultaneously in functional structures and satisfied in parallel.

---

[1] The research presented in this paper was entirely performed while the author was working at the Computer Sciences department of University of Liege, Belgium.

In this paper, we wish to present our arguments in favors of integration, and then to discuss two specific technical proposals. Our general position can be stated as follows:

1. That at least **some** form of syntax is necessary for natural language processing should be by now fairly obvious and should need no further argumentation.

2. Syntax, however, is not a goal per se. The basic goal of NLP, at least from the point of view of AI, is to give a computer a way to "understand" natural language input, and this clearly requires a semantic component. The utility or necessity of syntax should only be evaluated in the light of the help it can provide to this semantic component.

3. Grammaticality is not an essential issue, except in language generation and in specific applications like CRITIQUE (Jensen et al. 86), where the purpose is to correct the syntax and the style of a writer. For the general task of understanding, achieving comprehension, even in the face of incorrect or unusual input, is more important than enforcing some grammatical standards. And we believe that robustness is more easily achieved in the context of a semantic system than in the predictive paradigm of the grammatical approach.

If we want to avoid the use of a full scale grammar, the syntactic processes necessary to support the semantic module must be implemented by special dedicated procedures. This paper describe the solutions used in a semantic analyser of French called SABA, developed at the Computer Sciences department of University of Liege, Belgium. Specifically, we shall argue in favor of two syntactic processes: fragmentation, which can be interleaved with semantic processing, and part of speech disambiguation, which is usefully performed in a preprocessing step. We shall start by a brief description of the SABA system.

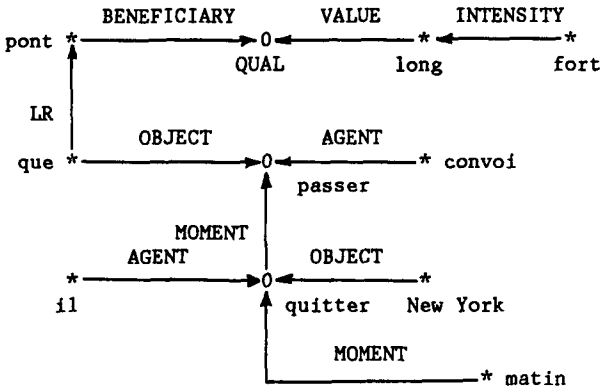
## *2. Overview of the SABA system.*

SABA ("Semantic Analyser, Backward Approach", (Binot, 1985), (Binot et al., 1986)) is a **robust** and **portable semantic** parser of written French sentences. A prototype of this parser is running in MACLISP and in ZETALISP; it has been tested successfully on a corpus of about 125 French sentences. This system is not based on a French grammar, but on semantic procedures which, with some syntactic support, build directly a semantic dependency graph from the natural language input. The following example is typical of the level of complexity that can be handled by the system:

(1) *Le pont que le convoi a passe quand il a quitte New York ce matin etait fort long.*
(The bridge that the convoy crossed when it left New York this morning was very long.)

To allow for portability, the SABA parser translates its natural language input into an "intermediate" semantic network formalism called SF (for "Sentence Formalism"), presented in details in (Binot, 1984, 1985). Before generating the SF output, SABA builds a simplified semantic graph expressing all the semantic dependencies established between the meaningful terms of the sentence. The graph established for sentence (1) is shown in figure (2).

(2)




These kinds of dependencies are established by using the "dual frames" method described in (Binot and Ribbens 86). Dual frames is a general method for establishing binary semantic dependencies between all possible types of meaningfull terms. This method supports also a hierarchy of semantic classes and an inheritance mechanism allowing the designer to specify generic semantic frames at a general level. However, we are not concerned here by the specifics of a particular semantic method, but by the kind of syntactic support necessary to establish such dependencies (or, to put it another way, by the kind of syntactic support needed to identify accurately the arguments filling the role slots of various meaningfull terms).

# 3. Fragmentation

## 3.1 General discussion

Consider again sentence (1) and suppose that a purely semantic system were to understand it by establishing semantic dependencies between words. There would be no reason for such a system to refrain from attempting to connect "was long" to "convoy", for example. And, if the attempt is made, no amount of semantic or pragmatic knowledge will be able to prevent the connection, which is perfectly valid as such. Note also that a simple proximity principle would not work in this case.

Thus, any natural language processing system must take into account, in some way, the **structure** of a sentence. However, we don't necessarily need to build an intermediate syntactic structure, such as a parse tree, showing the detailed "phrase structure" of the input. The most crucial structural information needed for an accurate semantic processing concerns "boundaries" across which semantic processing should not be allowed to relate words. These boundaries can be identified by a fragmentation process which will cut a sentence into useful fragments by looking for specific types of words.

Except maybe in Wilks' system fragmentation has not received the attention it deserves as a faster alternative to full syntactic parsing. Wilks' fragmentation process, however, was by his own admission too simple. In his system, fragmentation was performed only once as a preprocessing step, and was designed around the size of his notion of "template". Both of these characteristics, we think, give rise to problems.

Performing fragmentation as a single preprocessing step is obviously insufficient for garden path sentences and for all the structural ambiguities that cannot be solved without the help of the semantic module. Although Wilks said something about involving some semantic processing at the fragmentation stage, notably for handling the ambiguity about "that", he never presented, to our knowledge, a systematic procedure to integrate fragmentation and semantics. On the other hand, we believe that template sized fragments are more troublesome and less usefull than clause sized fragments. Even in straightforward active declarative sentences, two distinct mechanisms must be provided to establish semantic dependencies in Wilks system: template matching, which identifies "agent-action-object" triples, and paraplates, which are used to tie these templates together. A prepositional phrase constitutes a separate template. One problem with that approach is that in sentences such as "The old man / in the corner / left", fragmented by Wilks as shown by the "/", the agent ends up in a different fragment than the action and an additionnal step will be required to relate the two. The same problem seems to arise in passive structures ("John is loved / by Mary"). To avoid these kinds of problems, we decided to use clause sized fragments and to establish semantic dependencies directly at the clause level.

A third difference between the two approaches is that, while Wilks never provided a systematic method to solve part of speech ambiguities, SABA makes use of a part of speech disambiguation preprocessor, which will be described in the second part of this paper. This module being applied before fragmentation, we shall assume in the following discussion of the fragmentation mechanism that each word has a single part of speech.

## 3.2 The fragmentation mechanism.

We have implemented in the SABA system a fragmentation mechanism which uses the clause as the fundamental fragmentation unit and which is repetitively applied and interleaved with the semantic processing. We start by presenting the basic algorithm, then, in the next sections, we shall discuss some more difficult problems and show how the introduction of two additionnal mechanisms, ejection and backtracking, can solve them.

### Fragmentation algorithm:

Repeat the following until success or dead end

1. Fragment the sentence into clauses;
2. Select the innermost clause;
3. Process the selected clause, which includes:
   a. The fragmentation of the clause into groups;
   b. The establishement of semantic dependancies inside each group;
   c. The establishement of semantic dependancies at the clause level;
4. If the processing is successfull, erase the text of the clause from the input and replace it by a special non terminal symbol.

This algorithm follows a bottom-up strategy in which the innermost clause (the most dependent one) is always processed first. Ties are resolved by a left to right preference rule. The special symbols used in step 4 are PP ("Proposition Principale") for a main clause, PR for a relative clause, PC for a conjunctive subordinate clause and PINF for an infinitive clause. Participe clauses are processed as special kinds of relatives, as we explain in section 4.2.

Success in the above algorithm means that the input has been reduced to the PP symbol or to a string of such symbols and conjunctions. A dead end is reached if fragmentation can find no new clause or if the selected clause cannot be processed. What happens then will be discussed in the next sections.

As can be seen in the above algorithm, fragmentation in SABA is in fact a two level process: sentences are fragmented into clauses and clauses into groups. Fragmentation into groups, wich gives far less problems than fragmentation into clauses, will not be discussed at all in this paper.

Fragmentation of a sentence into clauses proceeds by extending to the left and to the right of each verb[2] and checking each encountered word looking for clause delimiters. The checks are performed by heuristic rules based on the part of speech of each word. Other rules will then look at the delimiters to find the innermost clause.

The rules checking if a given word is a delimiter are given below. The term "explicit clause boundaries" used in the rules denotes the following kinds of words: relative or interrogative pronouns, relative or interrogative adjectives, subordinate conjunctions and coordinate conjunctions. Coordinate conjunctions, which raise special problems, will not be discussed before section 3.5.

### *Clause fragmentation rules.*

1. Explicit clause boundaries other than coordinate conjunctions are always clause delimiters; they are included in the clause on the left and excluded on the right.[3]
2. The special symbols PR, PC, PINF are never clause delimiters.
3. Sentence boundaries are always clause delimiters.
4. Another verb and the symbol PP are always clause delimiters, and are always excluded from the clause.
5. Negation particles ("ne", "n'") are considered as (excluded) clause delimiters when expanding to the right of the verb of the clause.

Rules 1 to 4 are rather immediate. Rule 5 takes into account the fact that negation particles in French are always placed before the negated verb.

The basic clause selection rules (for choosing the innermost clause) are equally simple. A clause is subordinate if its left bound is a relative or interrogative pronoun (or adjective), or a subordinate conjunction, or if its verb is an infinitive. A clause is said to be free (meaning that it is not qualified by other subordinate clauses which should be processed first) if its right bound is not one of these terms. The leftmost free and subordinate clause, or, if none, the leftmost free clause will be chosen.

Let us illustrate the effect of the above rules on example (1). The figure (3) below shows the successive states of the input text. In each state, the last fragmentation result is indicated by underlining the identified clauses. The semantic processing of the innermost clause selected at each step leads to the building of the corresponding part of the graph of figure (2).

(3) *Le pont que le convoi a passe quand il a quitte New-York ce matin etait fort long.*

*Le pont que le convoi a passe PC etait fort long.*

*Le pont PR etait fort long.*

*PP*

As can be seen, a single fragmentation pass will often yield imperfect results. There will be **holes** (sentence fragments which are not included in any clause, like "Le pont" in the first two steps) and **overlappings** (fragments which could be included in two clauses, like "New-York ce matin" in the first step). This is where the repetitive nature of the fragmentation process comes into play. Successive erasing of the innermost clauses from the input text, once they have been processed by the semantic module, will gradually cause the holes to disappear, and thus reveal the content of the main clause(s). Terms in overlapping areas will be automatically tried first in the innermost clause to which they could belong, in effect implementing a kind of deepest attachment preference. What happens when that first try is semantically inacceptable is discussed in the next section. Another interesting feature of the bottom-up algorithm is that the special symbol representing a processed subordinate clause will be naturally included, in later fragmentation steps, in the clause qualified by this subordinate, thus permitting to process correctly inter clause dependencies.

### *3.3 The ejection mechanism.*

A first class of problems for which the above fragmentation algorithm is not sufficient concerns cases when the deepest attachment preference fails. This problem occurs typically when a clause has no explicit clause boundary on one side, as in the examples (4) and (5) below:

(4) *J'aime l'homme que je presente a mon pere.*
(I love the man whom I introduce to my father)

(5) *Je presente l'homme que j'aime a mon pere.*
(I introduce the man whom I love to my father)

In both cases the relative clause has no explicit right boundary, and the attachment problem concerns the group "a mon pere". The fragmentation result (shown by underlines) will in both cases include this group in the relative clause, which is wrong for (5). In such cases, the fragmentation will be automatically corrected, after the semantic processing of the relative clause, by a "right-ejection" mechanism:

### *Right ejection mechanism*

> If a group G on the right of the verb remains unconnected after the semantic processing of a clause, and if there is no other term on the right of G which has been connected to a term on its left, then G and all terms on its right will be excluded from the current clause.

In the case of example (5), assuming reasonnably that no semantic dependency can be established between "aime" and "a mon pere", this last group will be ejected from the relative clause, giving the situation shown in (6):

(6) *Je presente l'homme que j'aime a mon pere.*

Since fragmentation is interleaved with the semantic processing, the next fragmentation step will automatically pick up the discarded term after the processing of the relative clause, and insert it at the correct level:

(7) *Je presente l'homme PR a mon pere*

The same mechanism applies to overlapping cases, such as in example (8):

(8) *L'homme que j'ai rencontre sur la place m'a offert un cafe.*
(The man that I met in the square bought me a coffee)

Here, two groups appear in the overlapping fragment. The first one, "sur la place" ("on the square"), can easily be connected to the relative verb (as a location argument) and will remain in the relative clause. The second, "m" ("me") cannot be connected to "rencontre" ("met"), the object slot of that verb being already filled by the relative pronoun "que". "m" will thus be ejected from the relative clause, and included correctly in the main clause during the next fragmentation step.

It is worth mentionning that this mechanism involves no backtracking and is extremely cheap in computational ressources. The only processing required is the displacement of the right clause boundary before erasing the text of the processed clause.

---

2 Except auxiliaries that are part of a compound verbal form.

3 If the left clause bound is a relative pronoun preceeded by a preposition, the preposition will also be included in the clause.

### *3.4. Infinitive clauses and backtracking.*

Infinitive clauses without an explicit left boundary (such as a subordinate conjunction) give rise to several interesting problems concerning both fragmentation itself and the selection of the innermost clause. Consider the following examples:

(9) *J'irai ce soir a Paris voir l'exposition.*
(I will go this evening to Paris to see the exposition)

(10) *Je n'ai jamais vu Jacques travailler.*
(I never saw Jacques working)

In both cases, there is an attachment problem for the terms in the overlapping area. In (9), all the terms in that area belong to the relative clause, while in (10) Jacques is the subject of the infinitive clause. One might want to define here a "left-ejection" mechanism similar to the one described in the last section; however it would almost never work properly. Indeed, if terms such as "this evening" or "to Paris" are tried in the infinitive clause first, there would be no reason to reject them during the semantic processing of that clause, and they will never be ejected. Things work out better if we try first the terms in balance in the main clause. This choice will be wrong when one of these terms is in fact the subject of the infinitive verb; but in that case, as we shall see, this term will conflict with the infinitive verb for filling the OBJECT slot of the main verb, and the system will have a reason to reject the wrong choice. Accordingly, we apply the following strategy:

1. try first to place the terms of the overlapping area in the main clause; in effect, this consists in preventing the infinitive clause to extend to the left of its verb;
2. if the choice made at point 1 fails, use a backtracking mechanism that will restore the proper state of the analysis and try to extend, one group at a time, the left bound of the infinitive clause.

With this strategy, (9) will be processed correctly at the first try. (10) will lead to the following (erroneous) state of the analysis:

(11) *Je n'ai jamais vu Jacques PINF,*

where "Jacques" and PINF compete for the object slot of the main verb. The term PINF will then be ejected by the mechanism of the last section, giving the following state:

(12) *PP PINF*

This is a dead end state, since the sentence is not reduced to a PP symbol, and yet no further clause to process can be found. The backtracking mechanism will then restore the state shown in (10) with the following fragmentation, which leads to a successfull analysis:

(13) *Je n'ai jamais vu, Jacques travailler,*

Infinitive clauses raise also problems concerning the selection of the innermost clause. Consider the following examples:

(14) *J'ai vu un homme, qui voulait dormir sur le trottoir,*
(I saw a man who wanted to sleep on the street)

(15) *J'ai vu un homme, qui avait bu, dormir sur le trottoir,*
(I saw a man who was drunk sleep on the street)

In both cases, the selection rules will choose to process the infinitive clause first. This choice is wrong for (15): if the relative relative clause is not processed first, its presence will prevent the system to find out that the group "un homme" is in fact the subject of the infinitive clause. Processing the infinitive first, the system will reach a dead end after the following steps:

(16) *J'ai vu un homme, qui avait bu PINF,* (ejection of PINF)

*J'ai vu un homme PR PINF,* (ejection of PINF)

*PP PINF* (dead end)

This problem is again handled by backtracking. Let us note first that the problem arises only when the subject of the infinitive verb is separated from that verb by a relative clause. In such a case, the system will try to process the infinitive first, but will save the current state of the analysis so that it can later backtrack and process the relative first. In the case of our example, backtracking to (15) from the dead end state in (16), and processing the relative clause first, we obtain a correct analysis, as shown in (17):

(17) *J'ai vu un homme PR, dormir sur le trottoir,*

*J'ai vu PINF,*

### *3.5 Coordinate conjunctions*

Fragmenting sentences with coordinate conjunctions requires to make a decision regarding the scope of these conjunctions; specifically we need to distinguish between the conjunctions which coordinate clauses and the ones which coordinate groups inside a same clause. The following rules are used:

#### *Clause delimiter rules for coordinate conjunctions*

1. If the word to the right of the conjunction is a right delimiter, or if next word in the current direction is the special symbol PP, the conjunction is taken as delimiter (excluded).
2. If the next clause delimiter in the current direction is an explicit clause boundary or a sentence boundary, the conjunction is not taken as delimiter.
3. Otherwise **choose** to consider first the conjunction as a delimiter (excluded); this choice can be undone by backtracking.

Rule 1 is based on the fact that there must always be at least one conjunct to each side of a conjunction. If a delimiter is found immediately to the right, then the conjunction must connect clauses. The same is true if the conjunction is adjacent to the PP symbol. The following example illustrates the use of this rule:

(18) *J'aime les chiens, qui m'obeissent, et qui ne mordent pas,*
(I love the dogs which obey me and which do not bite)

*J'aime les chiens PR, et qui ne mordent pas,*

*J'aime les chiens PR et PR,*

*PP*

If the next delimitor is an explicit clause boundary, then there is no verb between the conjunction and this delimiter, and thus the conjuncts cannot be clauses. This fact, captured by rule 2, can be illustrated by the following example:

(19) *J'ai appris que les pommes et les poires etaient cheres,*
(I learned that apples and pears were expensive)

*J'ai appris PC*

Finaly, if the next delimitor is a verb, the scope ambiguity cannot be resolved at this stage. The conjunction could be a clause delimiter, as in (20), or not, as in (21):

(20) *Connors a vaincu Lendl et McEnroe a vaincu Connors.*
(Connors defeated Lendl and McEnroe defeated Connors)

(21) *Les hommes qui aiment les pommes et les poires aiment aussi les oranges.*
(People who like apples and pears like also oranges)

In such cases, the system will choose to take the conjunction as a delimiter, and record the state of the analysis, so that the choice can be modified by backtracking. The choice will be correct for sentence (20). For sentence (21), the incorrect choice will lead to a dead end, as shown in (22), when the semantic module will try to coordinate "hommes" and "poires" as agents of "aiment". Backtracking to the choice point, followed by a new fragmentation, leads to the correct solution.

(22) *Les hommes qui aiment les pommes et les poires aiment aussi les oranges.*

*Les hommes PR et les poires aiment aussi les oranges.*

*BACKTRACKING*

*Les hommes qui aiment les pommes et les poires aiment aussi les oranges.*

*Les hommes PR aiment aussi les oranges.*

*PP*

# 4. Part of speech disambiguation

## 4.1 General discussion

Many lexically ambiguous words can have different parts of speech (hereafter POS). The following table enumerates the main POS ambiguities for example (1).

**Le** (occurs twice): article or personal pronoun (the, him, it)
**que**: subordinate conjunction, relative or interrogative pronoun, particle (that, which, what, than)
**quand**: subordinate conjunction or adverb (when)
**fort**: noun or adverb (castle, very).

The ambiguity problem is further compounded by an accentuation problem. "Passe", third person of the present of the indicative of the verb "passer", is quite different in French from "passe", past participle of the same verb.[4] Similarly, "a", indicative of avoir ("to have"), has nothing to do with the preposition "a". However, forgetting an accent is one of the most common spelling mistakes. A robust system such as SABA must consider words such as "a", "passe" and "quitte" as ambiguous. This would give at least 1024 possible POS combinations for example (1)!

Part of speech ambiguity is, of course, part of the more general problem of lexical ambiguity. Thus, one could argue that it doesn't need an independent solution. However, in the context of a fragmentation system such as the one presented here, a POS disambiguation preprocessor is necessary. To give a simple example, the relative pronoun and subordinate conjunction senses of "que" are clause boundaries, while the (comparative or restrictive) particle sense is not. Many other problems of semantic processing need a prior decision regarding the POS of the words involved. Thus the French word "or" can be a noun ("gold"), and as such can fill a semantic role slot of some verb, or can be a coordinate conjunction ("however"); "le" can be pronoun ("him", "it") and as such induce a search for a pronoun reference, or can be a determiner ("the"). Many other examples could easily be found.

Other works have already investigated the usefullness of a POS. disambiguation preprocessor, but for syntactic parsers. (Klein and Simmons 63) presented very early a table based system for English where the emphasis was on the capability to classify "unknown words", and thereby to reduce the size of the dictionnary. Much more recently, (Merle 82) described a rule based POS disambiguator for French, its main objective being a gain of performance obtained by the reduction of combinatorial explosion during syntatic parsing. Merle's rules, however, were rather unwieldy for two reasons:

1. each rule must make a final decision regarding the POS of one word; the designer must ensure himself the absence of contradictions between the rules.
2. The rules permitted only to test for fixed patterns in the input.

In contrast to that, we have developped a method permitting the use of cumulative rules and providing the possibility to test variable patterns through the use of a search function.

## 4.2. The part of speech preprocessor for the SABA system.

We have developped a part of speech disambiguation preprocessor for French, which is used as the first stage of the SABA system. This preprocessor consists of heuristic rules which are applied to each word in order to assign to every possible part of speech a certainty factor. The different combinations of possible parts of speechs are then tried in decreasing order of likeliness.

The heuristic rules are based on the well known fact that it is not necessary to scan the entire sentence to choose correctly the appropriate part of speech for most words. The "local context" (i.e. the few surrounding words) proves often enough to provide an accurate indication. Thus, if a word like "passe" is closely preceeded by an auxiliary, it is almost certainly a participe. As another example, "fort", if closely preceeded by a determiner, is more likely to be a noun than an adverb.

We have captured such insights into heuristic rules which assign to each possible part of speech a certainty factor, according to the local context. Two of these rules, relating to the examples just mentionned, are given in natural language form below:

### *Rule 2*

If the current word can be a past participe and has other possible POS, then

1. If the current word is preceeded by a word that could be an auxiliary, and is only separated from that word by words that could be adverbs, personal pronouns or particles, then
   past participle CF = 0.7; other possibles POS CF = 0.3;
2. Else:
   relative participe[5] CF = 0.7; other possible POS CF = 0.3.

### *Rule 5*

If the current word can be a noun and has other possible POS, then

1. If it is preceeded by a word that could be a determiner, and is only separated from it by words that could be adjectives or adverbs, then
   noun CF = 0.9; other possible POS CF = 0.1;
2. else:
   noun CF = 0.4; other possible POS CF = 0.6;

---

[4] Verb mood ambiguities can usefully be considered at the same level as POS ambiguities.

[5] We distinguish between a participe used in a complex verbal form and a participe clause, as in "The man defeated by Connors was ill". In the later case, the participe will receive a POS called PPAREL ("relative participe") because the participe clause is then processed exactly like a relative clause: in fact, when the POS PPAREL is assigned to a participe, a relative pronoun is inserted just before it.

These rules need several comments:

1. Each rule can be seen as a production rule with a condition and an action. The condition is the clause starting with the first "if" of the rule; if it is not satisfied, this particular rule is not applied to the current word. The action is often itself a conditionnal statement, each branch of which must include a certainty factor assigment statement.

2. The certainty factors that we are using range from 0 (absolute uncertainty) to 1 (absolute certainty). They can be compared to the belief factors used in the MYCIN system (Shortliffe 76).

3. The application of any rule must result in one assigment of certainty factors to all possible POS of the current word. However, a given word could possess other possible POS than those that need to be explicitly mentionned in a given rule. These are refered to by the formal expression "other possible parts of speech".

4. The intermediate words tested by a rule can also have several possible parts of speech. The expression "if such word could be of part of speech x" denotes a test bearing on all possible parts of speech of that word.

5. We must be able to specify rules at varying levels of details. Sometimes, we will need to test if a word is a personal pronoun; at another time, knowing that it is a pronoun of any kind is sufficient. The system offers the possibility to specify a hierarchy of parts of speech, which is taken into account by the rules.

The part of speech disambiguation preprocessor works in the following way. It processes successively all the words of the input. For each word, it checks the conditions of all rules and fires all applicable rules. If several rules are applied to a same word, certainty factors are combined by the following formula:

$$CF = 1 - ((1 - CF1)*(1 - CF2))$$

where CF1 and CF2 are the certainty factors to be combined. When this is done, possible POS combinations are ordered by decreasing order of likeliness. The likeliness of a combination is simply defined as the product of the certainty factors of the parts of speech included in that combination.

Although each rule is considered for every word, the resulting process is very fast. The first reason for that is that there are very few rules: 14 in the current implementation. This is nothing compared to the size of the rule base needed for a large grammar, and yet these few rules are sufficient to choose the correct POS at the first try in more than 80% of our test sentences. The second reason is that each rule is garded by a short, easy to check and very selective condition, so that most of the rules are immediately discarded for a given word.

### *4.3. Implementation of the rules.*

The rules are implemented in a "semi-declarative" way: they can be specified separately, each being described as a condition-action pair. However, both condition and action can be any evaluable LISP form. In order to ease the task of rule specification, we have defined a set of primitive operations. The figure (23) gives the formal specification of Rule 2.

HOMOGRAPH checks if a word has more than one possible parts of speech. POSSIBLE-STYPE checks if the specified part of speech is one of the possible parts of speech of the word. DEFINE-PTYPELIST assigns to each part of speech of the word a specific certainty factor. EXISTWORD, lastly, is a highly parametered function performing searches in the input sentence. Its parameters are:

1. POSITION: the starting word for the search;
2. DIRECTION: the direction of the search (LEFT or RIGHT);
3. LIMIT: the ending word, beyond which the search should be stopped;
4. GOAL-NAMES: admissible names for the target word
5. GOAL-TYPES: admissibles parts of speech for the target word;
6. GOAL-CLASSES: admissible semantic classes for the target word;
7. BETWEEN-NAMES: admissible names for intermediate words
8. BETWEEN-TYPES: admissible parts of speech for intermediate words;
9. BETWEEN-CLASSES: admissible semantic classes for intermediate words;
10. EXCLUDED-NAMES: excluded names for intermediate words;
11. EXCLUDED-TYPES: excluded parts of speech for intermediate words;
12. EXCLUDED-CLASSES: excluded semantic classes for intermediate words.

Parameters 3 through 12 are optional. The default value for LIMIT is the sentence boundary. The default value for parameters 4 through 9 is "(ALL)", denoting that all values are accepted. The default value for parameters 10 through 12 is NIL (no value is excluded).

## *5. Results and conclusions*

We have presented two syntactic processes which offer useful and necessary support for semantic processing. syntactic parser. Both are based on simple heuristic rules assisted by a backtracking mechanism. Both have been implemented in the SABA system and tested on a corpus of about 125 sentences. Less than 5% of these required a backtracking of the fragmentation process. Since we tried to characterize precisely the situations in which a backtracking could arise, in most sentences there is not only no backtracking, but also no bookkeeping of the intermediate steps.

(23)

```
(ADD-SYNT-RULE R R2
   Condition
   (AND (POSSIBLE-STYPE WORD) 'PPA) (HOMOGRAPH WORD))
   Action
   (COND ((EXISTWORD position (LEFT WORD)
                     direction 'LEFT
                     goal-classes '(AUX)
                     between-types '(PR ADV PT))
          (DEFINE-PTYPELIST WORD '((PPA . .7)(OTHERS . .3))))
         (T (DEFINE-PTYPELIST WORD '((PPAREL . .7)(OTHERS . .3))))))
```

As for the part of speech disambiguation preprocessor, the 14 rules that we implemented were sufficient to make the right choice in more than 80% of the cases. The very small size of this preprocessor is an important advantage if we think at the high human and computational costs involved in developing and using large size grammars.

Although the specific rules that we implemented were designed for French, we believe that the approach could be applied to other languages as well.

## *ACKNOWLEDGMENTS*

Thanks are due to Professor D. Ribbens for his numerous helpfull comments and for his active support.

## *REFERENCES*

1. Binot, J-L. 1984. A Set-oriented semantic network formalism for the representation of sentence meaning. In *Proc. ECAI84*, Pisa, September 1984.
2. Binot, J-L. 1985. *SABA: vers un systeme portable d'analyse du francais ecrit.* Ph.D. dissertation, University of Liege, Belgium.
3. Binot J-L. and Ribbens D. 1986. Dual frames: a new tool for semantic parsing. In *Proc. AAAI86*, Philadelphia, August 1986.
4. Binot J-L, Gailly P-J. and Ribbens D. 1986. Elements d'une interface portable et robuste pour le francais ecrit. In *Proc. Huitiemes Journees de l'Informatique Francophone*, Grenoble, January 1986.
5. Boguraev B.K. 1979. *Automatic resolution of linguistic ambiguities.* Ph.D. thesis, University of Cambridge, England, 1979.
6. Jensen K., Heidorn G.E., Richardson S. and Haas N., PLNLP, PEG and CRITIQUE: three contributions to computing in the Humanities. In *Proc. of the conf. on Computers and Humanities*, Toronto, April 1986.
7. Klein S. and Simmons R.F. A computational approach to grammatical coding of English words. *Journal of the ACM*. 10, March 1963.
8. Lytinen S.L. 1986. Dynamically combining syntax and semantics in natural language processing. In *Proc. of AAAI86*, Philadelphia, August 1986.
9. Merle A. 1982. *Un analyseur presyntaxique pour la levee des ambiguites dans des documents ecrits en langue naturelle: application a l'indexation automatique".*Ph.D. thesis, Institut National Polytechnique de Grenoble.
10. Ristad E.. 1986. Defining natural language grammars in GPSG. In *Proc. of the 24th meeting of the ACL*, New-York, June 1986.
11. Schank R.C., Leibowitz M. and Birnbaum L. 1980. An integrated understander. In *Journal of the ACL*, 6:1.
12. Shieber S. 1986. *An introduction to unification-based approaches to grammar*, University of Chicago Press.
13. Shortliffe E.H. 1976. *Computer-based medical consultation: MYCIN* Elsevier.
14. Weir D.J., Vijay-Shanker K. and Joshi A.K. 1986. The relationship between Tree adjoining grammars and head grammars. In *Proc. of the 24th meeting of the ACL*, New-York, June 1986.
15. Wilks Y. 1975. An intelligent analyser and understander of English. *CACM* 18:5, May 1975.