A DETERMINISTIC SYNTACTIC-SEMANTIC PARSER

Gerard SABAH, Mohamed RADY

GR22, PARIS VI, 4 Place Jussieu, 75230 PARIS CEDEX 5

ABSTRACT

We consider that we have made a decisive step towards determinism in parsing. We agree with Winograd's hesitation to evaluate the determinism hypothesis as formulated by Marcus. However, this does not make us doubt about the possibility of determinism; on the contrary, we examined not only how to improve over Marcus, but also the historical reasons of the non-determinism of most systems.

Our improvements are based on two principles: syntactic-semantic integration, and quasi-simultaneousness. The first means that there is no such thing as "the autonomy of syntax" (Marcus); so, we agree with Schank and, further, we showed that local semantic ambiguities could be solved deterministically (Marcus (ch.10) claims that these ambiguities need parallel processing). The second permits the processing of structures too difficult for PARSIFAL e.g. locally ambiguous PP attachments.

Detailed examples support our proposals.

I POSSIBLE FACTORS OF NON-DETERMINISM

With the notable exception of Marcus, already mentioned , most "parser makers" seem to agree on having a non-deterministic strategy (backtracking or parallelism). Schank (et al.) have always perceived the inadequacy of those strategies, and practised the integration of syntax and semantics, but they have conserved a top-down approach ('prediction' cf Wilks)

We shall try to show the importance of 4 factors of non-determinism which are not related to NL ambiguity: the architecture of systems, rules and interpreter, stack mecanism, word by word processing.

A. Architecture of systems

Decision elements for a parser are of 4 kinds: morphosyntactic, semantic, pragmatic and contextual. So, it is obvious that considering only one of them brings about non-determinism, independently of NL. itself. However, even when a system uses all of this information, non-determinism may appear due to an artificial separation between modules (eg syntactic & semantic). Some non-determinism remains even in the most sophisticated modular systems and that's because:
-one module is dominating the others (as in SHRDLU)
-too many interaction points exist
-the non-det. may find refuge in one of the modules (as in PARSIFAL, the semantic one).

B. Representation for a grammar and interpretor

The interpreter has usually either a top-down or a bottom-up strategy, but most frequently top-down. But a relative clause, for instance, should rather be parsed bottom-up, in order to be attached deterministically. In theory, the Marcus parser seems to permit this strategy ("attention shifting rules"). But the grammar Marcus actually wrote does not take advantage of this facility \nd relative clauses are also parsed top-down. From his theoretical work , we still have the confirmation of following point:

A completely top-down or completely bottom-up mecanism must parse non deterministically some sentences that would not give the same trouble to a mixed strategy. (In the field of programming languages, such a mixed strategy has already been applied, Earley)

Now, another important point: in a traditional declarative system (ATN or PSG), the rules are independent of the interpreter. So, when a conflictarises between two valid rules, it is not possible for the interpreter to make a decision. Hence,one has to limit the role of the interpreter and the rules must be non ambiguous (their condition part being as detailed as possible).

C. The stack

One essential step in NL processing is to gene rate from a text a so-called meaning representation. Concretely, this representation is an interconnected graph of internal structures, built step by step. At a certain step, processing may be interrupted in order to start working on some other structure. This makes necessary to store a trace of the interrupted structure. In general, this is done by means of a stack.

Each new interruption makes the structure at the top of the stack inaccessible. This implies two crucial problems: when to start building a certain structure, and then, how does one know it is finished.

A complication may appear when a certain word functions as a starter for several structures. Without elaborating on this problem, we can say it has to do with top-down or bottom-up interpretation of the rules. Even harder is the decision of ending a structure: take the ambiguous sentence "I saw the man with a telescope". The difficulty comes from having to decide either to end the direct object after 'man' or to continue it by a NP modifier.

That proves a certain inadequacy of the stack mecanism. One could think of a list instead of a stack, in order to solve this problem (cf Schank); however, when there are several modifiers in succession, attachment becomes so complicated that this necessary improvement is by no meansaufficient: it seems

necessary to make use of a quasi-simultaneous mecanism like ours (cf § II-C).

D. Word by word processing

Most parsers "read" one word at a time, integrating it into a representation before any processing of the next word. This strategy could be termed "hie et nunc" and opposed to the famous "wait and see" strategy. The obvious advantage of the second one is that the parser may delay a difficult decision until it becomes easier. In particular, many potential ambiguities disappear as soon as what canes next is known.

E. The design of our parser will result from the solution *of* the four above problems:
-Integration of syntax and semantics
-Top-down and bottom-up rules
-Quasi simultaneous elaboration of structures
-Look ahead facility.

II ORIGINAL FEATURES OF THE PARSER

A. Formal rules

The final aim of our parser is to generate a semantic representation. But, it also generate a provisional syntactic representation based on systemic grammars (Halliday), in order to have reference either to semantic or syntactic information when its own functioning makes it necessary.

The "condition part" of a rule considers syntactic and semantic features of the current structure but also of whatever comes next, words *or* structures.

The "action part" can do several things:
-generate and develop a syntactic structure
-develop a semantic description of a syntactic structure
-attach a structure to another
-detach or resume a structure
-activate or deactivate other rules ("meta-actions")

Both the actions and the conditions allow a fusion of syntax and semantics,and may call domain specific procedures (eg for inferences,...). The rules are arranged in overlapping packets. Each packet corresponds to a well defined kind of situation (top-down definition). Interruption rules (§II-B2) allow a bottom-up mecanlsm.

Take a rule activated by 'to cure',for example
IF .the verb is followed by a NP & .the semantic description (SD.) of this NP is compatible with the description of the 'objective' case of the verb
THEN .attach NP to verb as direct object & .attach SD of NP to SD of verb with label 'objective'.

This rule is OK for "the doctor cures the flu", but does not apply for "the doctor cures my daughter". Such cases are to be treated by a rule similar to the preceding one, replacing 'objective' by 'dative'.

In an ambiguous case like "the doctor cures it" the two rules by themselves do not resolve the semantic problem at this level, but, having tried them, we may find that the search for the antecedent of 'it' is made easier: it is possible to start inferences from the conditions of these non applied rules.

B. Look ahead

This notion is well known to people who write compilers (cf Ako & Ullman quoted by Marcus). Marcus was the first to apply it to the syntax of NL and we tried to go over to semantics.

Materially, look ahead means that the parser has a certain field of vision, including the current structure and the three next ones, which may be non terminal (3 is an experimental choice, cf Marcus for the necessity of a limitation).

1. Look ahead and semantic ambiguity

To show the connection between look ahead and choice of a meaning representation, let us consider the two sentences: "John makes a program in Simula" and "John makes a program conscious".

Each sentence corresponds to a different meaning of 'to make[1] and it is necessary to know what comes after the first syntactic object to resolve the ambiguity. So, this part of the sentence should be processed before choosing a meaning for the verb. The look ahead facility permits to examine simultaneously the 1st and 2nd complements. Looking at this latter, one can decide between 'makel' and 'make2' and correctly attach the semantic structures.
In case of truly ambiguous sentences, what remains arc syntactic and semantic preference heuristics.

2. Interruption rules

We want to make it. possible for a structure to wait completion of another one and resume then.
If SO, SI, S2, S3 are the viewed structures (the current one plus the 3 in the field of vision) and RI the rule: IF SI is a NP & S2 an adjective & S3 a PP, THEN ...; and, if this rule being active the field of vision is:
S0=(VP(...(Verb:MAKE))) , SI-(Article:A)
S2=,(Noun: PROGRAM) , S3=(Adjective:CONSCIOUS)

then, it is visible that RI does not apply, for SI is not a NP. So, the NP and PP have to be generated first. This will be done by means of:

R2: IF SI begins a NP THEN generate a new PP & insert it before SI & activate NP rules.
R3: IF S3* begins a PP THEN generate a new PP & insert it before S3 & activate PP rules.

R2 and R3 must be active at the time as RI. There is no need to order them since the conditions in RI will be satisfied only after R2 and R3 are applied. When they apply, they interrupt the development of the active structure by creating the NP and PP, and activating corresponding rules. Once the NP and PP are detached, the interpreter goes back to SI and applies RI

C. Quasi simultaneous execution

This is another original feature of our parser. As we have shown, the stack mecanism does not permit the correct recognition of the end of a constituent

* These rules have a variable ($) in this place: then it applies either if SI or S2 or S3 matches the condition part (cf). This seems an improvement on the "attention shifting rules" developped by Marcus.

** Inspired by SIMULA

"Quasi simultaneous" generation brings a new solution to this problem: a developping structure can be detached as soon as possible and elaborated latter if it is needed. This makes possible more intelligent choices in the situation of complex attachment mentioned above: when attaching a new structure, the parser may consider all the previously detached ones.

Two operations are involved: detach and resume. They are to be performed in following cases:
1) Creating a new structure; means detaching the current one before activating the new one.
2) End the current structure, which is then detached; the previous one being resumed.
3) Explicit detach instruction in a rule, implying that the last detached structure is resumed.
A) Similarly, explicit resume instruction in a rule.

Let us have some examples: "le cahier de l'etu diantc que Jean a examine." ("The note book of the girl that John examined-'it'.") (In French, the verb has a mark permitting to distinguish between "examined it", the note book and "examined her",the girl)

We see that: .a deterministic processing of the relative clause needs to wait for its attachment until one has treated the verb, which has a gender mark capable of indicating the correct attachment. ."To examine" has several meanings depending on what is examined. The choice of a certain meaning needs a recognition of the actual object of examine, that is equivalent, to the attachment of the relative clause. Here we have a couple of contradictory requirements.

Let. us detail how a quasi simultaneous treatment can get. us out of this predicament.
When the verb *of* the relative clause is seen,certain syntactic and semantic i'eatures of its object can be recognised; the parser then detaches the relative clause and resume the last seen NP, here, "the girl". There is no match with the 'it' mark, so the relative clause cannot, modify 'girl'. This NP is then terminated, and, the NP corresponding to 'note book' is resumed. This time, the relative clause is accepted as a modifier and can itself be resumed in order to solve the ambiguity in 'examine'.

We can also give examples of PP attachment:

"John puts the wine on the table."
"John drinks the wine on the table."

As soon as the noun 'the wine' has appeared, the NP can be detached, permitting the PP 'on the table' to be started. When considering possible attachments for this PP, the rules allow to take into account the features of the verb.

The differences between 'put' and 'drink' explains that it is possible to attach the PP to the verb in one case and to the NP in the second case.

CONCLUSION

After examining the main factors of non-determinism in parsers, we tried to solve the corresponding problems. Actual experimentation confirms the interest of such an approach. We made a SIMULA program of about 3000 instructions which parses sentences of about 10 words in 2 seconds on a PDP10 (using an approximate total of 100 K core).

A LISP version is being considered.

The present conceptual representation permits to consider synonymy, homonymy and selection restrictions. Moreover, in a perspective of integration of the domain knowledge to linguistic competence, this representation (or a more frame-like one) will make possible not only these tasks but also more sophisticated and useful inferences.

The program is already able to process the French forms of every example mentioned in the text, and we hope to make it usable to other research teams in the next future.

REFERENCES

(1) Ako, A. & Ullmann,"The theory of parsing, trans lation and compiling" Vol.] Prentice-hall 1972 .

(2) Fillmore, C.J. "The case for case" in Bach and Harms, Universals in linguistic theory. Holt, Rinehart & Winston, 1968.

(3) Halliday, M. "System and function in language" G.Kress, Oxford Un.press, 1976.

(4) Marcus, M.P. "A theory of syntactic recognition for natural language", MIT press, 1980.

(5) Riesbeck,C. & R. Schank "comprehension by computer: expectation-based analysis of sentences in context", Yale University, 1976.

(6) Rustin, R. "Natural languge processing" Algorithmic press, 1973.

(7) Sager, N. "Natural language information processing, a computer grammar of English and its application", Addison-Wesley, 1981.

(8) Winograd, T. "Understanding natural language" Academic press, New York, 1972.

(9) Winograd, T. "Language as a cognitive process" Vol.1, Addison-Wesley, 1982.

(10) Wilks, Y. "computational semantics: an introduction to artificial intelligence and NL comprehension", North Holland, 1976.

(11) Woods, W.A. "An experimental parsing system for TN grammars" in (6), 1973.

(12) Earley, J. "An efficient context-free parsing algorithm", CACM 6:8, 1970.

(13) Rady, M. "L' ambiguite" du LN est-elle la source du non determinisme des procedures de traitement" These d'etat, PARIS VI, 1983.