A Mechanism for ellipsis resolution in dialogued systems.

Díaz de Ilarraza Sanchez, A.*, Rodríguez Hontoria, H.+, Verdejo Maíllo, F.+

*Informatika Saila. Euskal Herriko Unibertsitatea
Informatika Fakultatea
Barrio de Ibaeta s/n. 20080 Donostia. Spain
Tno. (34) (43) 218000
E-mail: aradiaz@gorria.if.ehu.es

+ Department Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya.
C/ Pau Gargallo, 5 08028 Barcelona Spain
Tno. (34) (3) 4017000
E-mail: horacio@fib.upc.es

ABSTRACT: An ellipsis resolution mechanism is presented. The mechanism is a part of a Natural Language Understanding System developed in the last years in order to be included as a main component of several projects based on man/machine interactions. CAPRA, an intelligent system for teaching programming, and GUAI, a natural language interfaces generator, are two of such applications. In our approach, syntactic and knowledge-based techniques are combined in order to get a great coverage of elliptical cases.

1. Introduction.

Anaphoric reference always appear in any Natural Language application. Its occurrence is common in Dialogued Based systems. Present work describes an approach for the most emphasized form of anaphoric reference: the ellipsis. It has been implemented in a Natural Language Understanding System. This system is the core of some projects based on Human/Computer interaction. Tutor/student Interface of the CAPRA system [Garijo et al,87], and the interfaces generator GUAI [Rodríguez,89], are two of such applications.
Ellipsis resolution has to deal with two major subproblems:

1.- The analysis of the elliptical sentence.
2.- The reconstruction of the elided fragments.

Related to the first point, usually, flexible analysis techniques are applied [Hayes,Reddy,83]. For the second point, there is not a general solution. Several partial approaches have been made by means of syntactic [Weischedel,Sondheimer,82] or conceptual techniques based on focus exploration [Sidner,83].
Our approach uses both techniques in order to get a great coverage of resolved cases.

2. Dialogue management

NL-Dialogue systems are usually based on three data structures [Grosz,Sidner,86]: Dialogue Structure, to represent the organization of the interactions between the speakers, the Intentional Structure to organize the intentions of the speakers, core of the communicative process during the dialogue, and the Attentional Structure where topics of the Dialogue are represented.

The Dialogue Structure
The Dialogue Structure is a modelization of the communication process. This structure is dynamically built and is represented by a tree. **Conversation** is the root of the tree. A conversation takes place between several participants and it is composed by one or more **Dialogues.** Dialogues are units of communication characterized by a specific topic. A Dialogue is composed by one or more **Interchanges.** Diaalogues are tied to the Attentional Structure.
An Interchange has an **Objective** which must be filled. An Interchange has information about its goal, level of satisfaction and its evolution.
The Intervention is the elemental component of the Dialogue Structure. An Intervention is produced in a specific time, and implies that a message has been sent by a speaker to a hearer. The content of the message is represented in different linguistic levels.

The Attentional Structure
The Attentional Structure is used as a search-space in the resolution of some types of References. In our system, definite reference, direct and anaphoric, and, specially, pronominal and elliptical reference use the Attentional Structure. In our system, the Attentional Structure is mainly conceptual.

The organization of the analysis process.

In our approach the linguistic description follows a stratified model; however, parsing is a cooperative process between different **K**nowledge Sources. These KS generate partial interpretations, interrupt each other, ask for information to other components, etc .
In order to describe and implement different parsing strategies, our system is based on the use of a multilevel **Blackboard Architecture.**
In a descriptive level, at any moment, there will be a collection of concepts, already built in the parsing process, called **Realizations,** and another collection of virtual concepts, objects we wait for, called **Expectations.** The core of the process consists of obtaining some new realizations satisfying current expectations. Realizations and Expectations are created at different levels and implemented in a frame-like formalism.

3. Ellipsis.

Since the appearence of C. Sidner's works the relation between anaphora and focus is commonly accepted. The anaphoric treatment we propose is performed in a parallel way to the construction of the Attentional Structure. Relationships between anaphoric resolution and the Attentional Structure are as follows:
During the process of anaphora resolution, queries to the Attentional Structure are made in order to find the antecedent of the anaphoric expression, leading, sometimes to the modification of such structure.
We will examine here one of the more important and frecquent type of anaphora in systems based on dialogues: Ellipsis.
Two types of ellipsis can be considered: syntactical and conceptual. We say that a syntactical ellipsis happens when a syntactical component is missed. A conceptual ellipsis is detected when the value of a mandatory descriptor for that concept has not been given.

Conceptual ellipsis
When values of mandatory descriptors are not present, the system will generate expectations for instances that could fill the descriptors. The reference resolution process will try to solve these expectations, first, by means of the Attentional Structure and then by means of default values defined in the KB.

Figure 1 shows the objects generated during the interpretation process for the query " the current price of B.T. stocks".

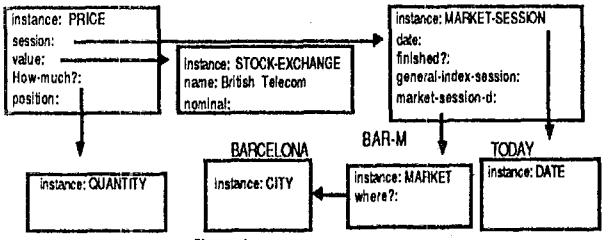

fig. 1.

Syntactical ellipsis

Syntactical ellipsis are usually short term anaphoric references. The resolution mechanism we have adopted considers that syntactical elements that are antecedent of the elided ones, must be found in the previous message of the user himself/herself. The method emphasizes the parallelism between whole and elided expressions.

The syntactic formalism we used is based on a R.P.S.G. ([Sager,81], [Hirschman,Puder,86]).

The syntactic structures managed by the system are parse trees and basically the process consists of an unification of two parse trees. Formally, they are n-ary labeled trees. The label keeps only the syntactical category and, eventually a list of associated syntactic features.

The classical formulation of unification for two trees is based on a preorder traversal of both trees, in parallel, trying to unify the different nodes. A strict application of the unification algorithm will not be very useful. In order to get a great coverage it is necessary to make more flexible the unification conditions. This will be done in three aspects:

Taking into account the type of components to be unified. In the kind of syntactical formalism we use, rules, and so the structure of the parse tree, are derived by (1) **Substitution** of a syntactic category by another, or (2) **Adjunction** of modifiers chains to a central one. Grammatical categories are so subcategorized into central and adjunct.

In our unification algorithm, the adjunct components of the first tree (ea: ellipsis antecedent) dont't have to participate in the unification process. The adjunct components of the second tree (ec: elliptical component) are considered in the unification process but they can be unified with empty trees.

On the other hand, the order of the adjunct components in any of the trees doesn't matter to the unification process (even in the case that this order would had been important during the syntactic process)

Another point to be considered is the relaxation of the unification conditions between components. Usually, two nodes are unifiable if they have the same syntactical category and their syntactic features are compatible. We establish a flexible criterium introducing the concept of compatibility between categories. We will not require that two categories must be identical but compatible. The idea of compatibility between categories is based on the equality of distributional features, though it has been weakened (so, for example, <*N > and <*PRON> (Noun and Pronoun) are compatible).

3.1 Unification Algorithm

The algorithm has two phases: During the first one *ea* (ellipsis antecedent) and *ec* (elliptical component) trees are unified. If the unification has been successful, the result of this phase is a target tree in which some adjunct components can be incorrectly placed. The second phase transforms the target tree in order to get a right placement of all the constituents.

Unification Phase

Components of *ea* categorized as central are incorporated to the target tree in the same structural position they had in *ea* . If the algorithm finds theirs equivalents in *ec* then the label of constituents in the target tree will hold the information contained in *ec*.

The adjunct components of *ec* are incorporated, together with their labels, to the target tree having or not been unified with a corresponding component in *ea* .

The adjunct components of *ea* whitout equivalent in *ec* are not incorporated to the target tree. They are saved in a list of pending components.

If there is a central component in *ec* whitout equivalent in *ea* , the unification process fails.

The implementation of the algorithm is based on two main mutually recursive functions: **dd-tree-unification** and **dd-forest-unification.**

The function dd-tree-unification takes as arguments the trees to be unified. It examines the compatibility of the roots by means of a call to the **dd-compatible** function. If the roots are not compatible, the function returns an empty list.

If the roots are compatible, the function makes a call to dd-forest-unification whose arguments are forests composed by the children of both trees. The function returns a tuple composed by the target tree, the two pending forests and the list of pending adjunct elements we talked above.

The function dd-forest-unification takes as arguments two forests and tries to unify their initial fragments. Both forest are traversed in parallel trying matching their corresponding trees until one of the forests becomes empty or the unification fails. Each matching considers the following cases:

1.- Both trees are unifiable (a call to dd-tree-unification has been successfully made). In this case the unification goes on and, eventually, the size list of pendings is increased, the unified trees are eliminated from the respective forests and the pending forests that the function dd-tree-unification returns are incorporated to the new ones for their treatment.

2.- The tree of the first forest is neither unifiable nor adjunct. The component is incorporated to the target tree and its children to the correspondent forest.

3.- The tree of the second forest is not unifiable but adjunct. The component is incorporated to the target tree and its children to the correspondent forest.

4.- None of the previous cases happen. In this case, the unification process is stopped.

In any of the previous cases, the function returns a similar tuple to the dd-tree-unification. dd-forest-unification never fails. If the algorithm fails in this first phase, then the unification is not possible.

Transformation Phase

Inputs for the second phase of the algorithm are the target tree and the list of pending adjunct components. The adjunct components of the target tree not unified in the former phase are examined and their corresponding elements are searched in the list of pending elements. If an element is found, it is deleted from the list and the position of the component of the target tree is modified, indicating that its correspondent element has been found (unified).

When the process is finished, the pending adjunct components are not considered anymore and constituents of the target tree are confirmed in their positions.

Example

Let us follow the application of the algorithm with an example. Figure 2 shows the syntactic structure of the first intervention "Cuál es el índice de hoy?". The second intervention "Y el de ayer?" is shown in figure 3. Figure 4 shows the target tree "Cuál es el índice de ayer?", result of the unification process.

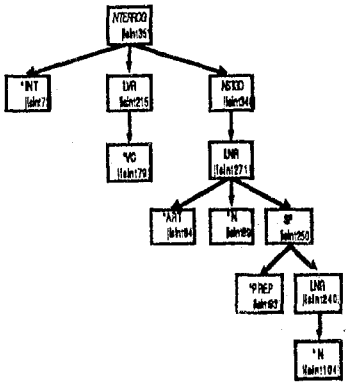

fig. 2.

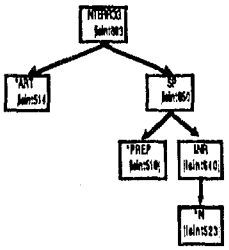

fig. 3.

The algorithm starts by trying to unify trees with roots lisint351l and lisint683l, there are not pending adjuncts. Syntactic categories of these components are identical, INTERROG, and there is not syntactical features, so the unification is possible. Now, the problem is reduced and we must unify two forests, the first one composed by the trees with roots lisint73l, lisint215l and lisint346l and the second one by lisint514l and lisint650l.

lisint73l and lisint215l, are central and without any possible unificator in the second forest, then they are incorporated to the target tree. The first forest now is composed by the trees lisint79l and lisint346l, and there are no changes in the second one.

The same considerations will lead to incorporate components lisint79l and lisint346l. The first forest is now composed by the element whith the lisint271l root. Second one remains without changes.

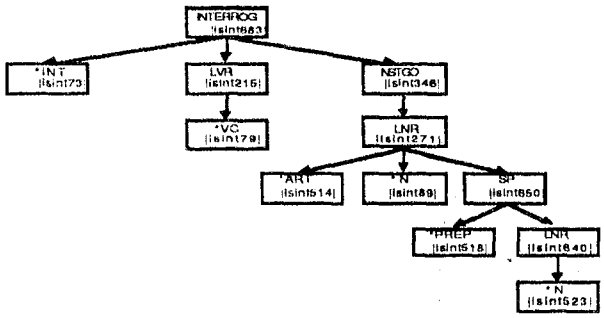

fig. 4.

The incorporation of lisint271l to the target tree produces the unification of the forest composed by lisint84l , lisint89l and lisint250l, and the second one, still remains without changes.

Nodes lisint84l and lisint514l, both with *ART category, "masculine" and "singular", are unifiable without problems. lisint89l being a central node is also incorporated. Now the problem is reduced to the unification of the prepositional phrases lisint250l and lisint650l which it is done easily and the resulting target tree is shown in figure 4. Further cases can be found in [Rodriguez, 89] and [Diaz de Ilarraza, 90]

4. Conclusions

We have presented here a mechanism for ellipsis resolution in dialogued systems. Our proposal combines syntactic and contextual information. First one is used to unify the parse tree of the elliptical sentence with trees of previous sentences. The unification algorithm is more flexible than those presented in the literature, and in consequence, solves more cases. Its flexibility is based on the differentiated treatment for central and adjunct components of the parsing trees as well as a greater freedom in the syntactic realization of the constituents. Syntactic resolution is complemented with resolution rules for conceptual ellipsis working on the dialogue structures. As a result a broad range of cases are covered by the system.

5. References.

[Diaz de Ilarraza, 90] Diaz de Ilarraza, A.: "Gestión de diálogos en lenguaje natural para un sistema de enseñanza inteligente". Tesis Doctoral. Universidad del País Vasco,1990

[Garijo et al,87] Garijo, F., Verdejo, F., Díaz de Ilarraza, A., Fernández, I. ,Sarasola, K.: "CAPRA: An intelligent system to teach novice programmers". International Conference on Artificial Intelligence, San Sebastian, 1987.

[Grosz,Sidner,86] Grosz, B., Sidner, C.: "Attention, Intentions and the Structure of Discourse". Computational Linguistics Vol. 12, Num. 3, 1986.

[Hayes,Reddy,83] Hayes, Reddy: "Steps toward a graceful interaction in spoken and written man/machine interaction". I.J.M.M.S. Vol. 19, 1983.

[Hirschman,Puder,86] Hirschman, L., Puder, P.: "Restriction Grammar: A Prolog implementation". En "Logic Programming and its applications". Ablex Series in Artificial Intelligence, 1986.

[Hirst,79] Hirst, G.: "Anaphora in Natural Language Understanding: A survey". Tech. Report 79.2, D.C.S.U. British Columbia, Vancouver, Canadá, 1979.

[Rodríguez,89] Rodríguez, H.: "GUAI: Un generador automático de interfaces en lengua natural". Tesis Doctoral. Facultat d' Informàtica U.P.C. Barcelona, 1989.

[Sager,81] Sager, N.: "Natural Language Information Processing". Addison-Wesley, 1981.

[Sidner,83] Sidner, C.: "Focusing in the comprehension of definite anaphora". En "Computational Models of discourse".Brody, M. y Berwick, R. eds. MIT Press, Cambridge, MA, 1983.

[Weischedel,Sondheimer,82] Weischedel, Sondheimer: "An improved heuristic for ellipsis processing". 20th. A.M. of A.C.L., 1982.