Pronominal and Anaphor Resolution

Nikitas N. Karanikolas
Department of Informatics, Athens University of Economics and Business, Athens, GREECE.

Chomsky's Binding Conditions imply that each position of an antecedent of an anaphor cannot be a position for the antecedent of a pronominal. However, there have been examples in the literature where the above implication does not hold. A brief review of the linguistic background and the various definitions, relevant to this problem, given by other researchers is presented. Careful examination of these definitions leads to the conclusion that the problem can be resolved by considering two different definitions of the governing category, one for the governing category of a pronominal and one for the governing category of an anaphor. The definitions and rules selected are used to design computer algorithms. Using the Binding Conditions it is possible to find only the impossible antecedents in case of pronominals. Other algorithms which combine the information available from these conditions, in order to find the possible antecedents of any pronominal are also suggested.

Keywords: natural language semantic interpretation, computational linguistics, anaphora resolution

1. Introduction

Both theoretical linguists and researchers in the field of Natural Language Understanding (NLU) have examined the problem of finding the possible antecedents of pronominals (pronouns) and anaphors (reflexive or reciprocal pronouns) from a different point of view. Recently, the NLU researchers, mainly, have tried to combine both approaches. They have tried to take advantage of linguistic theory approaches in their efforts to implement systems that are able to solve cases of anaphora. This paper falls in this category, i.e. it presents an approach that tries to integrate the linguistic theory and technological approaches in the field of NLU. It is, however, based on complete linguistic theories, namely those of Chomsky's (1981). Linguists such as Huang (1983) have suggested some modifications, in order to solve the problems that these theories exhibit. Although some problems have been solved, the modified theories are still more difficult to implement. This paper tries to combine two of Chomsky's theories, in order to acieve the results of Huang's modifications and at the same time it has no difficulties for implementation.

Linguistic theory is the subject of the following section. The subsequent section presents Chomsky's definition of governing category and the modifications suggested for the same concept by Huang. The following section examines the implementation issues of the suggested model. Because the government binding theory finds only the impossible antecedents of a pronominal, we shall present necessary extensions to cover the discovery of possible ones. The last two sections present examples of the system's operation and comparisons with other systems existing in the literature.

2. Linguistic background

Chomsky (1981, p. 188) proposed that binding theory has three conditions, namely:

An anaphor is bound in its governing category, (1a)

A pronominal is free in its governing category and (1b)

An R-expression is free inside the sentence it belongs to. (1c)

The anaphora conditions proposed by Reinhart (1983, p. 136) are the following:

A non-pronominal NP must be interpreted as non-coreferential with any NP that c-commands it, (2a)

A reflexive or reciprocal pronoun (an R-pronoun) must be interpreted as coreferential with (and only with) a c-commanding NP within a specified syntactic domain (e.g. its minimal governing category) and (2b)

A non-R-pronoun must be interpreted as non-coreferential with any c-commanding NP in the syntactic domain which is specified for (2b). (2c)

As Reinhart (1983 p. 139) points out, the conditions (2b), (2c) and (2a) are similar to the conditions (1a), (1b) and (1c) respectively, if we assume that:

R-pronouns are reflexive (e.g. *himself*) or reciprocal (e.g. *each other*) pronouns, (3)

Pronominals are non-R-pronouns and elliptical subjects of infinitive subsentences (noted with the symbol PRO in the literature), (4)

Anaphors are R-pronouns and traces of extraposed noun phrases (NP-traces), (5)

R-expressions are lexicaly realized noun phrases (e.g. *"the blue river"*), (6)

Lexical NPs are case marked, (7)

A NP is bound if it is coindexed with a c-commanding NP and (8)

A NP is free if it is not bound. (9)

3. C-Command

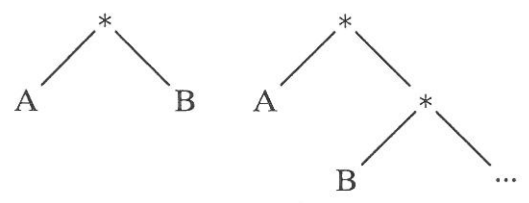
Reinhart (1983) presented two definitions of c-command. The first of these definitions (p. 18) is repeated here as definition (10), while the second (p. 23) is repeated as definition (11):

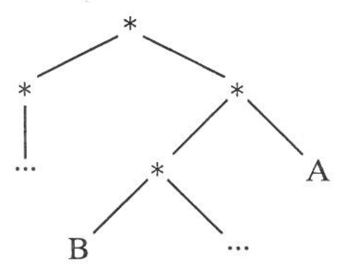
Node A c-commands node B, if and only if the branching node most immediately dominating A also dominates B. (10)

Node A c-commands node B, if and only if the branching node $\alpha 1$ most immediately dominating A either dominates B or is immediately dominated by a node $\alpha 2$ which dominates B and $\alpha 2$ is of the same category type as $\alpha 1$. (11)

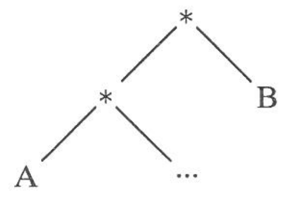
As Reinhart (1983, p. 24) points out, Chomsky uses essentially the simplified definition (10), requiring further that neither A contains B, nor B contains A. This modified definition is presented in Radford (1981, p. 314), and is adopted in our pronominal and anaphor resolution system.

According to definition (10), in each of the following syntactic structures, node A c-commands node B. The symbol "$*$" is used wherever the syntactic category of a node is of no interest.

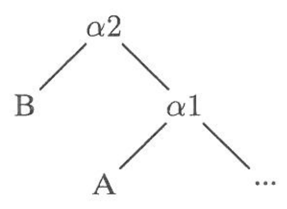




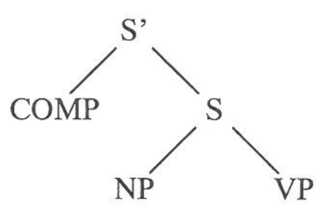
In the following syntactic tree node A does not c-commands node B.



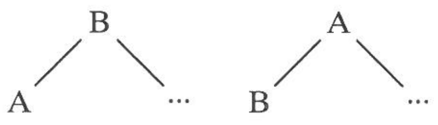
Definition (11) permis a node A to c-command a node B not only in every case where definition (10) permits it, but also in the following syntactic structure, provided the nodes $\alpha 1$ and $\alpha 2$ have the same syntactic category.



For example, in the following syntactic structure, the node NP c-commands the node COMP, because the nodes S and S' are of the same syntactic category:



Chomsky's demand, that neither A contains B, nor B contains A, prohibit A to c-command B in the following syntactic structures:



4. An approach that uses Reinhart's anaphora conditions

Conditions (1) and (2) use the terms *governing category* and *minimal governing category* for the same concept. Ingria and Stallard (1989) adopted conditions (2), but replaced the term *minimal governing category* with the term *minimal syntactic domain*. The definition of this term and another complementary definition of the same term presented in their paper (p. 263) and (p. 264) are given in (12) and (13) respectively:

> The immediately dominating finite clause (S) node always constitutes a minimal syntactic domain. NP nodes normally do not constitute a minimal syntactic domain, unless they contain a possessive and (12)

> The S node or NP node (when minimality has been induced by the presence of a possessive) that most immediately dominates the node being processed constitutes a minimal syntactic domain. (13)

The above definition of minimal syntactic domain (governing category) handles correctly the pronominal and anaphor reference problem for a large number of sentences (examples in Ingria and Stallard, 1989). There are, however, other cases of sentences, for example sentences (14) and (15), where the above definition of minimal syntactic domain is not the correct one.

> The men read each other's books. (14)

> The men read their books. (15)

For sentence (14) the minimal syntactic domain of the reciprocal pronoun *each other*, according to definitions (12) and (13), is the NP2 *each other's books*. The conditions (1a) and (2b) demand that the reciprocal pronoun must be coindexed with another NP inside NP2. Because there is no other NP inside NP2 other than the reciprocal, sentence (14) is characterized as ill formed.

For the pronominal *their* in sentence (15) the minimal syntactic domain is the NP4 *their books*. According to conditions (1b) and (2c) the pronominal cannot be coindexed with any other c-commanding NP inside NP4. The non existence of any other NP inside NP4, other than the pronominal *their*, satisfies (1b) and (2c). However, the NP3 *The men* is outside the minimal governing category of the pronominal *their* and can be coindexed with it.

We have shown that the definitions of the minimal syntactic domain (governing category) (12) and (13) give the correct result and as a consequence the resolution of the pronominal reference in (15) is correct. However, applying the same definitions to sentence (14) results to considering it as ill formed, instead of deducing that the whole sentence (14) is the governing category of the reciprocal pronoun *each other* and accepting the NP1 *The men* as coindexed with this reciprocal pronoun. In order to solve this problem, Ingria and Stallard (1989, p. 264) suggested modification (16).

> The NP that contains a possessive counts as a minimal syntactic domain, for all the nodes that it dominates, except the possessive itself (16)

Given the modified definition above the reciprocal *each other* of sentence (14) does not have as minimal syntactic domain the NP2 but the whole sentence. In this case, the conditions (1a) and (2b) suggest that the NP1 is a coindexed node with the reciprocal pronoun. Modification (16) introduces, however, another problem: the pronominal *their* of sentence (15) has now minimal syntactic domain the whole sentence and according to (1b) and (2c) it cannot be coindexed with NP3. Thus, modification (16) solves one problem but introduces another (maybe less severe) one.

5. Governing category

One definition of the term governing category, given in Chomsky (1981, p. 188), is repeated here (17):

> α is the governing category of β, if and only if α is the minimal syntactic category containing β and a governor of β, where α=NP or S. (17)

Another definition that Chomsky (1981, p. 211) gives is (18):

> α is a governing category of β if and only if α is the minimal category containing β, a governor of β and the SUBJECT accessible to β. (18)

Huang (1983, p. 554) gives (among others) the following examples (19) and (20), and points out that under definition (17) the bracketed part of them is the governing category of the reciprocals in (19) and the pronominals (in symmetric positions to the reciprocals in (19)) in (20). Those are implied by the fact that the governor (see definition (29)) in (19a) and (20a) is the noun *pictures*, the governor in (19b) and (20b) is the preposition of and the governor in (19c) and (20c) is the prepositional complementizer *for*.

> *They* saw each other's pictures. (19a)
>
> *They* expected that {pictures of *each other*} would be on sale. (19b)
>
> *They* expected that {for *each other* to come would be possible}. (19c)
>
> *They* saw {*their* pictures}. (20a)
>
> *They* expected that {pictures of *them*} would be on sale. (20b)
>
> *They* expected that {for *them* to come would be possible}. (20c)

Subsequently, condition (1b) correctly allows pronominal coreference in each of sentences (20), but condition (1a) wrongly excludes the sentences (19) as ill formed. Therefore definition (17) presents the same problems as definitions (12) and (13) before introducing definition (16).

Huang (1983, p. 555) also points out that under definition (18) the whole sentence (the outermost S not S') becomes the governing category for the reciprocals in sentence (19) and the pronominals (in symmetric possitions to the reciprocals in (19)) in sentence (20). Subsequently, the reciprocals in sentence (19) can be coindexed with the subject (of the outermost S) *They*, according to the condition (1a). But, according to condition (1b), the pronominals (in symmetric positions to the reciprocals in (19)) of sentence (20) must be disjoint in reference to the subject (of the outermost S) *They*. Huang (1983, p. 557) has proposed a minimal modification of the definition of a governing category, (21), that solves these problems:

> α is the governing category of β, if and only if α is the minimal category containing β, the governor of β and the SUBJECT accessible to β if β is an anaphor. (21)

In order for definition (21) to have the effect of (17) for pronominals and the effect of (18) for anaphors, Huang modified Chomsky's definition of the SUBJECT (22) to the following definition (23):

> The SUBJECT of a clause S is [AGRi, S], if there is one, otherwise it is [NPi, S]. The SUBJECT of a noun phrase NP is [NPi, NP]. The notation [X, Y] means that X is immediately dominated by Y, except [AGRi, S] where AGR can be dominated by a node Z (for example INFL) and Z is dominated by S. (22)
>
> The SUBJECT of a maximal phrase A is the subject of A or the nominal head of A. (23)

The definition (22) of SUBJECT can be decomposed to the following three cases:

i) The SUBJECT of a tensed sentence is its AGR node.

ii) The SUBJECT of a non-tensed sentence is its subject NP.

iii) The SUBJECT of an NP is its possessional NP.

Furthermore, we analyze sentences (20b), (20c), (19b) and (19c), according to definitions (21) and (23). The analysis of sentences (20a) and (19a) is similar to that of sentences (20b) and (19b) and is therefore omitted.

According to definition (23) the noun *pictures* in (20b), that is the nominal head of the NP *pictures of them*, constitutes its SUBJECT. Thus, the NP *pictures of them* is the governing category of the pronominal *them*, because, according to definition (21), it includes the pronominal, its governor of and the SUBJECT *pictures*.

In sentence (20c) the clause (S') *for them to come* is the SUBJECT of the clause (S) indicated by the curly brackets. Clause S also includes the pronominal *them* and its governor, namely, the prepositional complementizer *for*. Therefore, clause S, indicated by the curly brackets, is the governing category of the pronominal *them* in sentence (20c).

Before examining (19b) and (19c) according to the definitions (23) and (21) we present the definition of accessibility (24) and the coherent notion of the i-within-i condition (25):

> α is accessible to β, if and only if α c-commands β and the assignment of the index of α to β does not lead to a violation of the i-within-i condition (25). (24)

> Two nodes can not be coindexed (share a common index) when the one dominates the other. (25)

In the case of an anaphor, definition (21) demands that the SUBJECT must be accessible to the anaphor. In sentence (19b) the SUBJECT *pictures* c-commands the reciprocal *each other* and the assignment of the index (i) of the SUBJECT *pictures* to the reciprocal *each other* does not violate the i-within-i condition. Thus, the SUBJECT *pictures* is accessible to the reciprocal *each other*. Consequently, the rule (21) does not work in the same way as the rule (18) in the case of an anaphor, because it finds the NP *pictures of each other* as the governing category of the reciprocal *each other*. Huang's (1983) solution to this problem given in note 4 of his paper is repeated here as definition (26):

> We will assume that the referential index of a head N comes from the maximal NP node. Thus, the head N *pictures* as a SUBJECT in *pictures of* α is not accessible to α, since coindexing the head and α necessarily violates the i-within-i condition. (26)

Under the assumption (26), the NP *pictures of each other* does not have an accessible SUBJECT to the reciprocal *each other*. In this case the next structure that is in accordance with (21) is the S node of the whole sentence that includes the accessible SUBJECT *AGR* (not presented in the surface sentence (19b)).

The clause indicated with the curly brackets in sentence (19c) includes the reciprocal pronoun *each other*, its governor and the SUBJECT (S') *for each other to come*. In this case the SUBJECT is not accesible from the reciprocal pronoun, because the SUBJECT dominates the reciprocal pronoun and consequently the SUBJECT cannot c-command the reciprocal pronoun. Thus, the next structure that is in accordance with (21) is the whole sentence. This implies the possibility of coreference between the pronoun *They* and the reciprocal pronoun *each other*.

Using only one definition of the governing category, namely definition (21), Huang (1983) solved the problem of the sentences (19) and (20) in which pronominals and anaphors are not mutually exclusive. The cost of his solution is the modified definition of SUBJECT (23) and the assumption (26) which complicates the theory of pronominal and anaphor resolution. In this paper we propose that a better solution is to use two different definitions for the governing category — namely the definition (17) for the governing category of a pronominal and the definition (18) for the governing category of an anaphor — and avoid making any other modifications to the existing theory. This solution is adopted and is presented in the following section, in the context of transforming the presented rules and definitions to algorithms.

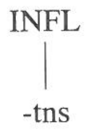
6. Implementing the three binding conditions

In this section, the algorithms that implement the three binding conditions (1) or (2) — that constitute the theory of pronominal and anaphor resolution — are presented. In the next section we will present an algorithm that uses the information produced by the algorithms presented here in order to find the possible antecedents of any pronominal in a sentence. We will present first the algorithms that implement condition (1b) and find the nodes by to which the processed pronominal must be disjoint. Furthermore, the algorithms that implement condition (1a) and produce the list of nodes which the processed anaphor must be bound will be presented. Finally, we will present the algorithms that implement (1c) or the more appropriate for implementation but similar condition (2a). Before giving the details of these algorithms we must present the following four conventions that we have adopted:

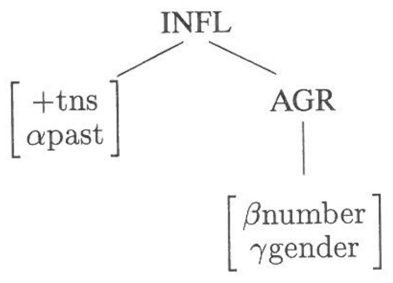
i) The term lexical noun phrase (or lexical NP) is used whenever we want to emphasize that the noun phrase is neither pronominal, nor R-pronoun.

ii) The COMP node in a sentence can be empty, filled by the complementizer *that* or filled by the prepositional like complementizer *for*. The complete structures of the COMP node in each case are the following: comp, comp(that), comp(p(for)).

iii) The INFL (inflection) node (van Riemsdijk, 1986, p. 274) of a sentence (S) is (27a) in the case of a non-tensed sentence and (27b) in the case of a tensed sentence.



(27a)



(27b)

We can assume that there is no INFL node at all for a non-tensed clause, because the information that this carries does not make any contribution to the pronominal and anaphor resolution. For a tensed clause we assume that the INFL node is reduced to its *AGR* (agreement) constituent. The latter assumption permits us to use the definition (22) of SUBJECT without the exception of non-immediate domination of AGR by S.

iv) The last assumption is that the syntactic trees are labeled. We have implemented an algorithm that takes unlabeled syntactic trees as input and produces the corresponding labeled ones. For example, if the syntactic tree (28a) of sentence (19a) is given to the algorithm, the latter produces the tree (28b).

```
s_( comp,
    s( np(pron(they)),
       agr,
       vp( v(see),
           np(
               np(recipr(each_other)),
               n(pictures)))))
```

(28a)

```
s_(0, comp(1),
      s(2, np(3,pron(4,they(5))),
           agr(6),
           vp(7, v(8,see(9)),
                 np(10,
                        np(11,recipr(12,each_other(13))),
                        n(14,pictures(15))))))
```

(28b)

In the above structure the label of each node is its first argument. Each underscore character following a node name indicates another bar of the node. For example, s_, s__ and n_ indicate S', S" and N' correspondingly.

7. Pronominal Algorithm

The first algorithm, called *pronominal*, takes as its input the labeled syntactic tree of a sentence and the label of a non-R-pronoun (P) and returns a list of free nodes according to definition (1b). Before the introduction of the suggested algorithm we will present the definition of the government (Chomsky, 1981, p. 164):

α governs γ when (29b), (29c) and (29d) hold. (29a)

None of α and γ dominates the other. (29b)

α is a node of type a (adjective), p (preposition), n (noun), v (verb), agr (agreement) or the prepositional complementizer *for* of a sentence. (29c)

Each maximal projection that contains one of α or γ must also contain the other. (29d)

Maximal projection can be every node of type s', np, ap, pp, vp. (29e)

The suggested algorithm is:

(1) Find the innermost maximal projection (IMP) that includes the non-R-pronoun (P).

(2) Find the governor (G) of P inside IMP.

(3) Find the innermost node of type S or NP that includes P and G. This node, according to the definition (17), is the governing category (GC) of P. (4) Find and return a list of (the labels of) the NPs internal to the GC.

(5) The NPs of the above list that c-command P are the nodes that cannot be coindexed with P and constitute the list of free nodes.

Step (1), finds the innermost maximal projection that includes γ (the governed node P). Step (2), checks if any node (α) of the nodes internal to the innermost maximal projection of γ satisfies each of the following tests:

i) it is one of the syntactic categories of (29c),

ii) its innermost maximal projection is the same with the innermost maximal projection of the governed node γ and

iii) γ does not dominate α.

Because α is one of the types in (29c), it dominates a leaf (actual word) and it is impossible for it to dominate γ. The node (α) that satisfies these tests is the governor (G) of P. Steps (3), (4) and (5) are straightforward and do not need any explanation.

8. Anaphor Algorithm

The second of the algorithms, called *anaphor*, takes as input the labeled syntactic tree of the sentence and the label of an R-pronoun (R) and returns a list of bounding nodes according to definition (1a). The suggested algorithm is:

(1) Find the innermost maximal projection (IMP) that includes the R-pronoun (R).

(2) Find the governor (G) of R inside IMP.

(3) Find the SUBJECT (ASUBJ) which is accessible to R.

(4) Find the governing category (GC) of R that, according to definition (18), includes R, G and ASUBJ.

(5) Find and return a list of (the labels of) the NPs internal to the GC.

(6) The NPs of the above list that c-command R are the nodes that must be coindexed with it and constitute the list of bounding nodes.

The above algorithm is very similar to the pronominal algorithm discussed previously. The main difference is the insertion of an additional step (3). This step is justified in the following paragraph.

For a SUBJECT to be accessible to another node (B), it must c-command node B according to (24). But, in each of the three cases above, the SUBJECT c-commands any node (except the nodes that it dominates) inside the S or the NP node of which it is the SUBJECT. Consequently, searching for an accessible SUBJECT for a given node (R) reduces the search space to the S or the NP nodes that include R. Thus, starting from the innermost S or NP nodes that include R, one can check if the SUBJECT of the processed S or NP node c-commands R and if the assignment of the index of the SUBJECT to R does not violate the i-within-i condition. If these restrictions are satisfied, then the SUBJECT of the processed S or NP node is the SUBJECT accessible to R, otherwise the next innermost S or NP node that includes R is examined.

There are four cases for which the SUBJECT of the processed S or NP node c- commands the R node and the i-within-i condition is not violated. First, when the processed node is an NP node, the leftmost node that it immediately dominates is a (possesional) NP node, which is the SUBJECT, and the R node isn't dominated by the SUBJECT NP:

$$[_{np}[_{np} \ldots] \ldots \mathrm{R} \ldots]$$

Second, when the processed node is a non-tensed S node, the leftmost node that it immediately dominates is an NP node, which is the SUBJECT, and the R node isn't dominated by the SUBJECT NP:

$$[_{s}\, [_{np} \ldots] \ldots \mathrm{R} \ldots]$$

In both these cases the SUBJECT NP c-commands R and is accessible to R. However, both cases demand that the R node is not dominated by the SUBJECT NP, since such domination will imply violation of the i-within-i condition:

$$[_{x}\, [_{np} \ldots \mathrm{R}_i \ldots]_i \ldots]$$

where x is s or np.

Third, when the processed node is a tensed S node, the leftmost node that it immediately dominates is an NP node and the NP node does not dominate the R node:

$$[_s\, [_{np} \ldots]\ \mathrm{AGR} \ldots \mathrm{R} \ldots]$$

In this case, the AGR node c-commands any node inside the tensed S node and it is the SUBJECT accessible to R. However, we need to test whether the R node is dominated by the NP node or not, because such domination implies the existence of a structure of the form:

$$[_s\, [_{np} \ldots \mathrm{R}_i \ldots]_i\ \mathrm{AGR}_i \ldots]$$

and, consequently, it implies violation of the i-within-i condition. In the above structure, the coindexing of the subject NP and the AGR results from definition (30), presented in Chomsky (1981, p. 211):

AGR is coindexed with the NP it governs. (30)

Fourth, the processed node is a tensed S node and the leftmost node that it immediately dominates is not an NP node. In this case, AGR is the accessible SUBJECT to R and R can lie anywhere inside the S node, without violating the i-within-i condition:

$$[_s\, [_x \ldots]\ \mathrm{AGR}_i \ldots \mathrm{R}_i \ldots]$$
$$[_s\, [_x \ldots \mathrm{R}_i \ldots]\ \mathrm{AGR}_i \ldots]$$

where x is not a NP node

9. FullNP Algorithm

Full NP is a NP that is neither a pronominal nor an anaphor. In other words, a full NP is a lexical NP. The third algorithm that we call fullNP takes as input the labeled syntactic tree of a sentence and the label of a full NP and returns a list of free nodes according to (2a). The suggested algorithm is:

(1) Find and return a list of (the labels of) all NPs of the sentence.

(2) The NPs of the above list that c-command the processed full NP (P) are the nodes that cannot be coindexed with P and constitute the list of free nodes.

10. Beyond the three binding conditions

The above three algorithms are invoked by another algorithm. The later traverses the labeled syntactic tree and wherever it finds a non-R-pronoun it invokes the pronominal algorithm, wherever it finds an R-pronoun invokes the anaphor algorithm and wherever it finds a full NP invokes the fullNP algorithm. The application of this algorithm gives three lists of compound (Prolog) objects. Each compound object of the first list includes information on an R-pronoun of the sentence, each compound object of the second list includes information on a non-R-pronoun of the sentence and each compound object of the third list includes information on a full NP. The information given on an R-pronoun includes its type (reflexive or reciprocal), its NP label and the bound list that is returned by the anaphor algorithm on the given R-pronoun. The information given for a non-R-pronoun of the second list includes its NP label, the free list returned by the pronominal algorithm and two other uninstantiated arguments. The information given for a full NP of the third list includes its label and the free list returned by the fullNP algorithm.

One of the goals pursued is to find the list of all its possible antecedents for each pronominal. Unfortunately, the information available for each pronominal and full NP is the list of impossible antecedents. This problem is resolved in two steps. The first step cross examines each pronominal of the second list with any full NP of the third list. If neither the NP label of the pronominal is in the free list of the full NP, nor the label of the full NP is in the free list of the pronominal, then the label of the full NP is appended to the possible antecedent NPs of the pronominal has been examined. This step results in the instantiation of the third argument of each compound object of the second list to a list of the labels of its possible antecedent full NPs. For example, if one of the Prolog objects (of the second list) that represents a pronominal (labeled x) with two contraindexed (free) NPs (labeled y and z), is:

pron(x, [y, z], ?, ?)

then, if there are three lexical NPs (in the third list), with labels k, l and m, that do not contain x in their own list of free NPs, this object will be transformed to:

pron(x, [y, z], [k, l, m], ?)

The second step cross examines all pairs of pronominals of the second list. If the first pronominal does not contain the NP label of the second in its free list and if the second pronominal does not contain the NP label of the first in its free list, then the NP label of the second pronominal is appended to the possible antecedents of the first pronominal. This step results in the instantiation of the fourth argument of each compound object of the second list to a list of the NP labels of its possible antecedent pronominals.

11. Examples of the system's operation

The system's operation is presented by analysing the following two sentences. The first presents a case of anaphor resolution, while the second sentence presents a case of pronominal resolution.

They trust each other. (31)

His mother loves John. (32)

The (Prolog) parse tree for sentence (31) is he following:

```
s_( comp,
    s( np(pron(they)),
       agr,
       vp(v(trust),np(recipr(each_other)))))
```
(33)

This parse tree constitutes the input to our pronominal and anaphor resolution system. As a first step, the algorithm inserts labels to each node of the tree and produces the labeled syntactic tree (34):

```
s_(0, comp(1),
      s(2, np(3,pron(4,they(5))),
           agr(6),
           vp(7, v(8,trust(9)),
                 np(10,recipr(11,each_other(12))))))
```
(34)

The execution of the proposed algorithm results in three lists for each sentence being examined. The format of each element of the first list is recipr(A,B) or reflex(A,B). A is the noun phrase (NP) label of the reflexive or reciprocal pronoun. B is the list of NP labels contraindexed with (impossible antecedents of) A. The format of each element of the second list is pron(C,D,E,F). C is the NP label of the pronominal. D is the list of NP labels contraindexed with C. E is the list of lexical NP labels which are possible antecedents of C. F is the list of pronominal NP labels which are possible antecedents of C. The format of each element of the third list is np(G,H), where G is the label of the lexical NP and H is the list of NP labels contraindexed with G. In the case of sentence (31) the execution of the algorithm produces the following three lists (35):

```
[recipr(10,[3])]
[pron(3,[ ],[ ],[ ])]
[ ]
```
(35)

The first of these lists contains one compound object, namely, recipr(10,[3]). This compound object corresponds to the NP *each other* which is a reciprocal pronoun, its NP label is 10 and its only possible antecedent is the pronoun *they* with NP label 3. The second of the above lists contains only one compound object, namely, pron(3,[ ],[ ],[ ]). This means that the only pronominal *they* of (31) with NP label 3 does not have any contraindexed NP in the sentence, therefore the first list of the compound object is empty. Secondly, there is no lexical (full) NP that can be an antecedent of the pronominal, therefore the second list of the compound object is empty. Finaly, there is no other pronominal that can be coreferential with this pronominal, therefore the third list of the compound object is empty. The third of the above lists is empty, because there is no lexical NP in (31).

The (Prolog) parse tree of sentence (32) now follows:

```
s_( comp,
    s( np(np(pron(his)),n(mother)),
       agr,
       vp(v(love),np(john))))
```
(36)

This parse tree feeds our pronominal and anaphor resolution system. The labeled syntactic tree

(37) and the lists resulting (38) by the execution of our algorithm are:

```
s_(0, comp(1),
   s(2, np(3,np(4,pron(5,his(6))),n(7,mother(8))),
       agr(9),
       vp(10,
            v(11,love(12)),
            np(13,john(14)))))
```

(37)

```
[ ]
[pron(4,[ ],[13],[ ])]
[np(3,[ ]),np(13,[3])]
```

(38)

The first list of (38) is empty, because there is no anaphor (reflexive or reciprocal pronoun) in sentence (32).

The second list of (38) contains one compound object, namely pron(4,[ ],[13],[ ]) that stands for the possesive pronoun *his* which has NP label 4. There is no containdexed NP with this pronoun — the first list of the compound object is empty. The possesive pronoun *his* with NP label 4 has only one possible antecedent lexical NP, namely the NP with label 13, therefore the second list contained in the compound object is [13]. There is no other pronominal that can be coreferential with this pronoun, therefore the third list of the compound object is empty.

The third list of (38) contains two compound objects, namely, np(3,[ ]) and np(13,[3]). The first of these compound objects stands for the lexical NP *his mother* which has label 3. There is no other NP of the sentence that is contraindexed with the lexical NP. The second of these compound objects stands for the lexical NP *John* which has label 13. There is only one NP that is contraindexed with the lexical NP, namely the NP with label 3.

12. Comparison with other systems

The behaviour of our system is examined using the following six sentences, taken from the literature:

That he$_i$ had done something terrible was disturbing to John$_i$ (39)

That he$_i$ had done something terrible disturbed John$_i$'s teacher (40)

That he$_i$ had done something terrible disturbed the teacher who punished John$_i$ (41)

Mary sacked out in his$_i$ appartment before Sam$_i$ could kick her out (42)

Girls who he$_i$ has dated say that Sam$_i$ is charming (43)

When he$_i$ is happy, John$_i$ sings (44)

The first five of these sentences were presented in Hobbs (1978, p. 322). Hobbs supposes that the pronoun with subscript i has as its possible antecedent the NP with the same subscript in sentences (39), (42) and (43). He also supposes that the pronoun with subscript i cannot have as its possible antecedent the NP with the same subscript in sentences (40) and (41). Our point of view regarding sentences (40) and (41) is different. We believe that the pronoun can have as its possible antecedent the coindexed — with the subscript i — NP. Hobb's naive algorithm finds the NP as a possible antecedent of the pronoun with the same subscript i only in sentence (39). Thus, according to his opinion, Hobbs' algorithm does not work properly in the case of sentences (42) and (43), while, in ours, it does not work properly also in the case of sentences (40) and (41).

Rich and LuperFoy (1988) observed that although many theories of anaphora resolution exist, neither one is complete. Based on this observation, they implemented a blackboard system in which individual partial theories interact in order to propose candidate antecedents. The modules that they have implemented and which interact in their blackboard system are: Recency, Number agreement, Gender agreement, Animacy, Disjoint reference, semantic type consistency, Global focus and Cataphora. Most of these modules can be integrated in our system in order to eliminate the possible antecedents of an anaphor or pronominal. Specificaly, we could test which of the possible antecedents proposed by our system is compatible in number, gender and animacy with the pronominal or anaphor under examination. Subsequently, we could test which of the remaining possible antecedents are compatible with the semantic type (selectional) restrictions that the verb (of the sentence where the pronominal or anaphor exists) imposes to the part of sentence that the pronominal or anaphor fills. The recency and global focus tests could also be used in order to rank the possible antecedents of a pronominal or an anaphor that passes the above possible tests. Rich and LuperFoy's disjoint reference module which proposes antecedents for reflexive pronouns and impossible antecedents for non-R-pronouns is a weak version of the system proposed in this paper. The cataphora module of their system knows about a class of syntactic constructions in which a pronoun can preceed the full lexical NP to which it corefers. Our system does not differentiate among cases where the pronoun preceedes or follows its antecedent and, therefore, it does not need a separate cataphora module. To clarify this, let us now analyze sentence (44) which is the only example resolved by the cataphora module in Rich's and Luperfoy's paper. According to Williams (1974, 1975) prepositional phrases composed by a *when, before, after* preposition and a sentence are always attached to the main sentence. Reinhart (1983) points out that sentence attached PPs, when preposed, are attached to a node, s", higher than s'. Thus, according to Reinhart (1983, p. 70), the parse tree of sentence (44) is (45):

$$[_{s''}[_{pp}[_{p}\text{when}][_{s'}[_{comp}\ ][_{s}\text{ he AGR is happy}]]][_{s'}[_{comp}\ ][_{s}\text{John AGR sings}]]] \qquad (45)$$

The labeled (Prolog) parse tree which is equivalent to tree (45) and which is analysed by our anaphor and pronominal algorithm is the following:

```
s__(0, pp(1, p(2,when(3)),
             s_(4, comp(5),
                   s(6, np(7,pron(8,he(9))),
                        agr(10),
                        vp(11,v(12,be(13)),
                          ap(14,a(15,happy(16))))))),
       s_(17, comp(18),
              s(19, np(20,john(21)),
                    agr(22),
                    vp(23,v(24,sing(25))))))
```

(46)

The governor of the pronominal *he* which has NP label 7 is the agreement node *agr* which has label 10. The governing category is the innermost S or NP node that includes the pronominal NP and its governor, namely, the s node with label 6. There is no other NP inside the governing category. Consequently, there is no NP contraindexed with the pronominal. These facts imply the creation of the compound object pron(7,[ ],?,?).

Furthermore, the lexical NP *John* which has NP label 20 is examined. The only NP, other than the NP *John*, in this sentence is the NP which has label 7. This NP does not c-command the lexical NP *John*. Consequently, there is no contraindexed NP with the NP 20 and the compound object, np(20,[ ]), is created.

The fact that neither the lexical NP 20 includes the pronominal NP 7, in its list of contraindexed NPs, nor the pronominal NP 7 includes the lexical NP 20, in its list of contraindexed NPs, results in the instantiation of the third argument of pron(7,[ ],?,?) to the list [20]. Because there is no pronominal NP, other than 7, in the sentence, the fourth argument of pron(7,[ ],?,?) is instantiated to the empty list. Execution of our algorithm produces the three lists (47) below. These lists contain information about the anaphor, the pronominal and the lexical NPs of the sentence respectively.

```
[ ]
[pron(7,[ ],[20],[ ])]        (47)
[np(20,[ ])]
```

Consequently, our system proposes the lexical NP *John* as a possible antecedent of the pronominal NP *he*.

13. Conclusions

A theory that suggests the possible antecedents of a pronominal or an anaphor has been suggested. A system which is based on this theory was implemented in Edinburgh compatible Prolog. This system overcomes the problems of existing linguistic theories, by providing two different definitions for the governing category of both anaphors and pronominals. Another advandage of the system presented is that the possible antecedents of a pronominal can be found, whereas existing theories provide rules for finding the impossible antecedents only. Finaly, using already known filters, the system presented can be further developed, so that it can eliminate possible additional antecedents of anaphors and pronominals.

References

N. CHOMSKY *Lectures on Government and Binding*. Foris Publications, Dordrecht, 1981.

R. J. HOBBS Resolving pronoun references. *Lingua*, **44** (1978), 311–338.

C. T. J. HUANG A Note on the Binding Theory. *Linguistic Inquiry*, **14** (1983), 554–561.

J. P. R. INGRIA, D. STALLARD A computational mechanism for pronominal reference. Presented at the Proceedings of the 27th Annual meeting of the Association of Computational Linguistic, (1989), 262–271.

A. RADFORD *Transformational syntax*. Cambridge University Press, Cambridge, 1981.

T. REINHART Anaphora and Semantic Interpretation. Croom Helm, London, 1983.

E. RICH, S. LUPERFOY An Architecture for Anaphora Resolution. Presented at the Proceedings of the second conference on Applied Natural Language Processing, Austin, Texas, 1988.

H. C. V. RIEMSDIJK *Introduction to the theory of grammar*. Massachusetts Institute of Technology, 1986.

E. S. WILLIAMS *Rule ordering in syntax*. Ph.D. dissertation, MIT, Cambridge, Mass, 1974.

E. S. WILLIAMS Small clauses in English. In *Syntax and Semantics* (J. P. KIMBALL, Ed.), vol. 4 (1975), pp. 249–273. Academic Press, New York.

Received: January, 1993
Accepted: November, 1993

Contact address:
Department of Informatics,
Athens University of Economics and Business,
76 Patission St., Athens 104 34, GREECE.
Phone: +30-1-8225268
Email: karanikolas@aueb.ariadne-t.gr
Fax: +30-1-8226204

Nikitas N. Karanikolas is a Ph.D. student of the Department of Informatics at the Athens University of Economics and Business. He is a holder of a Degree in Information Science, from the same University. He had also been a visiting research student in the London School of Economics. His current research interests lie in the Computational Linguistics, Natural Language Understanding, Information Retrieval, Interactive Multimedia and Distance Learning.