# A Computational Approach to Binding Theory*

Alessandra Giorgi, Fabio Pianesi, Giorgio Satta**
Istituto per la Ricerca Scientifica e Tecnologica, 38050 Povo (Trento), Italy
e-mail: satta@irst.it ; satta@irst.uucp

*Abstract*

This paper is a first step towards a computational account of Binding Theory (BT). Two algorithms that compute, respectively, Principle A and B have been provided. Particular attention has been devoted to possible interactions of BT with other modules of the linguistic theory, such as those ruling argumental chains. Finally, the computational complexity of the algorithms has been studied.

## 1 Introduction

This work is a contribution to the computational study of the referential properties of Noun Phrases (NPs). In particular, it focuses on the disjoint reference constraint for pronouns, and on the binding requirement for anaphors.[1] Unlike other authors, we do not output actual references for pronouns.[2] The reasons for such a move will be discussed below.

In pursuing these goals, we will refer to Binding Theory (henceforth BT), as developed within the Government and Binding framework by Chomsky and his collaborators (see Chomsky, 1981, 1986); in particular, algorithms will be presented that compute Principles A and B of BT.

Section 2 presents a brief introduction to BT. In Section 3, we will introduce a formal (computational) apparatus, which will then be used to formulate the algorithms. Finally, some considerations about their formal properties will be discussed. Section 4 illustrates, by means of an example, how the algorithms work. Finally, in Section 5, our approach and results will be compared with those already present in the literature.

*Every part of this work has been elaborated jointly by the three authors. However, as far as legal requirements are concerned, A.Giorgi takes responsability of Sections 1 and 2, F.Pianesi of Sections 4 and 5 and G.Satta of Section 3.

**This work has been done while G.Satta was completing his Doctoral Dissertation at the University of Padova (Italy).

[1]This work is part of a larger one providing a computational account of Binding Theory. See Giorgi, Pianesi, Satta (1989a), in which the present approach is extended to principle C of Binding Theory and weak-crossover core cases, and Giorgi, Pianesi, Satta (1989b), where the general problems of Binding Theory verification and satisfiability are addressed.

[2]See Berwick and Weinberg (1984), Correa (1988), Ingria and Stallard (1989), Berwick (1989); for a different approach see Ristad (1989).

## 2 Introduction to Binding Theory

Binding Theory (BT) is a module of the Government and Binding theory ruling the distribution and the referential properties of anaphors (such as *himself*), pronouns (such as *him* and *his*) and R(eferential)-expressions (such as *John, the man I met yesterday, my sister*, etc). Here we will briefly illustrate its scope, without entering into a detailed analysis.

It is a well known fact that lexical items, such as Noun Phrases, must undergo an interpretation process by which they are assigned a referent. Such a process is ruled by principles that vary according to the nature of the item in question, i.e. anaphor, pronoun or R-expression. A first generalization can be stated as follows: anaphors must have an antecedent in the syntax, i.e. in the same sentence where they appear; pronouns *can* directly identify a referent in the world or in the previous discourse; R-expressions are intrinsically referential, i.e. they need no antecedent.[3] Consider the following examples:

(1)a. John$_i$ loves himself$_i$
b. *I love himself

In (1)a. the anaphor *himself* takes *John* as an antecedent, i.e. in technical terms, it is bound by it; in (1)b, for morphological reasons, *I* cannot work as an antecedent for *himself*, so that the whole sentence is ruled out.[4]

Consider now what happens in the case of pronouns:

(2) John thinks that Mary likes him

*him* can either refer to *John* or to someone else in the world, for instance to someone mentioned in the previous discourse. The conclusion up to this point can be summarized as follows: an anaphor *must* have an antecedent, a pronoun *can* have one, an R-expression cannot. However further properties must be taken into account; let us consider pronouns again:

(3) John$_i$ likes him$_{*i}$

*Him* cannot be interpreted as *John*, contrary to what happens in (2): there is a "negative" condition on

[3]For reasons of space, the referential properties of quantified expressions and those of the so-called epithets are not considered here.

[4]R-expressions cannot be coindexed with any c-commanding item (see below for a definition of c-command). Consider the following example:
(i) John likes John
In this case, given that the first R-expression c-commands the second one, the two occurrences of *John* must refer to different persons.

pronouns, since they *cannot* have an antecedent lying within a certain context. Technically speaking, a pronoun must be *free* in its *local domain*; a similar locality condition holds with respect to anaphors, since an anaphor must have an antecedent inside a certain domain:

(4)a. *John$_i$ thinks that Mary likes himself$_i$
b. Mary thinks that John$_i$ likes himself$_i$

(4)a. is ungrammatical because the intended antecedent, *John*, lies too far away.

The *local domain* is the so-called Complete Functional Complex (CFC) pertaining to the item in question, i.e. containing both the item itself and its governor:[5]

(5) $\gamma$ is a Complete Functional Complex iff one of the following holds:
a) $\gamma$ is the (minimal) domain in which all the $\theta$-roles pertaining to a lexical head are realized;
b) $\gamma$ is the (minimal) domain in which all the grammatical functions pertaining to that head are realized.

Finally, an important structural condition must hold, namely c-command: (cf. Chomsky, 1981; 1986):

(6) $\alpha$ c-commands $\beta$ iff $\alpha \neq \beta$, $\alpha$ does not dominate $\beta$ and the first node, $\gamma$, dominating $\alpha$ also dominates $\beta$.

To be *bound* by a given item actually means: to be coindexed *and* c-commanded by that item; whereas to be *free* means: not to be coindexed with a c-commanding item (non c-commanding items might work, on the other hand, as possible antecedents). The principles of binding can be expressed as follows:

(7) A: An anaphor is bound in its local domain
B: A pronoun is free in its local domain
C: An R-expression is free

Note that, as will be shown below, in our system we can also handle constructions involving the so-called *pro*-drop phenomena, found in languages like Italian, Spanish and so on. Consider the following Italian example:[6]

(8) *pro*$_i$ arriva lui$_i$
lit: arrives he

The system must know that the coindexation in (8), i.e. the fact that the pronoun *lui* is coindexed with a c-commanding empty category (the expletive *pro*), is not a Principle B violation.

## 3 The Algorithms

With respect to the two principles considered here, i.e. Principles A and B, the output of BT can be represented as a formal language $L_B$. More precisely, given a sentence $w$, let $T$ be the set of all tuples $t = <\tau_w, \alpha, \beta_1, ..., \beta_n>$, $n \geq 0$, where $\tau_w$ is a parse tree for $w$, $\alpha$ is either an anaphor or a pronoun and the components $\beta_1 ... \beta_n$ represent any set of NPs.[7] Let us define $L_B \subseteq T$ to be the set of all tuples such that the following conditions hold:

*(i)* if $\alpha$ is an anaphor, $\beta_1 ... \beta_n$ are all and only the items that can be antecedents for $\alpha$, according to Principle A of BT;

*(ii)* if $\alpha$ is a pronoun, then $\beta_1 ... \beta_n$ are all and only the items disjoint from $\alpha$, according to Principle B of BT.

The algorithms to be presented can be seen as recognisers for $L_B$.

### 3.1 Definitions

Let $\mathcal{N} = \{n_1, ..., n_q\}$, $q \geq 1$, be the set of all *nodes* in $\tau_w$. We will also assume the following functions and predicates:[8] a function *father* from $\mathcal{N}$ to $\mathcal{N} \cup \{\perp\}$; a function *siblings* from $\mathcal{N}$ to $\mathcal{P}(\mathcal{N})$; a binary predicate *agreement*, defined on $\mathcal{N} \times \mathcal{N}$, such that *agreement*$(n_1, n_2)$=TRUE iff the agreement features of $n_1$ and $n_2$ are mutually compatible; a unary predicate *pt-antecedent*, defined in $\mathcal{N}$, such that *pt-antecedent*$(n)$=TRUE iff $n$ is a maximal projection of a head $N^0$ (a Noun) within $\tau_w$.

**Definition 1** A binary predicate *domain* is defined in $\mathcal{N} \times \mathcal{N}$ in such a way that *domain*$(n_d, n_\alpha)$=TRUE iff:

*(i)* $n_d$ is the least constituent such that either all the *$\theta$-roles* pertaining to a lexical head are realized, or all the *grammatical functions* pertaining to the same lexical head are realized;

*(ii)* $n_\gamma$ is the *lexical governor* of $n_\alpha$ and *father*$(n_\gamma) \neq$ *father*$(n_d)$.[9] □

Condition *(ii)* has been explicitely introduced in order to take care of cases of *government across the boundaries* (see Section 3.3).

To account for the interaction of BT with *pro*-drop (cf. ex.(8) above), we also need the following definitions. Let *ch-mark* be a procedure defined on $\mathcal{N} \times \mathcal{N}$: whenever *ch-mark*$(n, n_m)$ is invoked, if $n$ is a landing-site of a chain[10] $c$ within $\tau_w$, every node $n_c$ such that $n_c$ belongs to $c$, gets marked with a distinctive marker, which will be assumed to be the second argument node $n_m$. This marking relation will hold until a new call to the procedure takes place for any node corresponding to the same chain as $n$, with a different node-marker. We need

[5]The definition in the text is the one found in Giorgi (1987); cfr also Chomsky (1986). Note that, in most cases, the CFC coincides with the first Sentence or Noun Phrase dominating the item in question. However, this is not always the case and the systems defining the binding domain this way often run into trouble; this point will be further considered in Section 5; see also Giorgi, Pianesi, Satta (1989a).

[6]In *pro*-drop languages (see Chomsky 1981) typically, the subject can be non-lexical, i.e. can be an empty category, or can be expressed postverbally, leaving an expletive empty category in subject position.

[7]We assume that all the principles of the theory have already been applied to the sentence. Such an assumption is reasonable, given the modular nature of the theory; see Chomsky (1981, 1986).

[8]In the following, the symbol $\perp$ is meant to denote the undefined element. Also, for a generic set $\mathcal{A}$, $\mathcal{P}(\mathcal{A})$ denotes the set of all possible subsets of $\mathcal{A}$ (the *power set* of $\mathcal{A}$).

[9]The exact meaning of linguistic notions such as: $\theta$-roles, functional-roles, lexical governor and government can be found in Chomsky (1981, 1986). A computational account of the notion of local domain can be found in Giorgi, Pianesi and Satta (1989a), along with some formal properties of the predicate *domain*.

[10]Roughly, the notion of *chain* can be defined (cf. Chomsky, 1986) as the set of coindexed positions (*landing sites*) pertaining to the same syntactic object (where only one of such positions is lexically filled).

also a function *ch-marker*, from $\mathcal{N}$ to $\mathcal{N}\cup\{\perp\}$, defined such that *ch-marker*$(n)=n_m$ iff $n$ is a landing-site of a chain $c$ within $\tau_w$, and a previous call to the procedure *ch-mark* has marked each node in $c$ with the marker-node $n_m$.

## 3.2 Algorithm Schemata

The two algorithms behave in a very similar way; they take as input a node in $\mathcal{N}$ corresponding to an NP in $\tau_w$, and analyse some specific relations between the input node and each node in $\mathcal{N}$ that c-commands the input node, up to certain specific domain. The c-commanding relation is implicitly encoded in the way in which the algorithms apply the two functions *father* and *siblings*.

*An Algorithm for Principle A*

Given an input node $n$ which corresponds to an anaphor in $\tau_w$, the algorithm outputs a list of nodes corresponding to "actual antecedents" for the anaphor itself. The algorithm looks for a "potential antecedent" of the input anaphor, starting from node $n$ and proceeding from bottom-to-top. As soon as a potential antecedent is found, the algorithm restricts its search to the local domain it has just identified. Note that each potential antecedent must pass the agreement check to be considered an actual antecedent.[11]

We also consider some cases of referential circularity; in particular, problems arising in *pro*-drop constructions. More specifically, a node which belongs to the chain also containing the anaphor, cannot be collected as a potential antecedent. The following *circularity check* is therefore included: every chain whose landing-sites dominate the input anaphor, up to the domain of interest, is marked by the procedure *ch-mark* using the input-node as a marker. In this way a node c-commanding the input node and corresponding to a landing-site of a chain marked by the latter, cannot be taken as a potential antecedent for the input-node itself (for more discussion, see Section 5). The same mechanism also ensures that, for every possible chain, only one of its landing-sites is ever considered as a potential antecedent.

**Algorithm 1**

*Input-node:* A node corresponding to an anaphor in $\tau_w$.
*Output*: A list of nodes in $\mathcal{N}$ corresponding to actual antecedents for the input anaphor.
*Method.*

```
Step 1: Let input-node be the value of the program
    variable present-node. Initialize also the program
    variable local-domain-flag to the value FALSE and
    invoke the procedure ch-mark(present-node, input-node).
Step 2: For each value of the program variable present-sibling
    in siblings(present-node), if ch-marker(present-sibling)≠input-node
    and pt-antecedent(present-sibling)=TRUE, perform the following
    actions. Set the program variable local-domain-flag to TRUE if it
    is FALSE and invoke the procedure ch-mark(present-sibling,
    input-node); furthermore, if agreement(present-sibling,
    input-node)=TRUE then output present-sibling.
Step 3: If father(present-node)=⊥, go to Step 4,
    otherwise let father(present-node) be assigned as the
    value of present-node. Invoke the procedure ch-mark(present-node,
    input-node). If local-domain-flag=FALSE then restart at Step 2.
    Otherwise there are two possibilities: if domain(present-node,
    input-node)=FALSE then restart at Step 2; if
    domain(present-node, input-node)=TRUE go to Step 4.
Step 4: Stop.                                              □
```

*An Algorithm for Principle B*

The algorithm starts from an input node that corresponds to a pronoun in $\tau_w$. The algorithm visits all nodes in $\mathcal{N}$ which correspond to elements c-commanding the input pronoun and lie inside the local domain; finally, it outputs a list of disjoint elements. Indeed the algorithm is procedurally very similar to the one given for Principle A, with minor changes due to the differences in the definitions of the local domain. Algorithm 2 considers each chain only once, as does Algorithm 1. Observe that if a pronoun belongs to a certain chain, it cannot be disjoint from other elements of the same chain. An *identity check* is then carried out by the algorithm in the following way: the chain, which the input-node belongs to, is initially marked by the procedure *ch-mark*. Then, every node that c-commands the input-node inside its local domain, corresponding to a landing-site of this marked chain, will not be inserted in the output list of Algorithm 2 (see Section 5). The details are the following:

**Algorithm 2**

*Input-node:* A node corresponding to a pronoun in $\tau_w$.
*Output*: A list of nodes in $\mathcal{N}$ corresponding to the disjoint elements for the input pronoun.
*Method.*

```
Step 1: Let input-node be the value of the program
    variable present-node. Invoke the procedure chain-
    mark(present-node, input-node).
Step 2: For each value of the program variable present-
    sibling in siblings(present-node), perform the
    following action. If ch-marker(present-sibling)≠input-node
    and pt-antecedent(present-sibling)=TRUE, output
    present-sibling and invoke the procedure ch-
    mark(present-sibling, input-node).
Step 3: If father(present-node)=⊥, go to Step 4,
    otherwise let father(present-node) be assigned as the
    value of present-node. If domain(present-node, input-
    node)=FALSE then restart at Step 2, otherwise go to
    Step 4.
Step 4: Stop.                                              □
```

## 3.3 Some Formal Results

Some properties of Algorithms 1 and 2 will be stated; see also Giorgi, Pianesi and Satta (1989a).

**Theorem 1** The predicate *domain(present-node, input-node)* holds true at Step 3 in Algorithms 1 and 2 iff *present-node* corresponds in $\tau_w$ to the minimal CFC containing both *input-node* and $n_\gamma$, where $n_\gamma$ is the lexical governor of *input-node*.

[11] According to Chomsky (1986), the existence of the potential antecedent for an anaphor is crucial in defining its local domain. Note that such an item is not necessarily the actual antecedent.

*Proof*
*'Only if'* Condition *(i)* in Definition 1 guaranties that *present-node* is a CFC, as defined in (5). Furthermore *present-node* dominates *input-node* at Step 3, as it is easy to show. It remains to demonstrate that *present-node* dominates $n_\gamma$. A government relation between $n_\gamma$ and *input-node* can only be attained within the following three structural configurations. In the first, government is realized under sisterhood; thus, every node that dominates the governee will also dominate the governor. In the second configuration the governee is attached higher than its governor, within the maximal projection of the latter; again, every node that dominates the former will also dominate the latter. The third possibility concerns the so called *government across boundaries*: when a maximal projection ZP (or a Small Clause) is in sisterhood relation with a lexical category $X^0$, then the latter can govern the specifier position of the former (or the subject position, in the case of a Small Clause). ZP may well be a CFC, in the sense of (5), but it does not contain the governor $X^0$. Condition *(ii)* in Definition 1 explicitely rules out this case, so the claim is proved.
*'If'* The proof immediately follows from the given analysis of the possible configurations of government between the nodes $n_\gamma$ and *input-node*, and from Definition 1. ❑

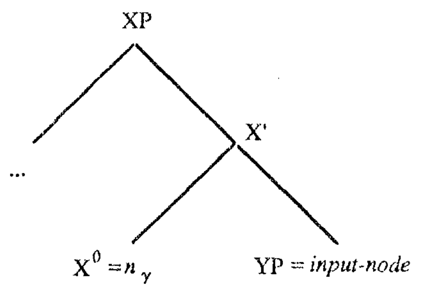

Figure 1

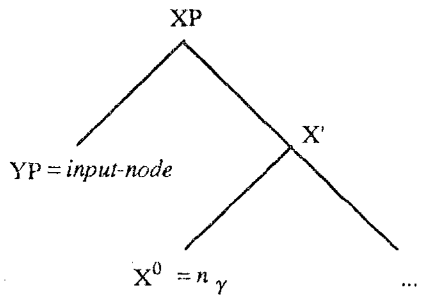

Figure 2

**Theorem 2** Let $\tau_w$ be an X-bar description for some sentence $w$ such that all the principles of GB hold true for it, apart from the BT principles, and let $\mathcal{N}$ be the set of all nodes in $\tau_w$. When *input-node* is assigned a value, which corresponds to an anaphor $\alpha$ in $\tau_w$, Algorithm 1 computes the whole list of nodes in $\mathcal{N}$ that corresponds to the antecedents of $\alpha$, in the sense of Principle A of BT.
*Proof* Omitted. ❑

**Theorem 3** Let $\tau_w$ and $\mathcal{N}$ be as in Theorem 2. Given as input a node that corresponds to a pronoun $\alpha$ in $\tau_w$, Algorithm 2 computes the whole list of nodes in $\mathcal{N}$ that must be disjoint from $\alpha$, in the sense of Principle B of BT.
*Proof* Omitted. ❑

Questions about time complexity are now addressed for Algorithms 1 and 2 (we assume, as the reference model for computation, a RAM).

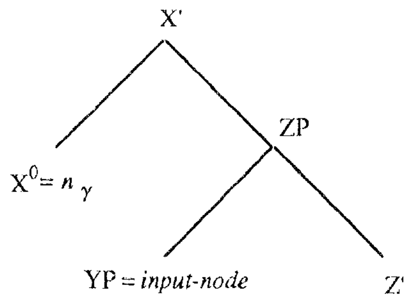

Figure 3

**Theorem 4** The running times of Algorithms 1 and 2 are given by two functions $f_{A_1}$ and $f_{A_2}$, such that $f_{A_1}$, $f_{A_2} \in O(n)$, where $n$ is the length of the sentence under analysis.[12]
*Proof (outline)* From elementary considerations about X-bar Theory,[13] it can be argued that set $\mathcal{N}$ has size bounded by an expression of the form $c_1n+c_2$. It is easy to show that no node in $\mathcal{N}$ is visited more than once by Algorithms 1 and 2 and that a constant amount of time is spent in visiting each node; then the result follows. ❑

## 4 A running example

Let us see how Algorithm 1 works with the following sentence, giving, as input node, the one corresponding to the anaphor *herself*:

(9) $[_{IP}$ Mary $[_{I'}$ $[_{VP}$ $[_{V'}$ sees herself $[_{PP}$ in the mirror]]]]] [14]

At the beginning, the variable *present-node* is set to the value of *input-node*, i.e. the anaphor node, while the variable *local-domain-flag* is set to FALSE. Then Algorithm 1 enters Step 2, where it scans the anaphor's siblings. Once *present-sibling* is set to the PP node, *pt-antecedent(present-sibling)*=FALSE. At this point, Algorithm 1 exits Step 2 and enters Step 3, where *present-node* is set to the value of *father(present-node)*, in this case, the V' node. Given that *local-domain-flag*=FALSE, Algorithm 1 enters Step 2 again. We do not follow the whole computation but directly skip to the point where Algorithm 1 enters Step 2 with *present-*

[12]The standard notation $g(n) \in O(f(n))$ means that there exist a positive constant $M$ and an integer $n_0$ such that $g(n)<Mf(n)$ for all $n \geq n_0$.
[13]See Jackendoff (1977).
[14]Another possible analysis of this sentence hypothesises, as the argument of *see*, a Small Clause including *herself* and the predicate PP.

*node*=I'. The only sibling of I' is the subject NP and, setting *present-sibling* to it, one has that *pt-antecedent(present-sibling)*=TRUE. Algorithm 1 then sets *local-domain-flag* to TRUE; furthermore, *agreement(present-sibling, present-node)*=TRUE, so that the value of *present-sibling* is output. After that, Algorithm 1 enters Step 3 and sets *present-node* to *father(present-node)*, i.e. to the topmost (IP) node. Now *local-domain-flag*=TRUE and *domain(present-node, input-node)*=TRUE (i.e. IP is the local domain for the input anaphor); therefore Algorithm 1 enters Step 4 and then stops.

## 5 Discussion

By fixing an upper bound, it has been shown that the computational complexity of the recognition problem of a language that encodes Principles A and B of BT is in $\mathcal{P}$.[15] These results are similar to those obtained by all authors who have studied BT from a computational point of view (Correa, 1988; Ingria and Stallard, 1989; Berwick and Weinberg, 1984; Berwick, 1989). Nevertheless, with respect to such works we have both taken a rather different perspective and paid more attention to the subtleties of the linguistic theory. Previous works were mainly concerned with providing actual referents (actual indexations) for the NPs of a sentence. We claim, on the contrary, that Principles A, B and C *per se* are not sufficient for this purpose, since BT only restricts the search space for indices selection, and does not actually provide them. For instance, Correa (1988) proposes an algorithm that builds lists of antecedents for pronouns and anaphors, and complements it with a Binding Rule, that selects, for each item, an indexation from such lists. However, the selection of an antecedent for a certain item could affect the indexation of other nodes, leading to violations of Principle B.[16] In this framework, a related problem arises considering split antecedents for pronouns; in fact, a pronoun can be coindexed with a set of items, provided that each of them has a different *thematic role*.[17] This point has never been explicitely addressed in computational works; nevertheless, if the purpose is to output actual indexations, it seems to us that the only possibility, in order to consider split antecedents, would be to compute lists of possible antecedents for pronouns and then to consider their power set; this way, however, the search space becomes exponentially large. Furthermore, the interactions of the referential properties of a split antecedent with those of other items (possibly other split antecedents) would thereby hardly be addressable.[18]

Finally, given the referential properties of pronouns , it seems that there is no point in trying to use the grammatical knowledge of BT to hypothesise intrasentential antecedents.[19]

Another crucial aspect concerns the treatment of local domains, whose importance has often been misconsidered in computational works on BT. Such a notion has been mainly seen as a static one, whereas, in our interpretation, the value of the actual domain depends on the interaction of structural and lexical properties of at least two different positions in the derivation tree. For example, in Ingria and Stallard (1989), an S node is taken to be the binding domain for every node it dominates. Consider, however, the cases in which government of the specifier position of a maximal projection is obtained through an external head; this situation arises, for instance, in exceptional case marking examples, as in *John believed him to be intelligent.* Ingria and Stallard's static definition of local domain would lead to the conclusion that the pronoun, being dominated by an S type node (the embedded sentence), is free in that category and, thus, could be coindexed outside, for instance with the R-expression *John*. But this is ungrammatical; according to the definition adopted here, the domain of binding for *him* is the matrix clause, so that the pronoun must, correctly, be free in it, i.e. disjoint from *John*. Our approach also improves on Ingria and Stallard's treatment of NP as a binding domain. If a node NP containes a possessive then they consider it a binding domain for all the nodes it dominates, except the possessive itself. There are at least two problems, though. First, they do not predict that a pronoun subcategorized for by the head cannot be bound in the domain of the NP; second the well-known not complementary distribution of pronouns and anaphors in the specifier position of an NP cannot be accounted for. The definition of binding domain adopted in (5) and the way it is computed allow our algorithms to avoid these problems; see Giorgi, Pianesi, Satta (1989a).

As a final remark, note that in this work the interaction with A-chains has been explicitely considered. This problem is particularly important in Italian which, being a *pro*-drop language, admits sentences like (8) and (10):

(10) Gianni$_i$ ha detto che *pro*$_j$ arriverà [la propria$_i$ madre]$_j$
lit.: Gianni told that will arrive self's mother
Gianni told that his mother will arrive

[15]$\mathcal{P}$ denotes the class of languages recognizable in polynomial time by a deterministic Turing machine.

[16]Consider, for instance, the following sentence:

(i) *Mary says that she saw her*

in this case both embedded pronouns can take *Mary* as an antecedent, according to Correa's Binding Rule, leading to a violation of Principle B. In our opinion, to avoid this incorrect result, it is necessary to put together the constraints that have been separately computed for each item according to Principles A and B (and C); this way we can account for the interactions between coindexations and disjointness. A possible way to do it, is to pose the problem of BT verification, i.e. whether a given index assignment for the NPs of a sentence complies with the restrictions of BT. See Giorgi, Pianesi, Satta (1989b).

[17]For instance, in *Mary told John that they should go home* the pronoun *they* can refer to the complex antecedent constituted by *Mary* plus *John*.

[18]Also the so called weak crossover phenomena may raise some problems. Roughly, pronouns cannot be coindexed with non c-commanding quantified expressions, as in **His*$_i$ *mother loves* [*every boy*]$_i$, where the embedded pronoun cannot be taken to refer to the quantified expression. But this fact raises some problems for both Correa's and Ingria's approach.

[19]Pronouns can refer intersententially or deictically; note that this property is shared with certain R-expressions, like the epithets, which obey Principle C (see Haïk, 1984).

in (8) the postverbal subject pronoun is coindexed with the expletive *pro*, but a procedure looking for disjoint elements would output a list containing *pro* (it lies in the local domain of the pronoun) thereby giving rise to a contradiction: the pronoun is coindexed with the expletive *pro* but must be disjoint from it. In (10), we must avoid the anaphoric possessive *proprio* being coindexed with the c-commanding expletive *pro*, in order to rule out circular interpretations. The circularity and identity checks, discussed in Section 3.2, explicitly take care of these cases.

## References

Berwick R.,(1989) , Natural Language Computational Complexity and Generative Capacity, to appear on *Computers and Artificial Intelligence*

Berwick R. and Weinberg A., (1984), *The Grammatical Basis of Linguistic Performance*, MIT Press, Cambridge MA

Chomsky N., (1981), *Lectures on Government and Binding*, Foris, Dordrecht

Chomsky N., (1986), *Knowledge of Language*, Praeger, New York

Correa N., (1988), A binding Rule for Government-binding Parsing, *Proceedings of COLING*, Budapest

Giorgi A., (1987), The notion of Complete Functional Complex: Some Evidence from Italian, *Linguistic Inquiry*, 18,3

Giorgi A., Pianesi F., Satta G.,(1989a), Towards a Complexity Analysis of Binding Theory, Technical Report n.8911-06, IRST, Trento, It.

Giorgi A., Pianesi F., Satta G.,(1989b), The Computational Complexity of Binding Theory's Satisfiability and Verification, ms, IRST, Trento, It

Haïk I., (1984), Indirect Binding, *Linguistic Inquiry*, 15, 2

Ingria R.P.J., Stallard D.,(1989), A Computational Mechanism for Pronominal Reference, Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics, Vancouver

Jackendoff R., (1977), *X-bar Syntax: A study of Phrase Structure*, MIT Press, Cambridge MA