A COMPUTATIONAL MECHANISM FOR PRONOMINAL REFERENCE

Robert J. P. Ingria
David Stallard
BBN Systems and Technologies, Incorporated
10 Moulton Street
Mailstop 009
Cambridge, MA 02238

ABSTRACT

This paper describes an implemented mechanism for handling bound anaphora, disjoint reference, and pronominal reference. The algorithm maps over every node in a parse tree in a left-to-right, depth first manner. Forward and backwards coreference, and disjoint reference are assigned during this tree walk. A semantic interpretation procedure is used to deal with multiple antecedents.

1. INTRODUCTION

This paper describes an implemented mechanism for assigning antecedents to bound anaphors and personal pronouns, and for establishing disjoint reference between Noun Phrases. This mechanism is part of the BBN Spoken Language System (Boisen, et al. (1989)). The algorithm used is inspired by the indexing scheme of Chomsky (1980), augmented by tables analogous to the "Table of Coreference" of Jackendoff (1972). This mechanism handles only intrasentential phenomena and only selects the syntactically and semantically possible antecedents. Ultimately, it is meant to be used in conjunction with an extra-sentential reference mechanism like that described in Ayuso (1989) to include antecedents from other utterances and to utilize discourse factors in its final selection of an antecedent.

In Section 2 the empirical and theoretical background to this treatment is sketched out. In Section 3, the actual algorithm used is described in detail. In Section 4, the associated semantic interpretation mechanism is presented. In Section 5, we compare the algorithm with related work. Finally, in Section 6, remaining theoretical and implementational issues are discussed.

2. THEORETICAL BACKGROUND

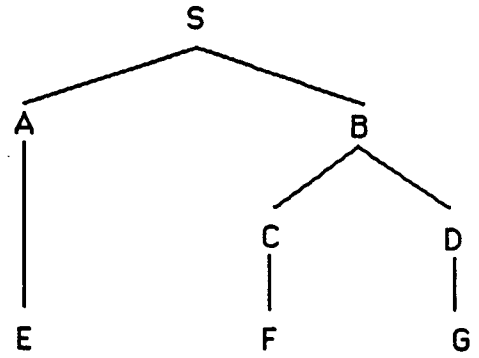
While most computational systems are interested in the potential antecedents of pronouns, work in generative grammar by Lasnik (1976) and Reinhart (1976) has led to the conclusion that sentential syntax is responsible for assigning possible antecedents to bound anaphors (reflexives, such as "himself", "herself", "themselves", etc., and the reciprocals "each other" and "one another") but not to personal pronouns ("he", "she", "they", etc). In the case of personal pronouns, sentential syntax only determines the syntactically *impossible* antecedents. This latter procedure is called *disjoint reference*, since the impossible antecedents can not even overlap in reference with the pronoun; compare the cases in sentences (1) and (2), where the underlined items are non-identical in reference, with those in (3) and (4), where they are non-overlapping in reference. In (1) and (2), "he" and "him" cannot refer to "John" (non-identical reference); while in (3) and (4) "John" cannot be a member of the set referred to by "they" and "them" (non-overlapping or disjoint reference).

(1) He likes John.
(2) John likes him.
(3) They like John.
(4) John likes them.

Disjoint reference is even more noticeable with first and second person pronouns where it does not merely produce impossible interpretations, but actual ungrammaticality:

(5) *I like me.
(6) *I like us.
(7) *We like me.
(8) *You like you.

A crucial notion both for assigining antecedents to bound anaphors and for establishing disjoint reference between Noun Phrases is that of *c-command*, a structural relation. Briefly, a node c-commands its sisters and any nodes dominated by its sisters.[1] Figure 2-1 illustrates this.



A c-commands B, C, F, D, and G
B c-commands A and E
C c-commands D and G
D c-commands C and F

Figure 2-1: C-Command

[1]This differs from Reinhart's (1976) definition, for reasons discussed in Section 6.

Essentially, the relation between c-command and reference phenomena is the following:

1. A non-pronominal NP cannot overlap in reference with any NP that c-commands it.
2. The antecedent of a bound anaphor must c-command it.[2]
3. A personal pronoun cannot overlap in reference with an NP that c-commands it.[2]

Condition 1 is motivated by sentences such as those in (9), where the underlined pronouns "he", "him", "they", and "them" must be disjoint in reference with "John". In each case, the pronouns c-command the NP "John". In (9a) "he"/"they" is in the subject position, and so c-commands "John", in the direct object slot. In (9b) the pronouns ("He", "They") are once again in the subject position, and "John" is the object of a preposition, itself contained in the direct object of the sentence. Finally, in (9c), the NP "John" appears as the object of a preposition, which is c-commanded by the subject ("He", "They") and the direct object ("him", "them").

(9) a. He likes John.
They like John.
b. He likes pictures of John.
They like pictures of John.
c. He told them about John.
They told him about John.

Condition 2 is motivated by examples such as those in (10), where the reflexive pronoun "himself" and its antecedent(s) are bracketed. As in the corresponding examples in (9), "himself" either appears as a direct object (10a), the object of a preposition within the direct object (10b), or as a prepositional object (10c). In all cases, the c-commanding subject ("John") is a possible antecedent; in (10c), where the c-commanding object NP "Bill" is added, it is also a possible antecedent.

(10) a. [John] likes [himself].
b. [John] likes pictures of [himself].
c. [John] told [Bill] about [himself].

Condition 3 is motivated by examples such as those in (11). The pronoun under consideration ("him" or "them") always appears as an object or prepositional object and is disjoint in reference to the c-commanding subject "John" (in (11a,b,c)) and to the c-commanding direct object "Bill" in (11c).

(11) a. John likes him.
John likes them.
b. John likes pictures of him.
John likes pictures of them.
c. John told Bill about him.
John told Bill about them.

While condition 1 is unconditionally true, conditions 2 and 3 are subject to a further constraint, which we might term *minimality*. Essentially, the structural theory of pronominal reference outlined here may be viewed as making the following claim. Bound anaphors are short-distance anaphors and require their antecedents to be c-commanding NPs within a minimal domain. Ordinary personal pronouns, on the other hand, are long-distance anaphors, and only permit antecedents to come from outside of their minimal domain, and exclude any c-commanding antecedents within their minimal domain. The most immediately dominating finite clause (S) node always constitutes a minimal domain for a bound anaphor or personal pronoun. NP nodes normally do not constitute a minimal domain, unless they contain a possessive. This is illustrated in (12)--(14) (underlining indicates disjoint reference; bracketing indicates co-reference). The subject NP in (13) is not a possible antecedent for the reflexive; while the subject NP in (14) need not be disjoint in reference with the underlined pronoun. Compare (13) with (10b) and (14) with (11b).

(12) He likes Bill's pictures of John.
They like Bill's pictures of John.
(13) John likes [Bill's] pictures of [himself].
(14) [John] likes Bill's pictures of [him].
[John] likes Bill's pictures of [them].

Given these paradigms of reference facts, we now turn to the theoretical linguistics literature for treatments that might be implemented in a natural language system. In the Government-Binding framework of Chomsky (1981), these generalizations are captured by the Binding Theory—a set of well-formedness conditions on syntactic structural representations annotated with subscript and superscript "indices". The paradigm assumed there is Generate and Test: indices are freely assigned and the Binding Conditions are applied to rule in or rule out a particular assignment. Clearly, from a computational standpoint this is grossly inefficient. However, in earlier work, Chomsky (1980, pp. 38--44) proposed a two pass indexing mechanism that captures these facts procedurally.

His proposal assigns each non-bound anaphor (i.e. non-pronominal NP or personal pronoun) the pair (r,A) where r (for *R*eferential index) is a non-negative integer and A (for *A*naphoric index) is a set of such integers. In the first pass, r and A are assigned from left-to-right in a depth-first manner. Each non-bound anaphor NP is assigned a unique r; in addition, the r index of each NP c-commanding it is added to its A index. This set of indices indicates all the other NPs with which it is disjoint in reference. For non-pronominal NPs, only one pass is needed:

(15) John_2 told $\text{Bill}_{(3,\{2\})}$ about $\text{Fred}_{(4,\{2,3\})}$

The indices here indicate that "John", "Bill", and "Fred" are all disjoint in reference.

In the case of personal pronouns, a second pass is necessary. Consider example (14), repeated here as (16), after the first pass:

[2]Within a minimal syntactic domain; this will be explained shortly.

(16) John$_2$ likes Bill's$_{(3,\{2\})}$ pictures of him$_{(4,\{2,3\})}$

The indexing at this stage indicates that "Bill" is disjoint in reference from "John" and that "him" is disjoint in reference from "Bill", which is correct, and also from "John", which is not. To correct this, Chomsky (1980, pp. 38--44) has a second pass, in which the r indices of NPs outside the current minimal domain are removed from the A index of personal pronouns, thereby allowing them to serve as potential antecedents. After this second pass, the indexing is:

(17) John$_2$ likes Bill's$_{(3,\{2\})}$ pictures of him$_{(4,\{3\})}$

At this stage "John" is no longer specified as being disjoint in reference with "him".

We have taken this procedure as the basis for a more efficient pronominal reference algorithm that improves on two problematic features. First, while Chomsky's procedure requires two passes, our algorithm is single pass. While there may not be a great computational loss in the two-pass character of Chomsky's original proposal, clearly it is cleaner to do things in one pass. Moreover, the mechanism is extensionally richer than Chomsky's: it also handles cases of backwards-pronominalization and split-antecedence.

A second problem with Chomsky's procedure is that the potential antecedents of a personal pronoun are only implicitly represented: any NP whose r index is not a member of that pronoun's A index set is a syntactically permissible antecedent, but this set of permissible antecedents is not enumerated. For example, in (17), "John" is indicated as a potential antecedent of "him" by virtue of the fact that its r index, 2, is not part of the A index of "him", and in no other way. Our algorithm explicitly indicates the potential antecedents of a personal pronoun. Again, this is more desirable than leaving this information implicit; besides the potential (and perhaps small) computational savings of not needing to recompute this information, there is the more general consideration that we are not interested in creating syntactic representations for their own sakes, but to make use of them. Explicitly representing antecedence information for personal pronouns contributes to this goal.

In the next section, we show how our algorithm overcomes these limitations.

3. THE ALGORITHM

Before giving the details of the algorithm, we will sketch its general structure. The algorithm applies to a completed parse tree and traverses it in a left-to-right, depth-first manner. The algorithm uses the notion of minimal domain introduced in the preceding section: 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, and the related notions of "internal" and "external" nodes. Internal nodes are dominated by the current minimal domain node; external nodes c-command the current minimal domain node. Essentially, the algorithm passes each node all the nodes that c-command it, subdivided into two sets, those that are internal to the current minimal domain and those that are external. As each node is processed, a subroutine is called that dispatches on the category of the node and performs any actions that are appropriate. It is this subroutine that implements the pronominal reference mechanism proper.

Given this overview, we can now turn to the data structures that are used by the algorithm, as well as to the details of the algorithm. Each node in a parse tree is a Common LISP structure; two of its slots are used for establishing pronominal reference:
:possible-antecedents—a list of all the nodes that can be co-referent or overlapping in reference with it.
:impossible-antecedents—a list of all the nodes that are disjoint in reference with it.
The algorithm also uses two global variables—***table-of-proforms*** and ***table-of-antecedents***—in a "blackboard" fashion.

The algorithm uses two major procedures. The first, **pass-down-c-commanding-nodes**, is responsible for actually traversing each node in the tree. The actual algorithm it uses is shown in Figure 6-1 in a LISP-type notation. Its functionality can be stated as follows. Whenever it encounters a new node, it first processes that node by calling the procedure **update-node**, which will be described shortly. It next determines whether the node being processed counts as a minimal domain for its children. When the node is a finite S node, it does count as a minimal domain, for all its children. Hence, only nodes that it dominates can be internal nodes for its children; all other nodes are now treated as external by its children. When the node is an NP, there are two possibilities. If there is no possessive NP, the NP does not count as a minimal domain, hence, the external nodes remain as before and the nodes it dominates are added to the set of internal nodes. However, when the NP does contain a possessive, it does count as a minimal domain, for all the nodes that it dominates, except the possessive itself.[3] Finally, if the node is of any other category, it is not a minimal domain, so the external nodes remain as before and the internal nodes are augmented by the constituents it dominates.[4] In all cases, **pass-down-c-commanding-nodes** calls itself recursively on the children of the node being processed, with the appropriate lists of internal and external nodes as arguments.

update-node, in turn, processes the node passed

[3]The reason for this exception will be explained in Section 6.

[4]Non-finite clauses also need special treatment. However, consideration of this case requires discussion of whether non-finite clauses are Ss or VPs, which is beyond the scope of this paper.

to it, on the basis of the nodes internal and external to the current minimal domain. In particular, **update-node** performs the correct pronominal assignment. The algorithm used by **update-node** is shown in Figure 6-2 in a LISP-type notation. We also discuss each clause separately.

Clause [I] implements condition 1 (non-pronominal NPs). Since there are no minimality conditions on disjoint reference for non-pronominal NPs, all NP nodes c-commanding a non-pronominal NP are added to its **:impossible-antecedents** slot, whether they are internal ([I.A]) or external to the current minimal domain ([I.B]). This handles sentences such as those in (9) and (12). While it might seem odd to specify that a non-pronominal NP has no antecedents, this information is useful in handling cases of backwards pronominalization, as in (18).

(18) [His] mother loves [John].

Clause [I.C] handles backwards pronominalization by making use of information in ***table-of-proforms***, a table of all the pronouns encountered so far in the course of the tree walk.[5] After **update-node** has added all c-commanding NP nodes to the **:impossible-antecedents** slot of a non-pronominal NP, it then searches ***table-of-proforms*** for any pronouns that are not on its **:impossible-antecedents** list; whenever it finds one, it adds the current non-pronominal NP to the pronoun's **:possible-antecedents** list. The last thing **update-node** does in processing a non-pronominal NP is to add it to ***table-of-antecedents*** ([I.D]), whose use will be explained shortly.

Clause [II] implements condition 2 (bound anaphors). Since bound anaphors are short-distance anaphors, all and only the c-commanding NPs internal to the current minimal domain are added to the **:possible-antecedents** slot of a bound anaphor.

Clause [III] implements condition 3 (personal pronouns). Since personal pronouns are long-distance anaphors, clause [III] performs a number of operations. First, all the c-commanding NPs internal to the current minimal domain are added to the **:impossible-antecedents** slot of a personal pronoun ([III.A]), disallowing them as antecedents. Next, all the c-commanding NPs external to the current minimal domain are added to the **:possible-antecedents** slot of a personal pronoun ([III.B]), indicating that they are potential antecedents. Clause [III.C] handles sentences like (19).

(19) [John's] mother loves [him].

in which a non-pronominal NP that does not c-command a personal pronoun serves as its antecedent. As was noted above, each non-pronominal NP is added to the ***table-of-antecedents*** by clause [I.D]. When **update-node** has added all the appropriate c-commanding nodes to the **:impossible-antecedents** slot of a personal pronoun, it then adds any NPs on ***table-of-antecedents*** that are not already on the pronoun's **:impossible-antecedents** slot to its **:possible-antecedents** slot. Finally, when **update-node** is finished processing a pronominal NP node, it adds it to ***table-of-proforms*** ([III.D]), for use in backwards pronominalization.

Note that, because our algorithm both establishes minimal domains and assigns possible and impossible antecedents during the course of the tree traversal, it can be single pass, in contrast to Chomsky's procedure, which assigned impossible antecedents in one traversal and checked for minimality during a second.

Since **update-node** is a general mechanism for adding or modifying information to a node on the basis of c-commanding constituents it is fairly straightforward to extend to handle other phenomena that involve c-command by modifying its top level **CASE** statement to dispatch on other categories. In fact, we have extended it in this manner to handle examples of "N anaphora"; i.e. cases where the head noun of a Noun Phrase is either "one" (which has been argued in Baker (1978) to be an anaphor for Ns, i.e. a noun and its complements, but not for full Noun Phrases) or phonologically null (∅), which seems to have the same possibilities for antecedents.

(20) Give me a list of ships which are in the gulf of Alaska that have casualty reports dated earlier than Esteem's oldest one.
(21) Is the Willamette's last problem rated worse than Wichita's ∅?

```
(when (pro-n-bar-p cfg-node)
      (loop for other-node
            in external-node-list
            (when (and (equal (category
                               other-node)
                              'NP)
                       (pro-n-bar-antecedent
                        other-node)
                       (add
                        (get-son-of-category
                         other-node 'N-BAR)
                        (possible-antecedents
                         cfg-node))))))
```

Figure 3-1: Algorithm for Pro N-BAR Anaphora

The addition to the algorithm that deals with this phenomenon is presented in Figure 3-1. This clause is considerably simpler that those that handle disjoint reference and co-reference phenomena for personal pronouns: only external nodes are involved and only forward antecedence is possible. This clause finds all the Noun Phrases that c-command an N pro-form and that are external to the current minimal domain. This excludes the possessive in a Noun Phrase such as "Esteem's oldest one" or "Wichita's ∅" from serving

[5] This table is filled in by Clause [III.D].

```
(defun pass-down-c-commanding-nodes (cfg-node external-node-list internal-node-list)
  (update-node cfg-node external-node-list internal-node-list)
  (cond ((finite-clause cfg-node)
         (let ((external-node-list (append internal-node-list external-node-list)))
           (loop for node in (children cfg-node)
                 (let ((internal-node-list (sisters node)))
                   (pass-down-c-commanding-nodes node
                                                 external-node-list
                                                 internal-node-list)))))
        ((equal (category cfg-node) 'NP)
         (cond ((equal (category (first (children cfg-node))) 'NP)
                (pass-down-c-commanding-nodes (first (children cfg-node))
                                              external-node-list
                                              internal-node-list)
                (let ((external-node-list (append external-node-list internal-node-list)))
                  (loop for node in (rest (children cfg-node))
                        (let ((internal-node-list (sisters node)))
                          (pass-down-c-commanding-nodes node
                                                        external-node-list
                                                        internal-node-list)))))
               (T (loop for node in (children cfg-node)
                        (let ((internal-node-list (append (sisters node) internal-node-list))
                          (pass-down-c-commanding-nodes node
                                                        external-node-list
                                                        internal-node-list))))))
        (T (loop for node in (children cfg-node)
                 (let ((internal-node-list (append (sisters node) internal-node-list)))
                   (pass-down-c-commanding-nodes node
                                                 external-node-list
                                                 internal-node-list)))))))
```

Figure 6-1: The Tree Walking Algorithm

```
(defun update-node (cfg-node external-node-list internal-node-list)
    (case (category cfg-node)
      (NP
        (cond ((non-pronominal cfg-node) [I]
               (loop for other-node in external-node-list [I.A]
                 (when (equal (category other-node) 'NP)
                   (add other-node (impossible-antecedents cfg-node))))
               (loop for other-node in internal-node-list [I.B]
                 (when (equal (category other-node) 'NP)
                   (add other-node (impossible-antecedents cfg-node))))
               (loop for pro in *table-of-proforms* [I.C]
                 (when (not (member pro (impossible-antecedents cfg-node)))
                   (add cfg-node (possible-antecedents pro))))
               (push cfg-node *table-of-antecedents*)) [I.D]
              ((bound-anaphor cfg-node) [II]
               (loop for other-node in internal-node-list
                 (when (equal (category other-node) 'NP)
                   (add other-node (possible-antecedents cfg-node)))))
              ((personal-pronoun cfg-node) [III]
               (loop for other-node in internal-node-list [III.A]
                 (when (equal (category other-node) 'NP)
                   (add other-node (impossible-antecedents cfg-node))))
               (loop for other-node in external-node-list [III.B]
                 (when (equal (category other-node) 'NP)
                   (add other-node (possible-antecedents cfg-node))))
               (loop for NP in *table-of-antecedents* [III.C]
                 (when (not (member NP (impossible-antecedents cfg-node)))
                   (add NP (possible-antecedents cfg-node))))
               (push cfg-node *table-of-proforms*))))))) [III.D]
```

Figure 6-2: The Reference Algorithm

as the antecedent to its pro-$\bar{N}$. External NPs that meet this criterion are filtered, since not all NPs can be antecedents of an $\bar{N}$ anaphor. For example, proper nouns cannot serve as such antecedents. Each NP that meets these criteria has its N-BAR added to the :possible-antecedents slot of the N-BAR node being processed.

4. INTERACTION WITH SEMANTIC INTERPRETATION

Syntactic constraints will not always identify just one allowable referent for a pronoun. Consider (22):

(22) The committee awarded the prize to itself.

Syntactically, "itself" in this sentence can refer to either "the prize" or "the committee". The additional use of semantic constraints is required to determine that the proper referent of the reflexive pronoun is "the committee".

Applying such constraints is the responsibility of the semantic interpretation component of our system. In the current implementation reported on here, semantic interpretation is applied after both parsing and the c-command tree-traversal have been performed. It is a two-stage process in which the first stage is concerned with "structural semantics"—the semantic consequence of syntactic structure—and the second stage with "lexical semantics"—the specific meanings of individual words with respect to a given application domain. This architecture for semantic interpretation was adopted from the PHLIQA1 system (Bronnenberg, et al. (1980)) and has been used in treating several difficult semantic phenomena (de Bruin and Scha (1988); Scha and Stallard (1988)).

The structural semantics stage operates on the parse tree to produce an expression of a language called "EFL" (for English-oriented Formal Language). This language is a higher-order intensional logic which includes a single descriptive constant for each word in the lexicon, however many senses that word may have. (From this standpoint, therefore, EFL is actually an ambiguous logical language.) Expressions of EFL are produced from the parse tree by a system of semantic rules, paired one-for-one with the syntactic rules of the grammar, which compute the EFL translation of a tree node from the EFL translations of its daughter nodes. The single EFL of a word is stored in its entry in the lexicon.

The lexical semantics stage operates on an expression of EFL to produce zero or more expressions of a language called "WML" (for World Model Language). WML is a higher-order intensional logic, with the same set of operations as EFL, but with unambiguous descriptive constants which correspond to the primitive concepts and relations of the particular application domain. WML expressions also have types, which are derived from the primitive disjoint categories of the application domain and which serve to delimit the set of meaningful WML expressions.

A set of translation rules pair ambiguous constants of EFL with one or more unambiguous expressions of WML. Translation to WML is performed by producing all possible combinations formed from replacing the EFL constants with their translations, and filtering to remove combinations which are disallowed by WML's type system. In this way selectional restrictions are represented and enforced.

The algorithms for producing EFL and WML are slightly modified in the case of anaphoric consituents: that is, reflexive pronouns, personal pronouns, and pro N-BARs. When the structural semantics component encounters an anaphoric constituent in the course of translating a parse tree to EFL, it creates a new EFL constant "on the fly" to serve as the EFL translation of this constituent. It marks this constant specially and attaches to it the EFL translations of the syntactically possible antecedents of the constituent, along with semantic type information (such as for gender) constraining the antecedents which make sense for it. If the constituent is a personal pronoun or pro N-BAR (but not a reflexive pronoun), a special constant of WML is also attached, marked with the EFL translations of the impossible antecedents of the constituent. This special WML constant represents the possibility of extra-sentential resolution of the anaphor.

The EFL to WML translation algorithm treats the anaphoric EFL constant specially, returning as its WML translations the translations of the "possible antecedents" that were attached in the EFL phase, together with the WML constant for extra-sentential reference (when this is appropriate). Expansion and filtering then proceed as described above.

(22) is handled as follows. We will suppose the following "domain model" of WML constants and types:

```
AWARD: (FUN (TUPLES AGENTS
                    VALUABLES AGENTS)
            TV)

SUB-TYPE(COMMITTEES,AGENTS)
SUB-TYPE(PRIZES,VALUABLES)

TYPE-INTERSECTION(VALUABLES,AGENTS)
     = NULL-SET
```

The structural semantics stage constructs the following clausal interpretation in EFL:

```
(AWARD (THE COMMITTEES) (THE PRIZES)
       ITSELF001)
```

where

```
ITSELF001 → (THE COMMITTEES)
            (THE PRIZES)
```

The combinatorially possible WML translations are the following, where anomally with respect to the type system is marked with a "*":

* (AWARD (THE COMMITTEES) (THE PRIZES)
(THE PRIZES))
(AWARD (THE COMMITTEES) (THE PRIZES)
(THE COMMITTEES))

The first interpretation is anomalous because the function "AWARD" is applied to an argument whose type is disjoint with the function's domain (in the third argument place). It is therefore discarded, leaving the second interpretation as the correct one.

A different example, in which a pronoun could have an extra-sentential antecedent, is:

(23) The committee awarded the prize to it.

In this case, neither NP inside the sentence is syntactically allowable as an antecedent of "it", and so only the extra-sentential possibility remains. The WML translation for (23) is:

(AWARD (THE COMMITTEE) (THE PRIZES) IT001)

where IT001 is a WML constant marked for disjoint reference:

IT001 ≠ (THE COMMITTEES)
≠ (THE PRIZES)

This information is necessary so that the module responsible for extra-sentential discourse can prevent *external* resolution of the pronoun to an internally (syntactically) forbidden antecedent—as could otherwise happen if "the committee" or "the prize" was mentioned in preceding discourse.

Unless the anaphoric constituent is a reflexive pronoun, an extra-sentential alternative will always be present as a WML translation option, and survive type filtering (since it is given the most general possible type). When both intra- and extra-sentential alternatives survive type filtering, our current heuristic is to prefer the intra-sentential one.

5. COMPARISON WITH RELATED WORK

Hobbs (1978) has done the only previous work we know of to use traversal of a syntactic parse tree to determine pronominal reference and we compare our algorithm with his in this section. Hobbs proposes a syntactic tree-traversal algorithm for pronominal reference that is "part of a larger left-to-right interpretation process" (Hobbs (1978, p. 318)). When a pronoun is encountered, the algorithm moves up to the nearest S or NP node (our "minimal domain nodes") that dominates the pronoun and searches to the left of the pronoun for any NP nodes that are dominated by an intervening S or NP node to propose as antecedents. The algorithm then proceeds up to the next NP or S node and searches to the left of the pronoun for any NP nodes to propose as antecedents. At this level, search is also made to the right for NP nodes to propose as antecedents. This will handle cases of backwards pronominalization, as in (18). However, this portion of the search is bounded; it does not seek antecedents below any NP or S nodes encountered. The search for c-commanding antecedents and antecedents for backwards pronominalization continues in this fashion until the top S is reached. At this point, preceding utterances in the discourse are searched, going from most recent to least recent. Each tree is searched in a left-to-right, breadth-first manner for NPs to propose as antecedents.

There are several differences between this algorithm and ours. The major one is that our algorithm is a single-pass, depth-first, exhaustive traversal whereas Hobbs' algorithm first walks down the tree, then up, and then back down and is not guaranteed to be exhaustive. Hobbs also imposes a "nearness" condition on the search for antecedents in the case of backwards pronominalization. However, as Hobbs points out, this restriction rules out the perfectly acceptable (24a) and (24b).

(24) a. Mary sacked out in [his] apartment before [Sam] could kick her out.
b. Girls who [he] has dated say that [Sam] is charming.

These examples show that the question of what the correct nearness constraint, if any, is remains open. Finally, Hobbs' algorithm handles both intra-sentential and extra-sentential pronominal reference relations, while ours is only intended to handle intra-sentential cases.

6. CURRENT STATUS AND FUTURE RESEARCH

In this section, we conclude by discussing some of the strengths and weaknesses of the current implementation and areas for future research. The shortcomings fall into two general categories: limitations of the implementation proper and limitations of the theory of pronominal reference that was implemented.

There are two general sorts of limitations to the mechanism described here: those that may be overcome by adding additional filtering devices to the basic tree-walking engine and those that may require a change in that basic engine. We begin with limitations of the first sort.

Currently, the algorithm does not do any checking on the potential antecedents of a pronoun or bound anaphora to see if they agree in person and number.[6] For bound anaphors, this is straightforward: a bound anaphor and its antecedent must agree in person and number. For personal pronouns, on the other hand,

[6]Currently, NPs are not specified for gender in our system, so this cannot be checked.

the situation is more complicated. In the singular, first ("I", "me"), second ("you"), and third ("he", "him", "she", "her", "it") personal pronouns require agreement in both person and number. In the plural, however, the number requirement is dropped because of "split antecedents" cases, in which more than one NP forms part of the antecedent of a pronoun, as in:

(25) [John] told [Bill] that [they] should leave.

where "John" and "Bill", together, antecede "they". Third person plural pronouns still require that each antecedent of a split antecedent itself be third person.

First person ("we", "us") and second person ("you") pronouns also allow split antecedents, but with looser person agreement requirements:

(26)a. [I] told [John] that [we] should go.
b. [I] told [you] that [we] should go.
c. [Bill] told [you] that [you] should go.
d. I told [you] that [you] should go.
e. John told Bill that we should go.
f. John told Bill that you should go.

Note that a first person plural pronoun allows split antecedents only if at least one of them is itself first person; contrast (26a) and (26b) with (26e). Similarly, a second person plural pronoun allows split antecedents only if at least one of them is also second person—contrast (26c) with (26f)—but not if one is first person; contrast (26c) with (26d).

While the constraints on singular and third person plural pronouns could be implemented as a local agreement check (e.g. as a pre-condition for being added to a pronoun's **:possible-antecedents** slot), the person agreement constraint on first and second person plural pronouns would require a separate post-process, since it is not a local constraint on individual split antecedents, but a global constraint on the set of them. Currently, since our algorithm imposes no agreement checks, it allows both the good cases of split antecedents as well as the impossible ones. We need to add the check to our algorithm and extend the semantics to also deal with split antecedents.

The algorithm also does not check for "crossover" cases. Roughly speaking, these are examples similar to backwards pronominalization cases such as (18) (repeated here as (27a)), in which the potential antecedent is a quantifier or a trace of a moved WH element. In such cases, overlapping reference is impossible. Contrast (27a) with (27b) and (27c).

(27) a. [His] mother loves [John].
b. His mother loves everyone.
c. Who does his mother love t_{who}?

These particular cases can be handled by adding a check to clause [I.C] to prohibit quantified NPs and WH-traces from participating in backwards pronominalization. However, the more general problem of how elements dislocated by WH movement or by topicalization interact with the algorithm given here is a topic that requires further work beyond this simple measure.

More seriously, there is also a well-known case of pronominal reference within NPs that is not handled by the algorithm. A constraint from the syntactic theory of reference implemented by our algorithm is that if the antecedent-anaphor relation holds between two positions, disjoint reference also holds between them; see examples (10) and (11), and (13) and (14). However, there is one position in English where this generalization is known not to hold: the possessive position of an NP. A bound anaphor is possible here, but a pronoun in the same position is not subject to disjoint reference; see (28):

(28) a. [The men] read [each other's] books.
b. [The men] read [their] books.

(28a) is correctly handled by the algorithm as already outlined; **pass-down-c-commanding-nodes** treats the nodes internal to the current minimal domain as internal nodes for the possessive in a Noun Phrase, so the NP "the men" will be added to the **:possible-antecedents** slot of a bound anaphor in this position. However, the same characteristics of the algorithm will also result in the NP "the men" being assigned to the **:impossible-antecedents** slot of "their" in (28b). One possible remedy for this situation is to add a clause to **update-node** that checks for possessive pronouns separately from other pronouns and that allows NPs both internal and external to the current minimal domain to be possible antecedents. However, the more far-reaching modifications proposed in the discussion below of the theory of pronominal reference would obviate this change.

There are several areas where our implementation points out problems with the structural theory of pronominal reference. The first of these is the definition of c-command itself.[7] Under Reinhart's (1976) original definition, a node A c-commands node B iff the branching node most immediately dominating A also dominates B and A does not dominate B. The difference between the two definitions can be seen in Figure 2-1; in addition to the c-command statements given there, Reinhart's definition adds the following:

E c-commands B, C, F, D, and G
F c-commands D and G
G c-commands C and F

These statements are true under Reinhart's definition of c-command, because no branching category intervenes between the c-commanding and c-commanded nodes, but not under that used in the implemented algorithm, since there is no sisterhood among the nodes. We have found this modified definition to be easier to implement; moreover, various researchers (e.g. Aoun and Sportiche (1983)) have pointed out problems with Reinhart's definition that the modified definition solves.

[7]Our algorithm uses a definition that is equivalent to the *in construction with* relation of Klima (1964, p. 297), which inspired c-command.

The implementation has also brought to light asymmetries in the strictness of c-command used to determine the antecedents of a bound anaphor and that used to determine the non-antecedents of a pronoun. In particular, none of the conjuncts of a conjoined NP can be the antecedent of a reflexive:

(29) *John and Mary like himself.

However, all of the conjuncts of a conjoined NP are impossible antecedents for any pronoun for which the entire conjoined NP is an impossible antecedent. In

(30) John and Mary like him.

"John" cannot be an antecedent of "him", despite the fact that "John" does not c-command "him". Contrast this with (19) where a non-c-commanding possessive *can* be the antecedent of a pronoun. This is handled correctly in the implementation. Whenever our algorithm adds a conjoined NP to the **:impossible-antecedents** slots of a pronoun or a non-pronominal NP, it adds all the conjuncts of that NP, as well. While this works, there is clearly something that is being missed here. Presumably, it should follow by definition that no individual conjunct of a conjoined NP can be a possible antecedent of a Noun Phrase with which the entire conjoined NP is disjoint in reference.[8]

A more serious problem with the theory of pronominal reference elaborated in Chomsky (1980) and (1981), and which our algorithm implements, is the crucial assumption that referentially dependent Noun Phrases can be exhaustively partitioned into bound anaphors vs. personal pronouns and that, therefore, they will be in complementary distribution. However, examples such as (28), as well as (31) (pointed out by Kuno (1987)) and (32) indicate that the notion of exhaustive partitioning of bound anaphors against personal pronouns is incorrect in the general case, even though it may be the typical state of affairs.

(31) a. [John] put the blanket under [himself].
b. [John] put the blanket under [him].
(32) a. I'll buy myself a beer.
b. I'll buy me a beer.

We can keep the insight of the structural theory of pronominal reference (i.e. that structural relations play a role in delimiting reference possibilities), while still incorporating these facts, if we give up the restriction that bound anaphors and personal pronouns are always in complementary distribution. One possible approach to this problem is to use feature decomposition to characterize bound anaphors and pronouns: the feature **±short-distance** indicates whether a pronominal can be used as a short-distance anaphor while the feature **±long-distance** indicates whether it can be used as a long-distance anaphor.[9]. While, in the normal case, personal pronouns in English are specified to be long-distance anaphors that cannot be used as short-distance anaphors, (i.e. as **[–short-distance +long-distance]**) this system would allow the feature governing a pronominal's use as a short-distance anaphor to be left free (i.e. as **?short-distance**) in certain syntactic contexts in English, such as the possessive position of a Noun Phrase, the object of certain prepositions, and the indirect object position of verbs.[10] Such a view of the syntax of personal pronouns could be implemented in a unification grammar fairly straightforwardly.

While such a treatment of personal pronouns as short-distance anaphors does not handle all the counter-examples to the syntactic theory of pronominal reference raised by researchers such as Kuno, it does begin to address them seriously. Clearly, it is more in accord with the facts than a theory that postulates an exhaustive partitioning of bound anaphors vs. personal pronouns, and so constitutes, in our opinion, a promising start towards handling the full range of pronoun reference facts in a reasonable manner.

Finally, we consider alternate ways of combining our pronominal reference mechanism with parsing and semantic interpretation. One possibility is a fully incremental architecture in which c-command constraints, semantic interpretations, and external reference resolution are computed simultaneously with the parse. Such an architecture might seem particularly attractive for processing large sets of alternatives, such as are encountered when processing spoken input. The intra-sentential reference phenomena described in this paper pose a problem for such an incremental approach, however. The possiblities for internal resolution for an anaphor cannot all be known locally to the anaphor, but must be obtained from elsewhere in the sentence. In many cases antecedents will lie to the left of the anaphor in the sentence, and thus will have been seen by a left-to-right parser by the time the anaphor is reached. But consider a case of backward pronominalization, as in (18), repeated here as (33):

(33) His mother loves John.

A wholly incremental mechanism, parsing the NP "his mother" first, would have to conclude that the referent of "his" was extra-sentential, since no intra-sentential referent was seen to the left. And if no extra-sentential referent could be found, the NP would have to be rejected. To be successful, such an incremental mechanism would have to be modified to include a kind of "lazy evaluation" which could rule out certain

[8]Thanks to Leland George for this insight, as well as for discussion of short and long distance anaphors.

[9]This is akin to the feature system **±anaphoric ±pronominal** of Chomsky (1981)

[10]This suggestion was originally made by Lust, et al. (1989) who support it on the basis of language acquisition data.

referents for an anaphor but never rule an anaphor empty of referents until utterance processing had been completed.

Another alternative would be to separate intra-sentential anaphor resolution from semantic interpretation, performing it instead in conjunction with extra-sentential discourse processing. A possible problem for this approach can be seen in sentences where the anaphor is combined with another ambiguous element, so that proliferation of semantic interpretations occur, as in:

(34) John's car is better than Bill's.

where the pro N-BAR, left completely unspecified during semantic interpretation, is free to generate all sorts of combinations with the possessive, including those in which the possession is appropriate to various "relational" interpretations of the pro N-BAR (de Bruin and Scha (1988)).

In future work, we plan to combine parsing and semantic interpretation into a single unification grammar incorporating semantic information in additional features. Part of that work will be to look for the optimal method of combining it with the pronominal reference mechanism presented here.

ACKNOWLEDGEMENTS

The work reported here was supported by the Advanced Research Projects Agency under Contract No. N00014-C-87-0085 monitored by the Office of Naval Research. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency of the United States Government.

REFERENCES

Aoun, Yousef and Dominique Sportiche (1983) "On the Formal Theory of Government", *The Linguistic Review* 2, pp. 211-236.

Ayuso, Damaris M. (1989) "Discourse Entities in Janus", *27th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ.

Baker, C.L. (1978) *Introduction to Generative-Transformational Syntax*, Prentice-Hall, Inc., Englewood Cliffs NJ.

Boisen S., Y. Chow, A. Haas, R. Ingria, S. Roucos, R. Scha, D. Stallard and M. Vilain (1989) *Integration of Speech and Natural Language: Final Report*, Report No. 6991, BBN Systems and Technologies Corporation, Cambridge, Massachusetts.

Bronnenberg, W.J.H.J., Harry C. Bunt, S.P. Jan Landsbergen, Remko J.H. Scha, W.J. Schoenmakers, and E.P.C. van Utteren (1980) "The Question Answering System PHLIQA1", in Leonard Bolc, ed., *Natural Language Question Answering Systems*, Hanser, Munich, pp. 217-305.

Chomsky, Noam (1980) "On Binding", *Linguistic Inquiry* 11.1, pp. 1-46.

Chomsky, Noam (1981) *Lectures on Government and Binding*, FORIS PUBLICATIONS, Dordrecht – Holland/Cinnaminson – U.S.A.

de Bruin, Jos and Remko Scha (1988) "The Interpretation of Relational Nouns", *26th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, pp. 25--32.

Hobbs, Jerry R. (1978) "Resolving Pronoun References", *Lingua* 44, pp. 311--338.

Jackendoff, Ray (1972) *Semantic Interpretation in Generative Grammar*, MIT Press, Cambridge, MA.

Klima, Edward S. (1964) "Negation in English", in J. A. Fodor and J. J. Katz, eds., *The Structure of Language: Readings in the Philosophy of Language*, Prentice-Hall, Englewood Cliffs, N. J.

Kuno, Susumu (1987) *Functional Syntax: Anaphora, Discourse, and Empathy*, The University of Chicago Press, Chicago and London.

Lasnik, Howard (1976) "Remarks on Coreference", *Linguistic Analysis* 2.1, pp. 1--22.

Lust, Barbara, Reiko Mazuka, Gita Martohardjono, and Jeong Me Yoon (1989) "On Parameter Setting in First Language Acquisition: The Case of the Binding Theory", Paper presented at The 12th GLOW Colloquium, Utrecht, April 5, 1989.

Reinhart, Tanya (1976) *The Syntactic Domain of Anaphora*, Ph.D. Dissertation, MIT, Cambridge, Massachusetts.

Scha, Remko and David Stallard (1988) "Multi-Level Plurals and Distributivity", *26th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ, pp. 17--24.