In *Proceedings of ACL-93*

# PRINCIPLE-BASED PARSING WITHOUT OVERGENERATION[1]

**Dekang Lin**
Department of Computing Science, University of Manitoba
Winnipeg, Manitoba, Canada, R3T 2N2
E-mail: lindek@cs.umanitoba.ca

## Abstract

Overgeneration is the main source of computational complexity in previous principle-based parsers. This paper presents a message passing algorithm for principle-based parsing that avoids the overgeneration problem. This algorithm has been implemented in C++ and successfully tested with example sentences from (van Riemsdijk and Williams, 1986).

## 1. Introduction

Unlike rule-based grammars that use a large number of rules to *describe* patterns in a language, Government-Binding (GB) Theory (Chomsky, 1981; Haegeman, 1991; van Riemsdijk and Williams, 1986) *explains* these patterns in terms of more foundmental and universal principles.

A key issue in building a principle-based parser is how to procedurally interpret the principles. Since GB principles are constraints over syntactic structures, one way to implement the principles is to

1. generate candidate structures of the sentence that satisfy X-bar theory and subcategorization frames of the words in the sentence.
2. filter out structures that violates any one of the principles.
3. the remaining structures are accepted as parse trees of the sentence.

This implementation of GB theory is very inefficient, since there are a large number of structures being generated and then filtered out. The problem of producing too many illicit structures is called *overgeneration* and has been recognized as the culprit of computational difficulties in principle-based parsing (Berwick, 1991). Many methods have been proposed to alleviate the overgeneration problem by detecting illicit structures as early as possible, such as optimal ordering of principles (Fong, 1991), coroutining (Dorr, 1991; Johnson, 1991).

This paper presents a principle-based parser that avoids the overgeneration problem by applying principles to descriptions of the structures, instead of the structures themselves. A structure for the input sentence is only constructed after its description has been found to satisfy all the principles. The structure can then be retrieved in time linear to its size and is guaranteed to be consistent with the principles.

Since the descriptions of structures are constant-sized attribute vectors, checking whether a structural description satisfy a principle takes constant amount of time. This compares favorably to approaches where constraint satisfaction involves tree traversal.

The next section presents a general framework for parsing by message passing. Section 3 shows how linguistic notions, such as dominance and government, can be translated into relationships between descriptions of structures. Section 4 describes interpretation of GB principles. Familiarity with GB theory is assumed in the presentation. Section 5 sketches an object-oriented implementation of the parser. Section 6 discusses complexity issues and related work.

## 2. Parsing by Message Passing

The message passing algorithm presented here is an extension to a message passing algorithm for context-free grammars (Lin and Goebel, 1993).

We encode the grammar, as well as the parser, in a network (Figure 1). The nodes in the networks represent syntactic categories. The links in the network represent dominance and subsumption relationships between the categories:

- There is a dominance link from node A to B if B can be immediately dominated by A. The dominance links can be further classified according to the type of dominance relationship.
- There is a specialization link from A to B if A subsumes B.

The network is also a parser. The nodes in the network are computing agents. They communicate

[1]The author wishes to thank the anonymous referees for their helpful comments and suggestions. This research was supported by Natural Sciences and Engineering Research Council of Canada grant OGP121338.

with each other by passing messages in the reverse direction of the links in the network.

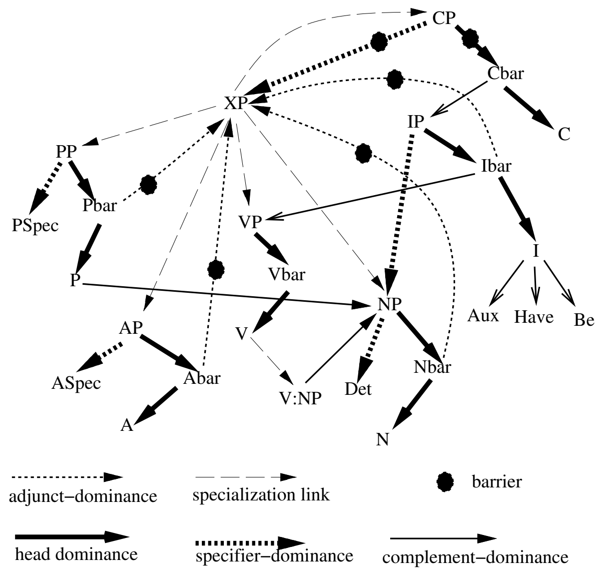



Figure 1: A Network Representation of Grammar

The messages contains items. An item is a triplet that describes a structure:
`<surface-string, attribute-values, sources>`, where

`surface-string` is an integer interval [i, j] denoting the i'th to j'th word in the input sentence.

`attribute-values` specify syntactic features, such as `cat, plu, case`, of the root node of the structure described by the item.

`sources` component is the set of items that describe the immediate sub-structures. Therefore, by tracing the sources of an item, a complete structure can be retrieved.

The location of the item in the network determines the syntactic category of the structure.

For example, [$_{NP}$ the ice-cream] in the sentence "the ice-cream was eaten" is represented by an item $i_4$ at NP node (see Figure 2):

`<[0,1], ((cat n) -plu (nform norm)`
`-cm +theta), {`$i_1$`, `$i_3$`}>`

An item represents the root node of a structure and contains enough information such that the internal nodes of the structure are irrelevant.

The message passing process is initiated by sending initial items externally to lexical nodes (e.g., N, P, ...). The initial items represent the words in the sentence. The attribute values of these items are obtained from the lexicon.

In case of lexical ambiguity, each possibility is represented by an item. For example, suppose the input sentence is "I saw a man," then the word "saw" is represented by the following two items sent to nodes N and V:NP[2] respectively:

```
<[1,1], ((cat n) -plu (nform norm)), {}>
<[1,1], ((cat v) (cform fin) -pas
        (tense past)), {}>
```

When a node receives an item, it attempts to combine the item with items from other nodes to form new items. Two items

<[$i_1$,$j_1$], $A_1$, $S_1$> and <[$i_2$,$j_2$], $A_2$, $S_2$>

can be combined if

1. their surface strings are adjacent to each other: $i_2 = j_1+1$.
2. their attribute values $A_1$ and $A_2$ are unifiable.
3. their sources are disjoint: $S_1 \cap S_2 = \emptyset$.

The result of the combination is a new item:

<[$i_1$,$j_2$], unify($A_1$, $A_2$), $S_1 \cup S_2$>.

The new items represent larger parse trees resulted from combining smaller ones. They are then propagated further to other nodes.

The principles in GB theory are implemented as a set of constraints that must be satisfied during the propagation and combination of items. The constraints are attached to nodes and links in the network. Different nodes and links may have different constraints. The items received or created by a node must satisfy the constraints at the node.

The constraints attached to the links serve as filters. A link only allows items that satisfy its constraints to pass through. For example, the link from V:NP to NP in Figure 1 has a constraint that any item passing through it must be unifiable with `(case acc)`. Thus items representing NPs with nominative case, such as "he", will not be able to pass through the link.

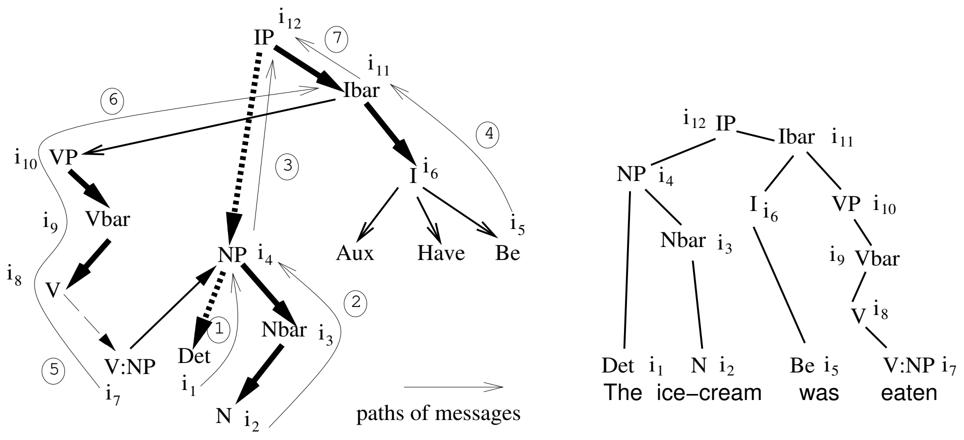
By default, the attributes of an item percolate with the item as it is sent across a link. However, the links in the network may block the percolation of certain attributes.

The sentence is successfully parsed if an item is found at IP or CP node whose surface string is the input sentence. A parse tree of the sentence can be retrieved by tracing the sources of the item.

### An example

The message passing process for analyzing the sentence

[2] V:NP denotes verbs taking an NP complement. Similarly, V:IP denotes verbs taking a CP complement, N:CP represents nouns taking a CP complement.




a. The message passing process

b. The parse tree retrieved

$i_1$ =<[0,0], ((cat d)), {}>

$i_2$ =<[1,1], ((cat n) -plu (nform norm) +theta), {}>

$i_3$ =<[1,1], ((cat n) -plu (nform norm) +theta), {$i_2$}>

$i_4$ =<[0,1], ((cat n) -plu (nform norm) -cm +theta), {$i_1$, $i_3$}>

$i_5$ =<[2,2], ((cat i) -plu (per 1 3) (cform fin) +be +ca +govern (tense past)), {}>

$i_6$ =<[2,2], ((cat i) -plu (per 1 3) (cform fin) +be +ca +govern (tense past)), {$i_5$}>

$i_7$ =<[3,3], ((cat v) +pas), {}>

$i_8$ =<[3,3], ((cat v) +pas +nppg -npbarrier (np-atts NNORM)), {$i_7$}>

$i_9$ =<[3,3], ((cat v) +pas +nppg -npbarrier (np-atts NNORM)), {$i_8$}>

$i_{10}$=<[3,3], ((cat v) +pas +nppg -npbarrier (np-atts NNORM)), {$i_9$}>

$i_{11}$=<[2,3], ((cat i) +pas +nppg -npbarrier (np-atts NNORM) (per 1 3) (cform fin) +ca +govern (tense past))), {$i_6$, $i_{10}$}>

$i_{12}$=<[0,3], ((cat i) +pas (per 1 3) (cform fin) +ca +govern (tense past)), {$i_4$, $i_{11}$}>

Figure 2: Parsing the sentence "The ice-cream was eaten"

(1) The ice-cream was eaten

is illustrated in Figure 2.a. In order not to convolute the figure, we have only shown the items that are involved in the parse tree of the sentence and their propagation paths.

The parsing process is described as follows:

1. The item $i_1$ is created by looking up the lexicon for the word "the" and is sent to the node Det, which sends a copy of $i_1$ to NP.

2. The item $i_2$ is sent to N, which propagates it to Nbar. The attribute values of $i_2$ are percolated to $i_3$. The source component of $i_3$ is $\{i_2\}$. Item $i_3$ is then sent to NP node.

3. When NP receives $i_3$ from Nbar, $i_3$ is combined with $i_1$ from Det to form a new item $i_4$. One of the constraints at NP node is:

   if `(nform norm)` then `-cm`,

   which means that normal NPs need to be case-marked. Therefore, $i_4$ acquires `-cm`. Item $i_4$ is then sent to nodes that have links to NP.

4. The word "was" is represented by item $i_5$, which is sent to Ibar via I.

5. The word "eaten" can be either the past participle or the passive voice of "eat". The second possibility is represented by the item $i_7$. The word belongs to the subcategory V:NP which takes an NP as the complement. Therefore, the item $i_7$ is sent to node V:NP.

6. Since $i_7$ has the attribute `+pas` (passive voice), an np-movement is generated at V:NP. The movement is represented by the attributes `nppg`, `npbarrier`, and `np-atts`. The first two attributes are used to make sure that the movement is consistent with GB principles. The value of `np-atts` is an attribute vector, which must be unifiable with the antecedent of this np-movement. `NNORM` is a shorthand for `(cat n) (nform norm)`.

7. When Ibar receives $i_{10}$, which is propagated to VP from V:NP, the item is combined with

$i_6$ from I to form $i_{11}$.

8. When IP receives $i_{11}$, it is combined with $i_4$ from NP to form $i_{12}$. Since $i_{11}$ contains an np-movement whose `np-atts` attribute is unifiable with $i_4$, $i_4$ is identified as the antecedent of np-movement. The np-movement attributes in $i_{12}$ are cleared.

The sources of $i_{12}$ are $i_4$ from NP and $i_{11}$ from Ibar. Therefore, the top-level of parse tree consists of an NP and Ibar node dominated by IP node. The complete parse tree (Figure 2.b) is obtained by recursively tracing the origins of $i_4$ and $i_{11}$ from NP and Ibar respectively. The trace after "eaten" is indicated by the np-movement attributes of $i_7$, even though the tree does not include a node representing the trace.

## 3. Modeling Linguistics Devices

GB principles are stated in terms of linguistic concepts such as barrier, government and movement, which are relationships between nodes in syntactic structures. Since we interpret the principles with descriptions of the structures, instead of the structures themselves, we must be able to model these notions with the descriptions.

### Dominance and m-command:

Dominance and m-command are relationships between nodes in syntactic structures. Since an item represent a node in a syntactic structure, relationships between the nodes can be represented by relationships between items:

**dominance:** An item dominates its direct and indirect sources. For example, in Figure 2, $i_4$ dominates $i_1$, $i_2$, and $i_3$.

**m-command:** The head daughter of an item representing a maximal category m-commands non-head daughters of the item and their sources.

### Barrier

Chomsky (1986) proposed the notion of barrier to unify the treatment of government and subjacency. In Chomsky's proposal, barrierhood is a property of maximal nodes (nodes representing maximal categories). However, not every maximal node is a barrier. The barrierhood of a node also depends on its context, in terms of L-marking and inheritance.

Instead of making barrierhood a property of the *nodes* in syntactic structures, we define it to be a property of *links* in the grammar network. That is, certain links in the grammar network are classified as barriers. In Figure 1, barrier links have a black ink-spot on them. Barrierhood is a property of these links, independent of the context. This definition of barrier is simpler than Chomsky's since it is context-free. In our experiments so far, this simpler definition has been found to be adequate.

### Government

Once the notion of barrier has been defined, the government relationship between two nodes in a structure can be defined as follows:

**government:** A governs B if A is the minimal governor that m-commands B via a sequence of non-barrier links, where governors are N, V, P, A, and tensed I.

Items representing governors are assigned `+govern` attribute. This attribute percolates across head dominance links. If an item has `+govern` attribute, then non-head sources of the item and their sources are governed by the head of the item if there are paths between them and the item satisfying the conditions:

1. there is no barrier on the path.
2. there is no other item with `+govern` attribute on the path (minimality condition (Chomsky, 1986, p.10)).

### Movement:[3]

Movement is a major source of complexity in principle-based parsing. Directly modeling Move-$\alpha$ would obviously generate a large number of invalid movements. Fortunately, movements must also satisfy:

**c-command condition:** A moved element must c-command its trace (Radford, 1988, p.564), where A c-command B if A does not dominate B but the parent of A dominates B.

The c-command condition implies that a movement consists of a sequence of moves in the reverse direction of dominance links, except the last one. Therefore, we can model a movement with a set of attribute values. If an item contains these attribute values, it means that there is a movement out of the structure represented by the item. For example, in Figure 2.b, item $i_{10}$ contains movement attributes: `nppg`, `npbarrier` and `np-atts`. This indicates that there is an np-movement out of the VP whose root node is $i_{10}$.

[3] We limit the discussion to np-movements and wh-movements whose initial traces are in argument positions.

The movement attributes are generated at the parent node of the initial trace. For example, V:NP is a node representing normal transitive verbs which take an NP as complement. When V:NP receives an item representing the passive sense of the word *eaten*, V:NP creates another item

```
<[i,i], ((cat v) -npbarrier +nppg
         (np-atts (cat n))), {}>
```

This item will not be combined with any item from NP node because the NP complement is assumed to be an np-trace. The item is then sent to nodes dominating V:NP. As the item propagates further, the attributes is carried with it, simulating the effect of movement. The np-movement land at IP node when the IP node combines an item from subject NP and another item from Ibar with np-movement attributes. A precondition on the landing is that the attributes of the former can be unified with the value of `np-atts` of the latter. Wh-movements are dealt with by attributes `whpg, whbarrier, wh-atts.`

This treatment of movement requires that the parent node of a initial trace be able to determine the type of movement. When a movement is generated, the type of the movement depends on the `ca` (case assigner) attribute of the item:

| ca | movement | examples |
|---|---|---|
| + | wh | active V, P, finite IP |
| - | np | A, passive V, non-finite IP |

For example, when IP node receives an item from Ibar, IP attempts to combine it with another item from subject NP. If the subject is not found, then the IP node generates a movement. If the item represent a finite clause, then it has attributes `+ca (cform fin)` and the movement is of type wh. Otherwise, the movement is of type np.

## 4. Interpretation of Principles

We now describe how the principles of GB theory are implemented.

**X-bar Theory:**

1. Every syntactic category is a projection of a lexical head.
2. There two levels of projection of lexical heads. Only the bar-2 projections can be complements and adjuncts.

The first condition requires that every non-lexical category have a head. This is guaranteed by a constraint in item combination: one of the sources of the two items being combined must be from the head daughter.

The second condition is implemented by the structure of the grammar network. The combinations of items represent constructions of larger parse trees from smaller ones. Since the structure of the grammar network satisfies the constraint, the parse trees constructed by item combination also satisfy the X-bar theory.

**Case Filter:** Every lexical NP must be case-marked, where A case-marks B iff A is a case assigner and A governs B (Haegeman, 1991, p.156).

The case filter is implemented as follows:

1. Case assigners (P, active V, tensed I) have `+ca` attribute. Governors that are not case assigners (N, A, passive V) have `-ca` attribute.
2. Every item at NP node is assigned an attribute value `-cm`, which means that the item needs to be case-marked. The `-cm` attribute then propagates with the item. This item is said to be the origin of the `-cm` attribute.
3. Barrier links do not allow any item with `-cm` to pass through, because, once the item goes beyond the barrier, the origin of `-cm` will not be governed, let alone case-marked.
4. Since each node has at most one governor, if the governor is not a case assigner, the node will not be case-marked. Therefore, a case-filter violation is detected if `+govern -cm -ca` co-occur in an item.
5. If `+govern +ca -cm` co-occur in an item, then the head daughter of the item governs and case-marks the origin of `-cm`. The case-filter condition on the origin of `-cm` is met. The `-cm` attribute is cleared.

For example, consider the following sentences:

(2) a. I believe John to have left.
b. *It was believed John to have left.
c. I would hope for John to leave.
d. *I would hope John to leave.

The word "believe" belongs to a subcategory of verb (V:IP) that takes an IP as the complement. Since there is no barrier between V:IP and the subject of IP, words like "believe" can govern into the IP complement and case-mark its subject (known as *exceptional case-marking* in literature). In (2a), the `-cm` attribute assigned to the item representing $[_{NP}$ John] percolates to V:IP node without being blocked by any barrier. Since `+govern +ca -cm` co-occur in the item at V:IP node, the case-filter is satisfied (Figure 3.a). On the other hand, in (2b) the pas-

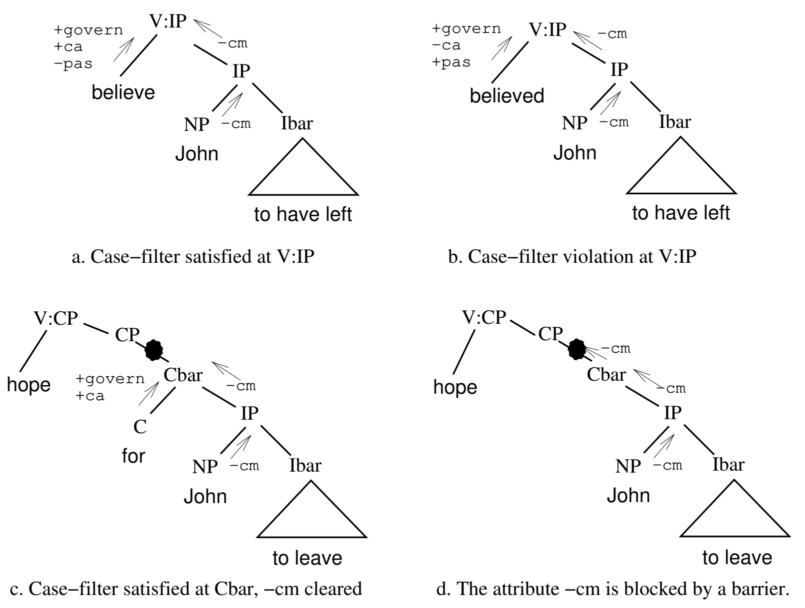



Figure 3: Case Filter Examples

sive "believed" is not a case-assigner. The case-filter violation is detected at V:IP node (Figure 3.b).

The word "hope" takes a CP complement. It does not govern the subject of CP because there is a barrier between them. The subject of an infinitive CP can only be governed by complement "for" (Figure 3.c and 3.d).

**$\theta$-criterion:** Every chain must receive and one only one $\theta$-role, where a chain consists of an NP and the traces (if any) coindexed with it (van Riemsdijk and Williams, 1986, p.245).

We first consider chains consisting of one element. The $\theta$-criterion is implemented as the following constraints:

1. An item at NP node is assigned `+theta` if its `nform` attribute is `norm`. Otherwise, if the value of `nform` is `there` or `it`, then the item is assigned `-theta`.
2. Lexical nodes assign `+theta` or `-theta` to items depending on whether they are $\theta$-assigners (V, A, P) or not (N, C).
3. Verbs and adjectives also have a `subj-theta` attribute.

| value | $\theta$-role* | examples |
|---|---|---|
| `+subj-theta` | yes | "take", "sleep" |
| `-subj-theta` | no | "seem", passive verbs |

*assigning $\theta$-role to subject

This attribute percolates with the item from V to IP. The IP node then check the value of `theta` and `subj-theta` to make sure that the verb assigns a $\theta$-role to the subject if it requires one, and vice versa.

Figure 4 shows an example of $\theta$-criterion in action when parsing:

(3) *It loves Mary

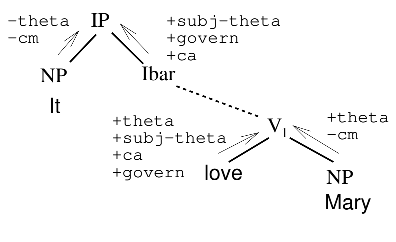



Figure 4: $\theta$-criterion in action

The subject NP, "it", has attribute `-theta`, which is percolated to the IP node. The verb "love" has attributes `+theta +subj-theta`. The NP, "Mary", has attribute `+theta`. When the items representing "love" and "Mary" are combined. Their `theta` attribute are unifiable, thus satisfying the $\theta$-criterion. The `+subj-theta` attribute of "love" percolates with the item representing "love Mary", which is propagated to IP node. When the item from NP and Ibar are combined at IP node, the new item has both `-theta` and `+subj-theta` attribute, resulting in a $\theta$-criterion violation.

The above constraints guarantee that chains with only one element satisfy $\theta$-criterion. We now consider chains with more than one element. The base-position of a wh-movement is case-marked and assigned a $\theta$-role. The base position of an np-movement is assigned a $\theta$-role, but not case-marked. To ensure that the movement chains satisfy $\theta$-criterion we need only to make sure that the items representing the parents of intermediate traces and landing sites of the movements satisfy these conditions:

- None of `+ca`, `+theta` and `+subj-theta` is present in the items representing the parent of intermediate traces of (wh- and np-) movements as well as the landing sites of wh-movements, thus these positions are not case-marked and assigned a $\theta$-role.
- Both `+ca` and `+subj-theta` are present in the items representing parents of landing sites of np-movements.

**Subjacency:** Movement cannot cross more than one barrier (Haegeman, 1991, p.494).

A wh-movement carries a `whbarrier` attribute. The value `-whbarrier` means that the movement has not crossed any barrier and `+whbarrier` means that the movement has already crossed one barrier. Barrier links allow items with `-whbarrier` to pass through, but change the value to `+whbarrier`. Items with `+whbarrier` are blocked by barrier links. When a wh-movement leaves an intermediate trace at a position, the corresponding `whbarrier` becomes `-`.

The subjacency of np-movements is similarly handled with a `npbarrier` attribute.

**Empty Category Principle (ECP):** A trace or its parent must be properly governed.

In literature, proper government is not, as the term suggests, subsumed by government. For example, in

(4) Who do you think [$_{\text{CP}}$ $e'$ [$_{\text{IP}}$ $e$ came]]

the tensed I in [$_{\text{IP}}$ $e$ came] governs but does not properly govern the trace $e$. On the other hand, $e'$ properly governs but does not govern $e$ (Haegeman, 1991, p.456).

Here, we define proper government to be a subclass of government:

**Proper government:** A properly governs B iff A governs B and A is a $\theta$-role assigner (A do not have to assign $\theta$-role to B).

Therefore, if an item have both `+govern` and one of `+theta` or `+subj-theta`, then the head of the item properly governs the non-head source items and their sources that are reachable via a sequence of non-barrier links. This definition unifies the notions of government and proper government. In (4), $e$ is properly governed by tensed I, $e'$ is properly governed by "think".

This definition won't be able to account for difference between (4) and (5) (That-Trace Effect (Haegeman, 1991, p.456)):

(5) *Who do you think [$_{\text{CP}}$ $e'$ that [$_{\text{IP}}$ $e$ came]]

However, That-Trace Effect can be explained by a separate principle.

The proper government of wh-traces are handled by an attribute `whpg` (np-movements are similarly dealt with by an `nppg` attribute):

| Value | Meaning |
|---|---|
| `-whpg` | the most recent trace has yet to be properly governed. |
| `+whpg` | the most recent trace has already been properly governed. |

1. If an item has the attributes `-whpg, -theta, +govern`, then the item is an ECP violation, because the governor of the trace is not a $\theta$-role assigner. If an item has attributes `-whpg, +theta, +govern`, then the trace is properly governed. The value of `whpg` is changed to `+`.
2. Whenever a wh-movement leaves an intermediate trace, `whpg` becomes `-`.
3. Barrier links block items with `-whpg`.

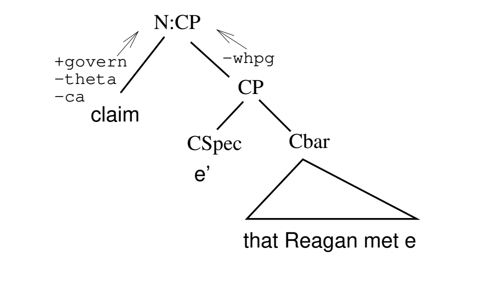



Figure 5: An example of ECP violation

For example, the word *claim* takes a CP complement. In the sentence:

(6) *Who$_i$ did you make the claim $e'_i$ that Reagan met $e_i$

there is a wh-movement out of the complement CP of *claim*. When the movement left an intermediate trace at CSpec, the value of `whpg` became `-`. When the item with `-whpg` is combined with the item representing *claim*, their unification has attributes (`+govern -theta -whpg`), which is an ECP violation. The item is recognized as invalid and discarded.

**PRO Theorem:** PRO must be ungoverned (Haegeman, 1991, p.263).

When the IP node receives an item from Ibar with `cform` not being `fin`, the node makes a copy of the item and assign `+pro` and `-ppro` to the copy and then send it further without combining it with any item from (subject) NP node. The attribute `+pro` represents the hypothesis that the subject of the clause is PRO. The meaning of `-ppro` is that the subject PRO has not yet been protected (from being governed).

When an item containing `-ppro` passes through a barrier link, `-ppro` becomes `+ppro` which means that the PRO subject has now been protected. A PRO-theorem violation is detected if `+govern` and `-ppro` co-occur in an item.

## 5. Objected-oriented Implementation

The parser has been implemented in C++, an object-oriented extension of C. The object-oriented paradigm makes the relationships between nodes and links in the grammar network and their software counterparts explicit and direct. Communication via message passing is reflected in the message passing metaphor used in object-oriented languages.

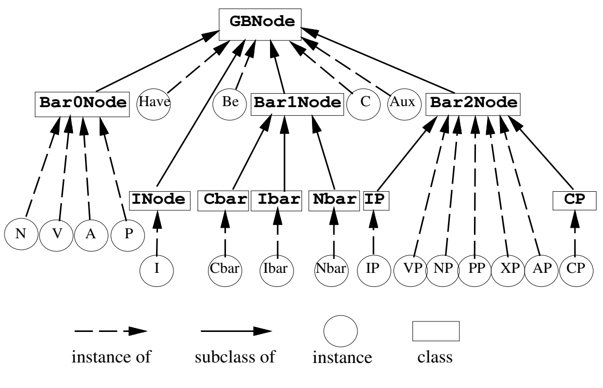



Figure 6: The class hierarchy for nodes

Nodes and links are implemented as objects. Figure 6 shows the class hierarchy for nodes. The constraints that implement the principles are distributed over the nodes and links in the network. The implementation of the constraints is modular because they are defined in class definitions and all the instances of the class and its subclasses inherit these constraints. The object-oriented paradigm allows the subclasses to modify the constraints.

The implementation of the parser has been tested with example sentences from Chapters 4–10, 15–18 of (van Riemsdijk and Williams, 1986). The chapters left out are mostly about logical form and Binding Theory, which have not yet been implemented in the parser. The average parsing time for sentences with 5 to 20 words is below half of a second on a SPARCstation ELC.

## 6. Discussion and Related Work

### Complexity of unification

The attribute vectors used here are similar to those in unification based grammars/parsers. An important difference, however, is that the attribute vectors used here satisfy the *unit closure* condition (Barton, Jr. et al., 1987, p.257). That is, non-atomic attribute values are vectors that consist only of atomic attribute values. For example:

(7) a. `((cat v) +pas +whpg (wh-atts (cat p))`
b. * `((cat v) +pas +whpg (wh-atts (cat v) (np-att (cat n))))`

(7a) satisfies the unit closure condition, whereas (7b) does not, because `wh-atts` in (7b) contains a non-atomic attribute `np-atts`. (Barton, Jr. et al., 1987) argued that the unification of recursive attribute structures is a major source of computational complexity. On the other hand, let $a$ be the number of atomic attributes, $n$ be the number of non-atomic attributes. The time it takes to unify two attribute vectors is $a + na$ if they satisfy the unit closure condition. Since both $n$ and $a$ can be regarded as constants, the unification takes only constant amount of time. In our current implementation, $n = 2$, $a = 59$.

### Attribute grammar interpretation

Correa (1991) proposed an interpretation of GB principles based on attribute grammars. An attribute grammar consists of a phrase structure grammar and a set of attribution rules to compute the attribute values of the non-terminal symbols. The attributes are evaluated after a parse tree has been constructed by the phrase structure grammar. The original objective of attribute grammar is to derive the semantics of programs from parse trees. Since programming languages are designed to be unambiguous, the attribution rules need to be evaluated on only one parse tree. In attribute grammar interpretation of GB theory, the principles are encoded in the attribution rules, and the phrase structure grammar is replaced by X-bar theory and Move-$\alpha$. Therefore, a large number of structures will be constructed and evaluated by the attribution rules, thus leading to a serious overgeneration problem. For this reason, Correa pointed out that the attribute grammar interpretation should be used as a specification of an implementation, rather than an implementation itself.

### Actor-based GB parsing

Abney and Cole (1986) presented a GB parser that uses actors (Agha, 1986). Actors are similar to objects in having internal states and responding to messages. In our model, each syntactic category is represented by an object. In (Abney and Cole, 1986), each instance of a category is represented by an actor. The actors build structures by creating other actors and their relationships according to $\theta$-assignment, predication, and functional-selection. Other principles are then used to filter out illicit structures. such as subjacency and case-filter. This generate-and-test nature of the algorithm makes it suscetible to the overgeneration problem.

## 7. Conclusion

We have presented an efficient message passing algorithm for principle-based parsing, where

- overgeneration is avoided by interpreting principles in terms of descriptions of structures;
- constraint checking involves only a constant-sized attribute vector;
- principles are checked in different orders at different places so that stricter principles are applied earlier.

We have also proposed simplifications of GB theory with regard to barrier and proper government, which have been found to be adequate in our experiments so far.

## References


Abney, S. and Cole, J. (1986). A government-binding parser. In *Proceedings of NELS*.

Agha, G. A. (1986). *Actors: a model of concurrent computation in distributed system*. MIT Press, Cambridge, MA.

Barton, Jr., G. E., Berwick, R. C., and Ristad, E. S. (1987). *Computational Complexity and Natural Language*. The MIT Press, Cambridge, Massachusetts.

Berwick, R. C. (1991). Principles of principle-based parsing. In Berwick, B. C., Abney, S. P., and Tenny, C., editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 1–38. Kluwer Academic Publishers.

Chomsky, N. (1981). *Lectures on Government and Binding*. Foris Publications, Cinnaminson, USA.

Chomsky, N. (1986). *Barriers*. Linguistic Inquiry Monographs. The MIT Press, Cambridge, MA.

Correa, N. (1991). Empty categories, chains, and parsing. In Berwick, B. C., Abney, S. P., and Tenny, C., editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 83–121. Kluwer Academic Publishers.

Dorr, B. J. (1991). Principle-based parsing for machine translation. In Berwick, B. C., Abney, S. P., and Tenny, C., editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 153–184. Kluwer Academic Publishers.

Fong, S. (1991). The computational implementation of principle-based parsers. In Berwick, B. C., Abney, S. P., and Tenny, C., editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 65–82. Kluwer Academic Publishers.

Haegeman, L. (1991). *Introduction to Government and Binding Theory*. Basil Blackwell Ltd.

Johnson, M. (1991). Deductive parsing: The use of knowledge of language. In Berwick, B. C., Abney, S. P., and Tenny, C., editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 39–64. Kluwer Academic Publishers.

Lin, D. and Goebel, R. (1993). Contex-free grammar parsing by message passing. In *Proceedings of PACLING-93*, Vancouver, BC.

Radford, A. (1988). *Transformational Grammar*. Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge, England.

van Riemsdijk, H. and Williams, E. (1986). *Introduction to the Theory of Grammar*. Current Studies in Linguistics. The MIT Press, Cambridge, Massachusetts.