Parsing with Principles: Predicting a Phrasal Node Before Its Head Appears[1,2]

Edward Gibson
Department of Philosophy
Carnegie Mellon University
Pittsburgh, PA 15213
eafg@cad.cs.cmu.edu

1 Introduction

Recent work in generative syntactic theory has shifted the conception of a natural language grammar from a homogeneous set of phrase structure (PS) rules to a heterogeneous set of well-formedness constraints on representations (see, for example, Chomsky (1981), Stowell (1981), Chomsky (1986a) and Pollard & Sag (1987)). In these theories it is assumed that the grammar contains principles that are independent of the language being parsed, together with principles that are parameterized to reflect the varying behavior of different languages. However, there is more to a theory of human sentence processing than just a theory of linguistic competence. A theory of performance consists of both linguistic knowledge and a parsing algorithm. This paper will investigate ways of exploiting principle-based syntactic theories directly in a parsing algorithm in order to determine whether or not a principle-based parsing algorithm can be compatible with psycholinguistic evidence.

Principle-based parsing is an interesting research topic not only from a psycholinguistic point of view but also from a practical point of view. When PS rules are used, a separate grammar must be written for each language parsed. Each of these grammars contains a great deal of redundant information. For example, there may be two rules, in different grammars, that are identical except for the order of the constituents on the right hand side, indicating a difference in word order. This redundancy can be avoided by employing a universal phrase structure component (not necessarily in the form of rules) along with parameters and associated values. A principles and parameters approach provides a single compact grammar for all languages that would otherwise be represented by many different (and redundant) PS grammars.

Any model of human parsing must dictate: a) how structures are projected from the lexicon; b) how structures are attached to one another; and c) what constraints affect the resultant structures. This paper will concentrate on the first two components with respect to principle-based parsing algorithms: node projection and structure attachment.

Two basic control structures exist for any parsing algorithm: data-driven control and hypothesis-driven control. Even if a parser is predominantly hypothesis-driven, the predictions that it makes must at some point be compared with the data that are presented to it. Some data-driven component is therefore necessary for any parsing algorithm. Thus, a reasonable hypothesis to test is that the human parsing algorithm is entirely data-driven. This is exactly the approach that is taken by a number of principle-based parsing algorithms (see, for example, Abney (1986), Kashket (1987), Gibson & Clark (1987) and Pritchett (1987)). These parsing algorithms each include a node projection algorithm that projects an input word to a maximal category, but does not cause the projection of any further nodes.

Although this simple strategy is attractive because of its simplicity, it turns out that it cannot account for certain phenomena observed in the processing of Dutch (Frazier (1987): see Section 2.1). A completely data-driven node projection algorithm also has difficulty accounting for the processing ease of adjective-noun constructions in English (see Section 2.2). As a result of this evidence, a purely data-driven node projection

---

[1] Paper presented at the International Workshop on Parsing Technologies, August 28-31, 1989.

[2] I would like to thank Robin Clark, Rick Kazman, Howard Kurtzman, Eric Nyberg and Brad Pritchett for their comments on earlier drafts of this paper, and I offer the usual disclaimer.

algorithm must be rejected in favor of a node projection algorithm that has a predictive (hypothesis-driven) component Frazier (1987)).

This paper describes a node projection algorithm that is part of the Constrained Parallel Parser (CPP) (Gibson (1987), Gibson & Clark (1987) and Clark & Gibson (1988)). This parser is based on the principles of Government-Binding theory (Chomsky (1981, 1986a)). Section 3.1 gives an overview of the CPP model, while Section 3.2 describes the node projection algorithm. Section 3.3 describes the attachment algorithm, and includes an example parse. These node projection and attachment algorithms demonstrate that a principle-based parsing algorithm can account for the Dutch and English data, while avoiding the existence of redundant phrase structure rules. Thus it is concluded that one should continue to investigate hypothesis-driven principle-based models in the search for an optimal psycholinguistic model.

2 Data-Driven Node Projection: Empirical Predictions and Results

2.1 Evidence from Dutch

Consider the sentence fragment in (1):

(1)
... dat het meisje van Holland ...
... "that the girl from Holland" ...

Dutch is like English in that prepositional phrase modifiers of nouns may follow the noun. Thus the prepositional phrase *van Holland* may be a modifier of the noun phrase *the girl* in example (1). Unlike English, however, Dutch is SOV in subordinate clauses. Hence in (1) the prepositional phrase *van Holland* may also be the argument of a verb to follow. In particular, if the word *glimlachte* ("smiled") follows the fragment in (1), then the prepositional phrase *van Holland* can attach to the noun phrase that it follows, since the verb *glimlachte* has no lexical requirements (see (2a)). If, on the other hand, the word *houdt* ("likes") follows the fragment in (1), then the PP *van Holland* must attach as argument of the verb *houdt*, since the verb requires such a complement (see (2b)).

(2)
a. ... dat [$_S$ [$_{NP}$ het meisje [$_{PP}$ van Holland]] [$_{VP}$ glimlachte]]
... "that the girl from Holland smiled" ...
b. ... dat [$_S$ [$_{NP}$ het meisje] [$_{VP}$ [$_{V'}$ [$_{PP}$ van Holland] [$_V$ houdt]]]]
... "that the girl likes Holland"

Following Abney (1986), Frazier (1987), Clark & Gibson (1988) and numerous others, it is assumed that attached structures are preferred over unattached structures. If we also assume that a phrasal node is not projected until its head is encountered, we predict that people will entertain only one hypothesis for the sentence fragment in (1): the modifier attachment. Thus we predict that it should take longer to parse the continuation *houdt* ("likes") than to parse the continuation *glimlachte* ("smiled"), since the continuation *houdt* forces the prepositional phrase to be reanalyzed as an argument of the verb. However, contrary to this prediction, the verb that allows argument attachment is actually parsed faster than the verb that necessitates modifier attachment in sentence fragments like (1). If the verb had been projected before its head was encountered, then the argument attachment of the PP *van Holland* would be possible at the same time that the modifier attachment is possible.[3] Thus Frazier concludes that in some cases phrasal nodes must be projected before their lexical heads have been encountered.

[3] It is beyond the scope of this paper to offer an explanation as to why the argument attachment is in fact *preferred* to the modifier attachment. This paper seeks only to demonstrate that the argument attachment possibility must at least be *available* for a psychologically real parser. See Abney (1986), Frazier (1987) and Clark & Gibson (1988) for possible explanations for the preference phenomenon.

2.2 Evidence from English

A second piece of evidence against this limited type of node projection is provided by the processing of noun phrases in English that have more than one pre-head constituent.

It is assumed that the primitive operation of attachment is associated with a certain processing cost. Hence the amount of time taken to parse a single input word is directly related to the number of attachments that the parser must execute to incorporate that structure into the existing structure(s). If a phrasal node is not projected until its head is encountered, then parsing the final word of a head-final construction will involve attaching all its pre-head structures at that point. If, in addition, there is more than one pre-head structure and no attachments are possible until the head appears, then a significant proportion of processing time should be spent in processing the head.

The hypothesis that a phrasal node is not projected until its head is encountered can be tested with the English noun phrase, since the head of an English noun phrase appears after a specifier and any adjectival modifiers. For example, consider the English noun phrase *the big red book*. First, the word *the* is read and a determiner phrase is built. Since it is assumed that nodes are not projected until their heads are encountered, no noun phrase is built at this point. The word *big* is now read and causes the projection of an adjective phrase. Attachments are now attempted between the two structures built thus far. Neither of the categories can be argument, specifier or modifier for the other, so no attachment is possible. The next word *red* now causes the projection of an adjective phrase, and once again no attachments are possible. Only when the word *book* is read and projected to a noun phrase can attachments take place. First the adjective phrase representing *red* attaches as a modifier of the noun phrase *book*. Then the AP representing *big* attaches as a modifier of the noun phrase just constructed. Finally the determiner phrase representing *the* attaches as specifier of the noun phrase *big red book*.

Thus if we assume that a phrasal node is not projected until its head is parsed, we predict that a greater number of attachments will take place in parsing the head than in parsing any other word in the noun phrase. Since it is assumed that an attachment is a significant parser operation, it is predicted that people should take more time parsing the head of the noun phrase than they take parsing the other words of the noun phrase. Since there is no psycholinguistic evidence that people take more time to process heads in head-final constructions, I hypothesize that phrasal nodes are being projected before their heads are being encountered.

3 Hypothesizing a Phrasal Node Before Its Head Appears

3.1 The Parsing Model: The Constrained Parallel Parser

This paper assumes the Constrained Parallel Parser (CPP) as its model of human sentence processing (see Gibson (1987), Gibson & Clark (1987) and Clark & Gibson (1988)). The CPP model is based on the principles of Government-Binding Theory (Chomsky (1981, 1986a)); crucially CPP has no separate module containing language-particular rules. Following Marcus (1980), structures parsed under the CPP model are placed on a stack and the most recently built structures are placed in a data structure called the *buffer*. The parser builds structure by attaching nodes in the buffer to nodes on top of the stack. Unlike Marcus' model, the CPP model allows multiple representations for the same input string to exist in a buffer or stack cell. Although multiple representations for the same input string are permitted, constraints on parallelism frequently cause one representation to be preferred over the others. Motivation for the parallel hypothesis comes from garden path effects and perception of ambiguity in addition to relative processing load effects. For information on the particular constraints and their motivations, see Gibson & Clark (1987), Clark & Gibson (1988) and the references cited in these papers.

3.1.1 Lexical Entries for CPP

A lexical entry accessed by CPP consists of, among other things, a *theta-grid*. A theta-grid is an unordered list of *theta structures*. Each theta structure consists of a thematic role and associated subcategorization formation. One theta structure in a theta-grid may be marked as *indirect* to refer to its subject. For example, the word *shout* might have the following theta-grid:[4]

(3)

```
((Subcat = PREP, Thematic-Role = GOAL)
 (Subcat = COMP, Thematic-Role = PROPOSITION))
```

When the word *shout* (or an inflected variant of *shout*) is encountered in an input phrase, the thematic role *agent* will be assigned to its subject, as long as this subject is a noun phrase. The direct thematic roles *goal* and *proposition* will be assigned to prepositional and complementizer phrases respectively, as long as each is present. Since the order of theta structures in a theta-grid is not relevant to its use in parsing, the above theta-grid for *shout* will be sufficient to parse both sentences in (4).

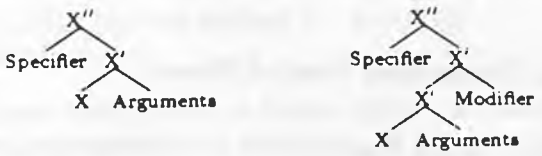
(4)

a. The man shouts [$_{PP}$ to the woman] [$_{CP}$ that Ernie sees the rock]

b. The man shouts [$_{CP}$ that Ernie sees the rock] [$_{PP}$ to the woman]

3.1.2 $\overline{X}$ Theory in CPP

The CPP model assumes $\overline{X}$ Theory as present in Chomsky (1986b). $\overline{X}$ Theory has two basic principles: first, each tree structure must have a head; and second, each structure must have a maximal projection. As a result of these principles and other principles, (*e.g.*, the θ–Criterion, the Extended Projection Principle, Case Theory), the positions of arguments, specifiers and modifiers with respect to the head of a given structure are limited. In particular, a specifier may only appear as a sister to the one-bar projection below a maximal projection, and the head, along with its arguments, must appear below the one-bar projection. The orders of the specifier and arguments relative to the head is language dependent. For example, the basic structure of English categories is shown below. Furthermore, binary branching is assumed (Kayne (1983)), so that modifiers are Chomsky-adjoined to the two-bar or one-bar levels, giving one possible structure for a post-head modifier below on the right.



3.1.3 The CPP Parsing Algorithm

The CPP algorithm is essentially very simple. A word is projected via node projection (see Section 3.2) into the buffer. If attachments are possible between the buffer and the top of the stack, then the results of these attachments are placed into the buffer and the stack is popped. Attachments are attempted again until no longer possible. This entire procedure is repeated for each word in the input string. The formal CPP algorithm is given below:

1. (Initializations) Set the stack to nil. Set the buffer to nil.

[4] In a more complete theory, a syntactic category would be determined from the thematic role (Chomsky (1986a)).

2. (Ending Condition) If the end of the input string has been reached and the buffer is empty then return the contents of the stack and stop.

3. If the buffer is empty then project nodes for each lexical entry corresponding to the next word in the input string, and put this list of maximal projections into the buffer.

4. Make all possible attachments between the stack and the buffer, subject to the attachment constraints (see Clark & Gibson (1988)). Put the attached structures in the buffer. If no attachments are possible, then put the contents of the buffer on top of the stack.

5. Go to 2.

3.2 The Projection of Nodes from the Lexicon

Node projection proceeds as follows. First a lexical item is projected to a phrasal node: a *Confirmed* node (*C-node*). Following $\overline{X}$ Theory, each lexical entry for a given word is projected maximally. For example, the word *rock*, which has both a noun and a verb entry would be projected to at least two maximal projections:

(5)
a. $[_{NP}\ [_{N'}\ [_{N}$ rock $]]]$
b. $[_{VP}\ [_{V'}\ [_{V}$ rock $]]]$

Next, the parser hypothesizes nodes whose heads may appear immediately to the right of the given C-node. These predicted structures are called *hypothesized* nodes or *H-nodes*. An H-node is defined to be any node whose head is to the right of all lexical input. In order to determine which H-node structures to hypothesize from a given C-node, it is necessary to consult the argument properties associated with the C-node together with the specifier and modifier properties of the nodal category and the word order properties of the language in question. It is assumed that the ability of one category to act as specifier, modifier or argument of another category is part of unparameterized Universal Grammar. On the other hand, the relative order of two categories is assumed to be parameterized across different languages. For example, a determiner phrase, if it exists in a given language, is universally allowable as a specifier of a noun phrase. Whether the determiner appears before or after its head noun depends on the language-particular values associated with the parameters that determine word order.

Three parameters are proposed to account for variation in word order, one for each of argument, specifier and modifier projections.[5] For each language, each of these parameters is associated with at least one value, where the parameter values come from the following set: {*head*, *satellite*}.[6] The value *head* indicates that a category C causes the projection to the right of those categories for which C may be head. Thus this value indicates head-initial word order. The value *satellite* indicates that a category C causes the projection to the right of those categories for which C may be a satellite category. Hence this value indicates head-final word order. H-node projection from a category C is defined in (6):

(6)
(Argument, Specifier, Modifier) H-Node Projection from category C: If the value associated with the (argument, specifier, modifier) projection parameter is *head*, then cause the projection of (argument, specifier, modifier) satellites, and attach them to the right below the appropriate projection of C. If the value associated with the (argument, specifier, modifier) projection parameter is *satellite*, then cause the projection of (argument, specifier, modifier) heads, and attach them to the right above the appropriate projection of C.

In English the argument projection parameter is set to *head*, so that arguments appear after the head. Hence, if a lexical entry has requirements that must be filled, then structures corresponding to subcategorized

[5] Furthermore, it is assumed that the value of the modifier projection parameter defaults to the value of the argument projection parameter.

[6] I will use the term *satellite* to indicate non-head constituents: arguments, specifiers and modifiers.

categories are hypothesized and attached. For example, the verb *see* subcategorizes for a noun phrase, so an empty noun phrase node is hypothesized and attached as argument of the verb:

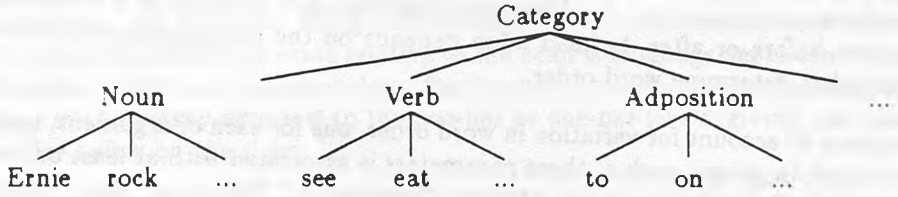
(7)
[$_{VP}$ [$_{V'}$ [$_{V}$ see] [$_{NP}$ e]]]

The specifier projection parameter, on the other hand, is set to the value *satellite* in English so that specifiers appear before their heads. If the category associated with a C-node is an allowable specifier for other categories, then an H-node projection of each of these categories is built and the C-node specifier is attached to each. For example, since a determiner may specify a noun phrase, an H-node noun phrase is hypothesized when parsing a determiner in English:

(8)
[$_{NP}$ [$_{DetP}$ [$_{Det'}$ [$_{Det}$ the]]] [$_{N'}$ [$_{N}$ e]]]

Thus the node projection algorithm provides a new derivation of language-particular word order. In previous principle-based systems, word order is derived from parameterized direction of attachment (see Gibson & Clark (1987), Nyberg (1987), Wehrli (1988)). An attachment takes place from buffer to stack in head-initial constructions and from stack to buffer in head-final constructions. Since attachment is now a uniform operation as defined in (17), this parameterization is no longer necessary. Instead, in head-initial constructions, nodes now project to the nodes that they may immediately dominate. In head-final constructions, nodes now project to those nodes that they may be immediately dominated by.

The projection parameters as defined in (6) account for many facts about word order across languages. However, most, if not all, languages have cases that do not fit this clean picture. For example, while modifiers in English are predominantly post-head, adjectives appear before the head. A single global value for modifier projection predicts that this situation is impossible. Hence we must assume that the values given for the projection parameters are only default values. In order to formalize this idea, I assume the existence of a hierarchy of categories and words as shown below:



It is assumed that the value for each of the projection parameters is the default value for that projection type with respect to a particular language. However, a particular category or word may have a value associated with it for a projection parameter in addition to the default one. If this is the case, then only the most specific value is used. For example, in English, the category adjective is associated with the value *satellite* with respect to modifier projection. Thus English adjectives appear before the head. The adjective *tall* will therefore cause the projection of both a C-node adjective phrase and an H-node noun phrase:

(9)
a. [$_{AP}$ tall]
b. [$_{NP}$ [$_{N'}$ [$_{AP}$ tall] [$_{N'}$ [$_{N}$ e]]]]

If recursive application of projection to H-nodes were allowed, then it would be possible, in principle, to project an infinite number of nodes from a single lexical entry. In English, for example, a genitive noun phrase can specify another noun phrase. This noun phrase may also be a genitive noun phrase, and so on. If H-nodes could project to further H-nodes, then it would be necessary to hypothesize an infinite number of genitive NP H-nodes for every genitive NP that is read. As a result of this difficulty, the H-node Projection Constraint is proposed:

(10)
The H-node Projection Constraint: Only a C-node may cause the projection of an H-node.

As a result of the H-node Projection Constraint, H-nodes may not invoke H-node projection. For example, if a specifier causes the projection of its head, the resulting head cannot then cause the projection of those categories that it may specify. As a result, the number of nodes that may be projected from a single lexical item is severely restricted.

3.3 Node Attachment

Given the above node projection algorithm, it is necessary to define an algorithm for attachment of nodes. Since structures are predicted by the node projection algorithm, the attachment algorithm must dictate how subsequent structures match these predictions. Consider the following two examples from English: the first is an example of specifier attachment; the second is an example of argument attachment. In English, specifiers precede the head and arguments follow the head. It is desirable for the attachment algorithm to handle both kinds of attachments without word order particular stipulations.

First, suppose that the word *the* is on the stack as both a determiner phrase and an H-node noun phrase. Furthermore, suppose that the word *woman* is projected into the buffer as both a noun phrase and an H-node clausal phrase:[7]

(11)
Stack: $[_{DetP}\ [_{Det'}\ [_{Det}$ the $]]]$
$[_{NP}\ [_{DetP}\ [_{Det'}\ [_{Det}$ the $]]]\ [_{N'}\ [_{N}\ e\ ]]]$
Buffer: $[_{NP}\ [_{N'}\ [_{N}$ woman $]]]$
$[_{XP_{clause}}\ [_{NP}\ [_{N'}\ [_{N}$ woman $]]]\ [_{X'_{clause}}\ [_{X_{clause}}\ e\ ]]]$

The attachment algorithm should allow two attachments at this point: the H-node NP on the stack uniting with each NP C-node in the buffer. It might also seem reasonable to allow the bare determiner phrase to attach directly as specifier of each noun phrase. However, this kind of attachment is undesirable for two reasons. First of all, it makes the attachment operation a disjunctive operation: an attachment would involve *either* matching an H-node *or* meeting the satellite requirements of a category. Second of all, it makes H-node projection unnecessary in most situations and therefore somewhat stipulative. That is, allowing a disjunctive attachment operation would permit many derivations that never use an H-node, so that the need for H-nodes would be restricted to head-final constructions with pre-head satellites (see Section 2). It is therefore desirable for all attachments to involve matching an H-node.

Two structures should be returned after attachments in (11): a C-node noun phrase and an H-node clausal phrase:

(12)
a. $[_{NP}\ [_{DetP}$ the $]\ [_{N'}\ [_{N}$ woman $]]]$
b. $[_{XP_{clause}}\ [_{NP}\ [_{DetP}$ the $]\ [_{N'}\ [_{N}$ woman $]]]\ [_{X'_{clause}}\ [_{X_{clause}}\ e\ ]]]$

Now consider an English argument attachment. Suppose that a prepositional phrase representing the word *beside* is on the stack and the noun *Frank* is represented in the buffer as a noun phrase and a clausal phrase:

(13)
Stack: $[_{PP}\ [_{P'}\ [_{P}$ beside $]\ [_{NP}\ e\ ]]]$
Buffer: $[_{NP}\ [_{N'}\ [_{N}$ Frank $]]]$
$[_{XP_{clause}}\ [_{NP}\ [_{N'}\ [_{N}$ Frank $]]]\ [_{X'_{clause}}\ [_{X_{clause}}\ e\ ]]]$

[7] A noun phrase is projected to an H-node clausal (or predicate) phrase since nouns may be the subjects of predicates.

Since the preposition *beside* subcategorizes for a noun phrase, there is an H-node NP attached as its object. The attachment algorithm should allow a single attachment at this point: the noun phrase representing *Frank* uniting with the H-node NP object of *beside*:

(14)
$[_{PP}\ [_{P'}\ [_{P}$ beside] $[_{NP}$ Frank]]]

As should be clear from the two examples, the process of attachment involves comparing a previously predicted category with a current category. If the two categories are *compatible*, then attachment may be viable.

3.3.1 Node Compatibility

Compatibility is defined in terms of *unification*, which is defined terms of *subsumption*.[8] A structure X is said to *subsume* a structure Y if X is more general than Y. That Y contains less specific information than Y. So, for example, a structure that is specified as *clausal* (e.g. t ıead of a predicate), but is not specified for a particular category subsumes a structure having the category *verb*, since verbs are predicative and thus clausal categories. Hence structure (15a) subsumes structure (15b):

(15)
a. $[_{XP_{clause}}\ [_{X'_{clause}}\ [_{X_{clause}}\ e\ ]]]$
b. $[_{VP}\ [_{V'}\ [_{V}$ walk]]]

The *unification* operation is the least upper bound operator in the subsumption ordering on information in a structure. Since structure (15a) subsumes structure (15b), the result of unifying structure (15a) with structure (15b) is structure (15b). Two structures are *compatible* if the unification of the two structures is non-nil. The information on a structure that is relevant to attachment consists of the node's bar level (*e.g.*, zero level, intermediate or maximal), and the node's lexical features, (*e.g.* category, case, *etc*).

3.3.2 Attachment

Roughly speaking, the attachment operation should locate an H-node in a structure on the stack along with a compatible node in a structure in the buffer. If both of these structures have parent tree structures, then these parent tree structures must also be compatible. In order to keep the process of attachment simple, it is proposed that each attachment have at most one compatibility check. This constraint is given in (16):[9]

(16)
Attachment Constraint: At most one nontrivial lexical feature unification is permitted per attachment.

A nontrivial unification is one that involves two nontrivial structures; a trivial unification is one that involves at least one trivial structure. For example, if the parent node of the buffer site is as of yet undefined, then the parent node of the stack site trivially unifies with this parent node. Only when both parents are defined is there a nontrivial unification.

Consider the effect of the following three requirements: first, the lexical features of the stack and buffer attachment sites must be compatible; second, the tree structures above the buffer and stack attachment sites must be compatible; and third, at most one lexical feature unification is permissible per derivation, (16). Since any attachment must involve at least one nontrivial lexical feature unification, that of the stack and buffer sites, any additional nontrivial unifications will violate the attachment constraint in (16). If both

[8] See Sheiber (1986) for background on the possible uses of unification in particular grammar formalisms.

[9] In fact, this constraint follows from the two assumptions: first, a compatibility check takes a certain amount of processing time; and second, attachments that take less time are preferred over those that take more time. See Gibson (forthcoming) for further discussion.

the buffer and stack attachment sites have parent tree structures, then the lexical features of these parents will need to be unified. Since the child structures will also need to be unified, (16) will be violated. Thus it follows that, in an attachment, either the buffer site or the stack site has no parent tree structure.[10]

Since the order of the words in the input must be maintained in a final parse, only those nodes in a buffer structure that dominate all lexical items in that structure are permissible as attachment sites. For example, suppose that the buffer contained a representation for the noun phrase *women in college*. Furthermore, suppose that there is an H-node NP on the stack representing the word *the*. Although it would be suitable for the buffer structure representing the entire noun phrase *women in college* to match the stack H-node, it would not be suitable for the C-node NP representing *college* to match this H-node. This attachment would result in a structure that moved the lexical input *women in* to the left of the lexical input dominated by the matched H-node, producing a parse for the input *women in the college*. Since the word order of the input string must be maintained, sites for buffer attachment must dominate all lexical items in the buffer structure.

Once suitable maximal projections in each of the buffer and stack structures have been identified for matching, it is still necessary to check that their internal structures are compatible. For example, suppose that an identified buffer site is a C-node whose head allows exactly one specifier and a specifier is already attached. If the stack H-node site also contains a specifier, then the attachment should be blocked. On the other hand, if the stack H-node site does not contain a specifier, and other requirements are satisfied, then the attachment should be allowed.

Testing for internal structure compatibility is quite simple if all tree structures are assumed to be binary branching ones. The only possible attachment sites inside the stack H-node are those nodes that dominate no other nodes. As long as there is some buffer node that both dominates all the buffer input and matches the H-node attachment site for bar level, then the attachment is possible.

Attachment is formally defined in (17):

(17)
A structure W in the buffer can attach to a structure X on the stack iff all of (a), (b), (c), (d) and (e) are true:

a. Structure W contains a maximal projection node, Y, such that Y dominates all lexical material in W;
b. Structure X contains a maximal projection H-node structure, Z;
c. The tree structure above Y is compatible with the tree structure above Z, subject to the attachment constraint in (16);
d. The lexical features of structure Y are compatible with the lexical features of structure Z;
e. Structure Y is *bar-level compatible* with structure Z.

Bar-level compatibility is defined in (18):

(18)
A structure U in the buffer is *bar-level compatible* with a structure V on the stack iff all of (a), (b) and (c) are true:

a. Structure U contains a node, S, such that S dominates all lexical material in U;
b. Structure V contains an H-node structure, T, that dominates no lexical material;
c. The bar level of S is compatible with the bar level of T.

If attachment is viable, then W contains a structure Y that is bar-level compatible with a structure Z that is part of X. Since Y and Z are bar-level compatible, there are structures S and T inside Y and Z

[10] It might seem that some possible attachments are being thrown away at this point. That is, in principle, there might be a structure that can only be formed by attaching a buffer site to a stack site where both sites have parent tree structures. This attachment would be blocked by (16). However, it turns out that any attachment that could have been formed by an attachment involving more than one lexical feature unification can always be arrived at by a different attachment involving a single lexical feature unification. For the proof, see Gibson (forthcoming).

respectively, that satisfy the conditions of bar-level compatibility, (18).

When the conditions for attachment are satisfied, structures W and X are united in the following way. First, W and X are copied to nodes W' and X' respectively. Inside X' there is a node, Z', that is a copy of Z. The lexical features of Z' are set to the unification of the lexical features of structures Y and Z. Next, structure T' in Z' (corresponding to structure T in Z) is replaced by S', the copy of structure S inside W. The bar level of T' is set to the unification of the bar levels of structures S and T.

Finally, the tree structures above Y and Z are unified and this tree structure is attached above Z'. That is, if Z has some parent tree structure and Y does not, then the copy of this structure inside X' is attached above Z'. Similarly, if Y has some parent tree structure and Z does not, then the copy of this structure inside W is attached above Z'. If neither node has any parent tree structure (*i.e.*, $W = Y$, $X = Z$), then the unification is trivial and no attachment is made. Since Y and Z cannot both have parent tree structures (see (16) and the discussion following it), unifying the parent tree structures is a very simple process.

3.3.3. Example Attachments

As an illustration of how attachments take place, consider once again the noun phrase *the big red book*. First the determiner *the* is read and is projected to a C-node determiner phrase. Since a determiner is allowable as the specifier of a noun phrase and specifiers occur before the head in English, an H-node NP is also built. These two structures are depicted in (19):

(19)
a. $[_{DetP}$ the]
b. $[_{NP}\ [_{DetP}$ the] $[_{N'_1}\ [_{N}\ e\ ]]]$

Since there is nothing on the stack, these structures are shifted to the top of the stack. The word *big* projects to both a C-node AP and an H-node NP since an adjective is allowable as a pre-head modifier in English. These two structures are placed in the buffer (depicted in (20)).

(20)
a. $[_{AP}$ big]
b. $[_{NP}\ [_{N'_2}\ [_{AP}$ big] $[_{N'}\ [_{N}\ e\ ]]]]$

An attachment between nodes (19b) and (20b) is now attempted. Note that: a) node (20b) is a maximal projection dominating all lexical material in its buffer structure; b) node (19b) is a maximal projection H-node on the stack; c) the tree structures above these two nodes are compatible (both are undefined); and d) the categories of the two nodes are compatible. It remains to check for bar-level compatibility of the two structures. Since: a) the N'_2 in structure (20b) dominates all the buffer input; b) the H-node N'_1 in structure (19b) dominates no C-nodes; and c) N'_1 and N'_2 are compatible in bar level, the structures in (19b) and (20b) can be attached. The two structures are therefore attached by uniting N'_1 and N'_2. The resultant structure is given in (21):

(21)
$[_{NP}\ [_{DetP}$ the] $[_{N'}\ [_{AP}$ big] $[_{N'_3}\ [_{N}\ e\ ]]]]$

Structure (21), the only possible attachment between the buffer and the stack, is placed back in the buffer, and the stack is popped. Since there is now nothing left on the stack, no further attachments are possible at this time. Structure (21) is thus shifted to the stack. The word *red* now enters the buffer as a C-node adjective phrase and an H-node noun phrase:

(22)
a. $[_{AP}$ red]
b. $[_{NP}\ [_{N'_4}\ [_{AP}$ red] $[_{N'}\ [_{N}\ e\ ]]]]$

An attachment between nodes (21) and (22b) is now attempted. Requirements (17a)-(17d) are satisfied and the requirement for bar-level compatibility is satisfied by the node labeled N'_3 in (21) together with N'_4 in (22b). Hence the structures are united, giving (23):

(23)
[$_{NP}$ [$_{DetP}$ the] [$_{N'}$ [$_{AP}$ big] [$_{N'}$ [$_{AP}$ red] [$_{N'_3}$ [$_{N}$ e]]]]]

Since (23) is the only possible attachment between the buffer and the stack, it is placed in the buffer and the stack is popped. Since the stack is now empty, structure (23) shifts to the stack. The noun *book* now enters the buffer as both a C-node noun phrase and an H-node clausal phrase:

(24)
a. [$_{NP}$ [$_{N'}$ [$_{N}$ book]]]
b. [$_{XP_{clause}}$ [$_{NP}$ [$_{N'}$ [$_{N}$ book]]] [$_{X'_{clause}}$ [$_{X_{clause}}$ e]]]

Two attachments are possible at this point. The NP structure in (23) unites with each NP C-node on the stack, resulting in the structures in (25):

(25)
a. [$_{NP}$ [$_{DetP}$ the] [$_{N'}$ [$_{AP}$ big] [$_{N'}$ [$_{AP}$ red] [$_{N'}$ [$_{N'}$ [$_{N}$ book]] [$_{PP}$ e] [$_{CP}$ e]]]]]
b. [$_{XP_{clause}}$ [$_{NP}$ the big red book] [$_{X'_{clause}}$ [$_{X_{clause}}$ e]]]

Note that only one attachment per structure takes place in the final parse step. Crucially, no more attachments per structure take place when parsing the head of the noun phrase than when parsing the pre-head constituents in the noun phrase.[11] Thus, in contrast with the situation when nodes are only projected when their heads are encountered, the node projection and attachment algorithms described here predict that there should not be any slowdown when parsing the head of a head-final construction.

The Dutch data described in Section 2.1 are handled in a similar manner.

4 Conclusions

This paper has described a) a principle-based algorithm for the projection of phrasal nodes before their heads are parsed, and b) an algorithm for attaching the predicted nodes. It is worthwhile to compare the new projection algorithm with algorithms that do not project H-nodes. The projection algorithm provided here involves more work and hence, on the surface, may seem somewhat stipulative compared to one that does not project H-nodes. However, it turns out that although projecting to H-nodes is more complicated than not doing so, attachment when H-nodes are not present is more complicated than attachment when they are present. That is, if a projection algorithm causes the projection of H-nodes, it will have a more complicated attachment algorithm. For example, if H-nodes are projected when parsing the noun phrase *the woman*, the determiner *the* is immediately projected to an H-node noun phrase, which leads to a simple attachment. If H-nodes are not projected, then projection is easier, but attachment is that much more complicated. When attaching, it will be necessary to check if a determiner is an allowable specifier of a noun phrase: the same operation that is performed when projecting to H-nodes. Thus although the complexity of particular components changes , the complexity of the entire parsing algorithm does not change, whether or not H-nodes are projected. Since the proposed projection and attachment algorithms make better empirical predictions than ones that do not predict structure, the new algorithms are preferred.

---

[11] Note that it is the number of attachments per structure that is crucial here, and not the number of total attachments, since attachments made upon two independent structures may be performed in parallel, whereas attachments made on the same structure must be performed serially. For example, since structures (24a) and (24b) are independent, attachments may be made to each of these in parallel. But if an attachment, *B* relies on the result of another attachment *A*, then attachment *A* must be performed first.

5 References

Abney (1986), "Licensing and Parsing", *Proceedings of the Seventeenth North East Linguistic Society Conference*, MIT, Cambridge, MA.

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

Chomsky, N. (1986a), *Knowledge of Language: Its Nature, Origin and Use*, Praeger Publishers, New York, NY.

Chomsky, N. (1986b), *Barriers*, Linguistic Inquiry Monograph 13, MIT Press, Cambridge, MA.

Clark, R. & Gibson, E. (1988), "A Parallel Model for Adult Sentence Processing", *Proceedings of the Tenth Cognitive Science Conference*, McGill University, Montreal, Quebec.

Frazier, L. (1987) "Syntactic Processing Evidence from Dutch", *Natural Language and Linguistic Theory* 5, pp. 519-559.

Gibson, E. (1987), *Garden-Path Effects in a Parser with Parallel Architecture*, Eastern States Conference on Linguistics, Columbus Ohio.

Gibson, E. (forthcoming), *Parsing with Principles: A Computational Theory of Human Sentence Processing*, Ms., Carnegie Mellon University, Pittsburgh, PA.

Gibson, E. & Clark, R. (1987), "Positing Gaps in a Parallel Parser", *Proceedings of the Eighteenth North East Linguistic Society Conference*, University of Toronto, Toronto, Ontario.

Kashket, M. (1987), *Government-Binding Parser for Warlpiri, a Free Word Order Language*, MIT Master's Thesis, Cambridge, MA.

Kayne, R. (1983) *Connectedness and Binary Branching*, Foris, Dordrecht, The Netherlands.

Marcus, M. (1980), *A Theory of Syntactic Recognition for Natural Language*, MIT Press, Cambridge, MA.

Nyberg, E. (1987), "Parsing and and the Acquisition of Word Order", *Proceedings of the Fourth Eastern States Conference on Linguistics*, The Ohio State University, Columbus, OH.

Pollard, C. & Sag, I. (1987) *An Information-based Syntax and Semantics*, CSLI Lecture Notes Number 13, Menlo Park, CA.

Pritchett, B. (1987), *Garden Path Phenomena and the Grammatical Basis of Language Processing*, Harvard University Ph.D. dissertation, Cambridge, MA.

Sheiber, S. (1986) *An Introduction to Unification-based Approaches to Grammar*, CSLI Lecture Notes Number 4, Menlo Park, CA.

Stowell, T. (1981), *Origins of Phrase Structure*, MIT Ph.D. dissertation.

Wehrli, E. (1988), "Parsing with a GB Grammar", in U. Reyle and C. Rohrer (eds.), *Natural Language Parsing and Linguistic Theories*, 177-201, Reidel, Dordrecht, the Netherlands.